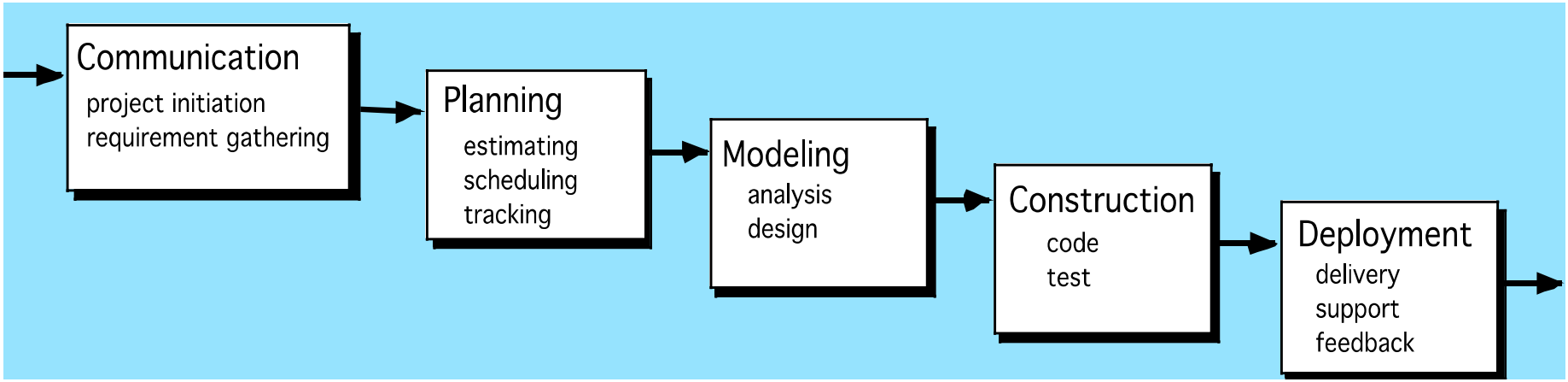
Traditional Software Process Models

Waterfall

- It is the oldest paradigm for Software Engineering.
- When requirements are well defined and reasonably stable, it leads to a linear fashion.
- The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction and deployment, culminating in ongoing support of the completed software.
- When to select?
- There are times when the requirements for a problem are well understood—when work flows from communication through deployment in a reasonably linear fashion.



Traditional Software Process Models

Waterfall - Problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.
- The waterfall model is mostly used for large system engineering projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Waterfall - Advantages

1. Simple
2. Easy to understand even for non technical customers
3. Oldest, widely used
4. Base for all other models by including feed back loops, iterations etc.

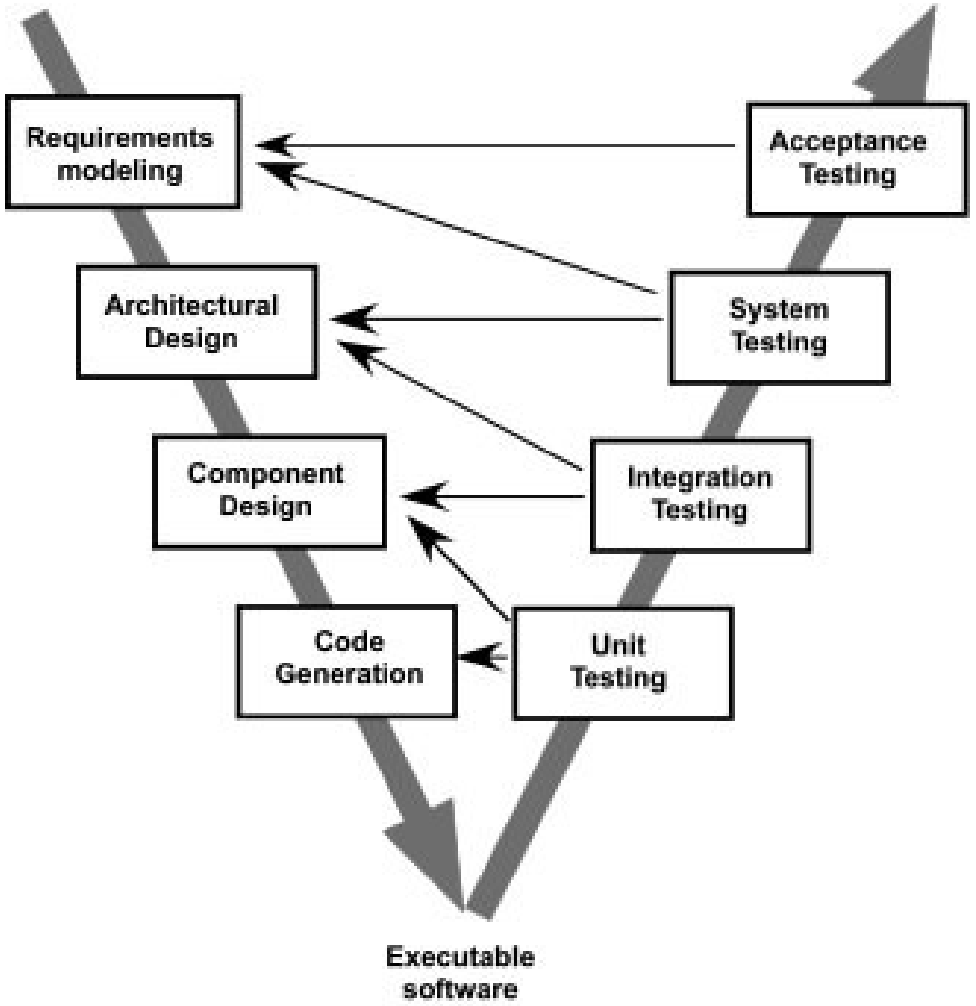
Traditional Software Process Models

Waterfall - Disadvantages

1. Real projects rarely follow this linear sequence.
2. Difficult for customer to state all requirements at one shot
3. Customer must have patience.

V-Model

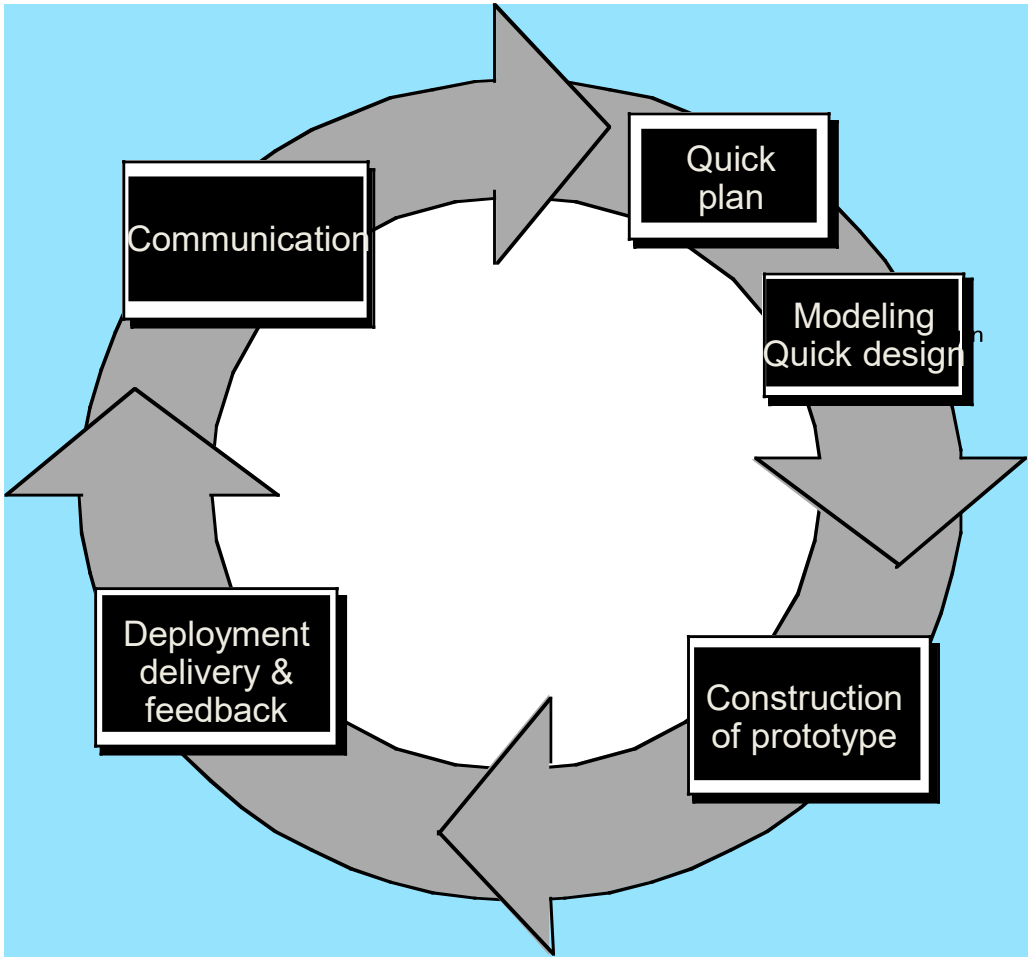
- A variation of waterfall model depicts the relationship of quality assurance actions to the actions associated with communication, modeling and early code construction activities.
- Team first moves down the left side of the V to refine the problem requirements.
- Once code is generated, the team moves up the right side of the V, performing a series of tests that validate each of the models created as the team moved down the left side.



Traditional Software Process Models

Prototyping

- Begins with communication
- A quick plan for prototyping and modeling occur.
- Quick design focuses on the representation of those aspects the software will be visible to end users. (Interface and output).
- Design leads to the construction of a prototype which will be deployed and evaluated.
- Stakeholder's comments will be used to refine requirements.



Traditional Software Process Models

Prototyping – When to Select

- Customer defines a set of general objectives
- Does not identify detailed requirements
- Developer may be unsure of the efficiency of an algorithm, the form that human computer interaction should take.
- When your customer has a legitimate need, but is clueless about the details, develop a prototype as a first step.

Prototyping - Advantages

1. Provides working model.
2. Customer is highly satisfied with such a modeling at initial stages
3. Developer gains business insight, reducing ambiguity
4. Great involvement of users
5. Reduce risks

Traditional Software Process Models

Prototyping - Disadvantages

1. Customer - not aware that only interface or appearance is concentrated much and long term quality is at stake.
2. False expectations from customer that end software modelling is finished or will have the same behavior/pace of the prototype.
3. Inappropriate choices of technology
4. Various iterations to a prototype that is to be discarded is expensive

Traditional Software Process Models

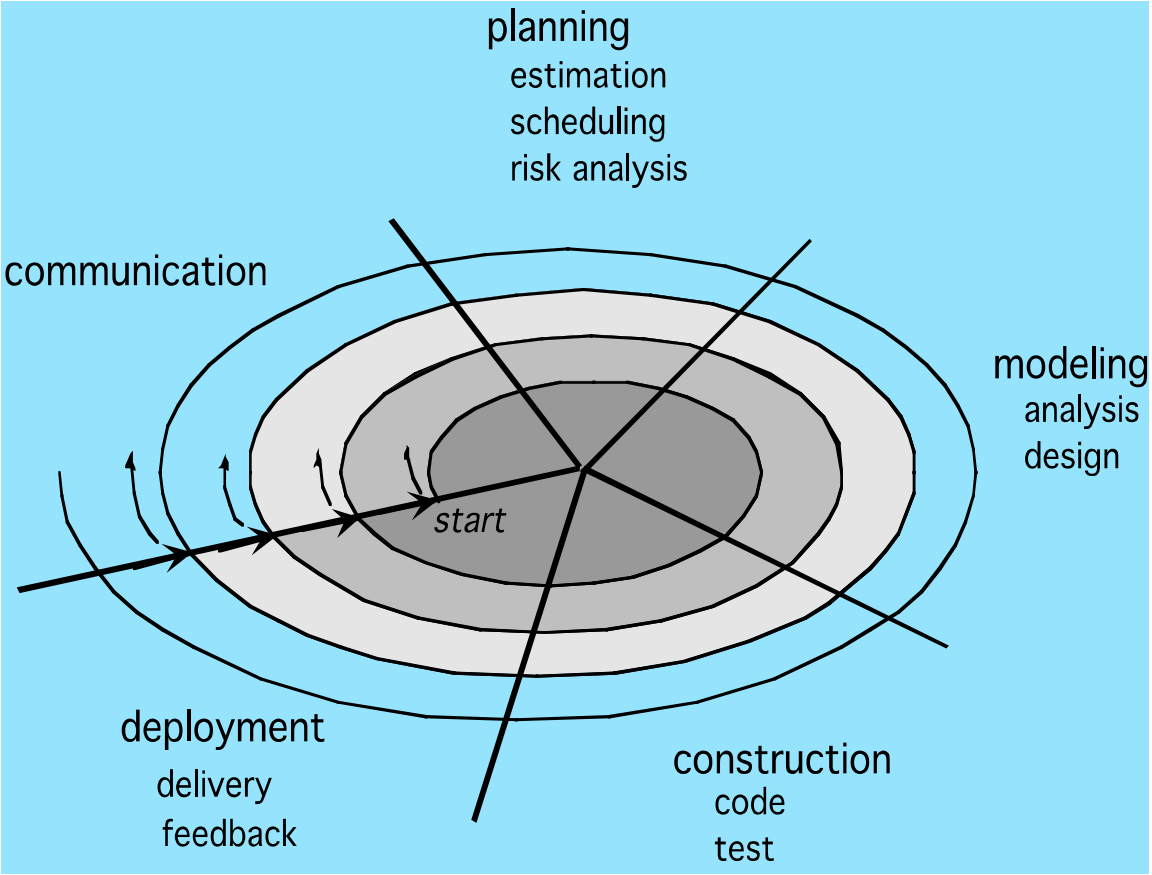
Spiral

- It couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model and is a risk-driven process model generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems.
- Two main distinguishing features: one is cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk. The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.
- A series of evolutionary releases are delivered.
- During the early iterations, the release might be a model or prototype.
- During later iterations, increasingly more complete version of the engineered system are produced.
- The first circuit in the clockwise direction might result in the product specification; subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software.
- Each pass results in adjustments to the project plan.
- Cost and schedule are adjusted based on feedback.

Traditional Software Process Models

Spiral

- Also, the number of iterations will be adjusted by project manager.
- Good to develop large-scale system as software evolves as the process progresses and risk should be understood and properly reacted to.



- Prototyping is used to reduce risk.
- However, it may be difficult to convince customers that it is controllable as it demands considerable risk assessment expertise.

Traditional Software Process Models

Spiral - Advantages

1. Applies throughout lifecycle
    - Concept Development
    - New Product Development
    - Product Enhancement
2. Risk is considered at each pass
3. Uses prototyping as risk reduction mechanism
4. Customer and developer understand and better react to risks

Spiral - Disadvantages

1. Difficult to convince customers that it is controllable
2. Demands considerable risk assessment expertise
3. Major risk is not uncovered/managed, problem will occur

Traditional Software Process Models

Rapid Application Development

- An approach to software development aimed at rapid delivery of the software.
- It often involves the use of database programming and development support tools such as screen and report generators.
- There are three broad phases to RAD:
  - Requirements planning
  - RAD design workshop
  - Implementation
- Requirements planning

  - Users and analysts meet to identify objectives of the application or system

  - Oriented toward solving business problems
- RAD Design Workshop
  - Design and refine phase
  - Use group decision support systems to help users agree on designs
  - Programmers and analysts can build and show visual representations of the designs and workflow to users

Traditional Software Process Models

Rapid Application Development

- RAD Design Workshop
  - Users respond to actual working prototypes
  - Analysts refine designed modules based on user responses
- Implementation Phase

  - As the systems are built and refined, the new systems or partial systems are tested and introduced to the organization

  - When creating new systems, there is no need to run old systems in parallel

- The Martin approach to RAD includes four phases:
  - Requirements planning
  - User design
  - Construction
  - Cutover