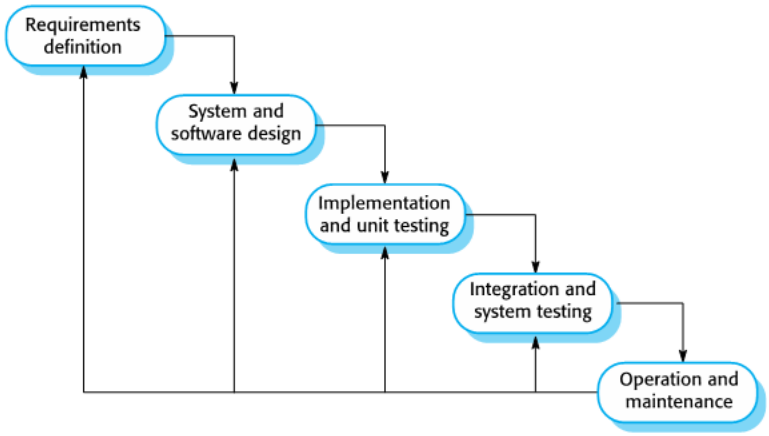
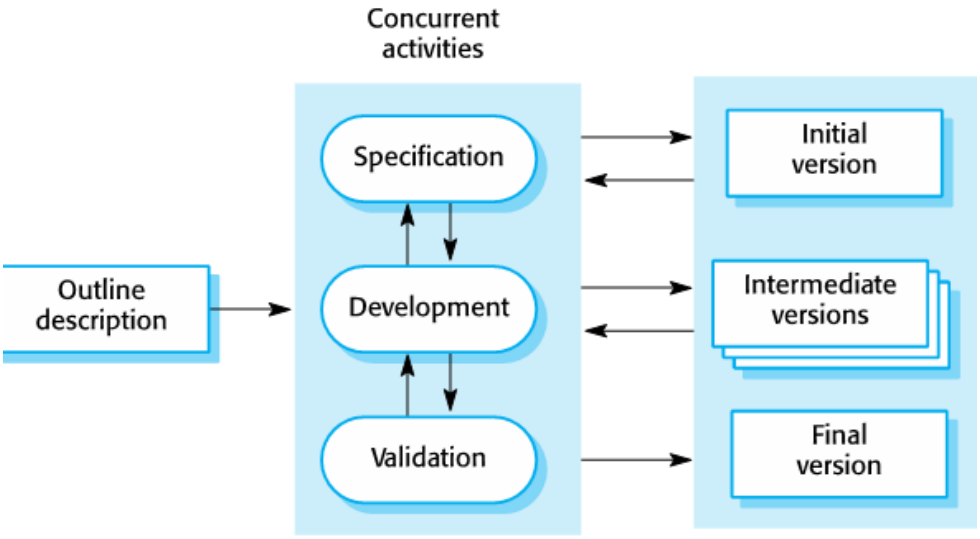
Chapter 2: Software Processes:

- Software Process: structured activities to specify, design, develop and test a software
- Software Process Model: abstract representation of a software process. describes sequence of activities in an SE project and its relative order.
- Process description includes:
    - activities such as UI design, ordering, making data models
    - Product of the activities
    - Roles of people in that process
    - Pre – post conditions
- Plan driven vs agile:
    - Plan driven: all process activities are planned in advance, with which the progress is measured
    - Agile: planning in increments, hence good for changing requirements
- Practically a mixture of both is used
- Process models
    - Waterfall model
    - Incremental model
    - Integration and Configuration
- Large systems use a mixture of all these
- Waterfall:



    - Definition: The waterfall model is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one
    - Plan-driven
    - Specification and development are in separate phases

    - Problems:
        - does not welcome change once started because a phase has to be completed before proceeding
        - unresponsive to changing customer needs as inflexible partitioning of phases
    - Good for projects with:
        - defined requirements and changes are limited,
        - large systems as it increases coordination in work
    - Disadvantages:
        - requirements must be stable
        - expensive to change decisions
        - less customer involvement (only in start and end phase)
- Incremental:
    - Definition: The incremental build model is a method of software development where the product is designed, implemented and tested incrementally until the product is finished.

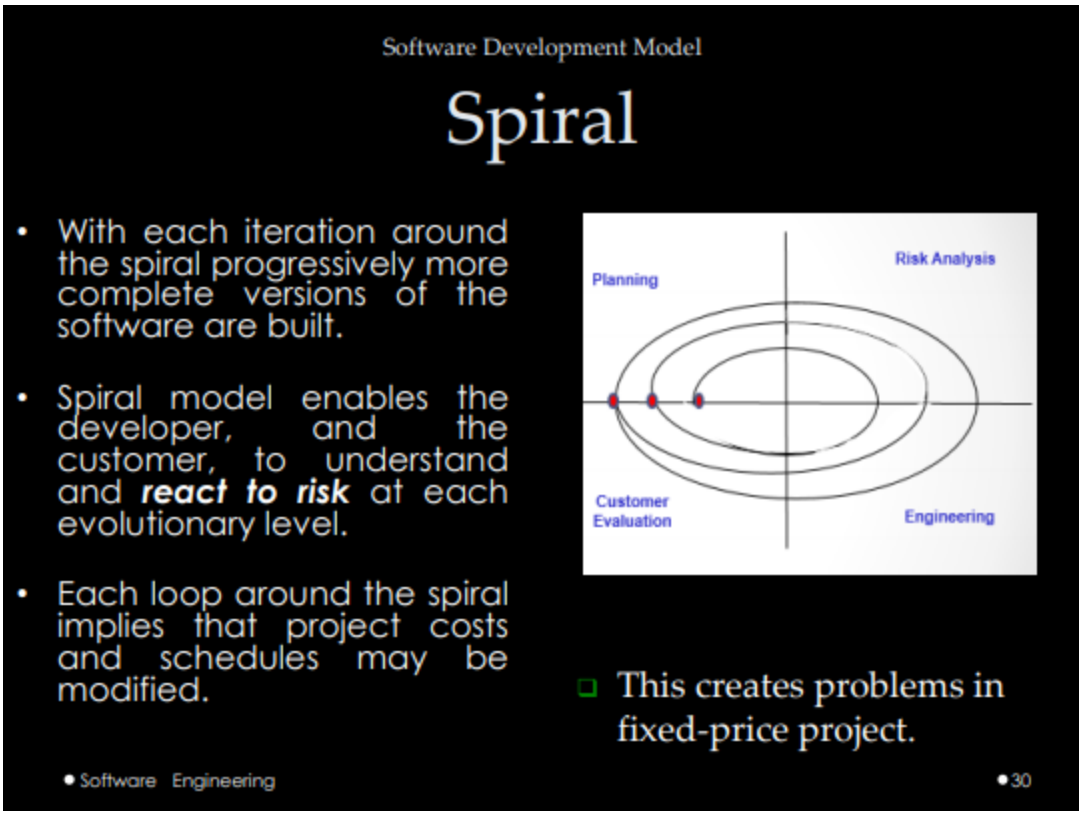


    - Specification, development, validation are interleaved
    - Maybe agile or plan driven
    - Benefits:
        - expense of changing decision reduced
        - easier to get customer feedback
        - rapid delivery and deployment
    - Problems:
        - need of regular refactoring as increments area added
        - Process invisible due to fast paced development

Evolutionary Models: Spiral

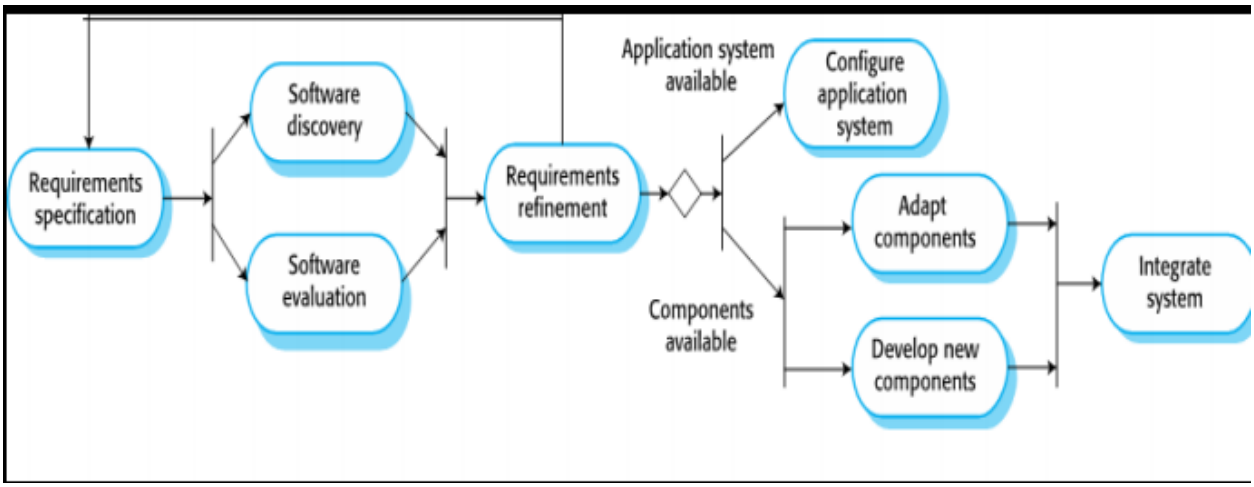
- Spiral is primary **Risk Driven** approach. Spiral model consist of different **cycle. In each cycle we try to address some risk elements.**

- Planning: Determines objectives, alternatives and constraints.
- Risk Analysis: Analysis of alternatives as well as an identification and/or resolution of risks.
- Engineering: Development of the next level of product
- Customer evaluation: Assessment of the results of engineering



- Integration and Configuration:

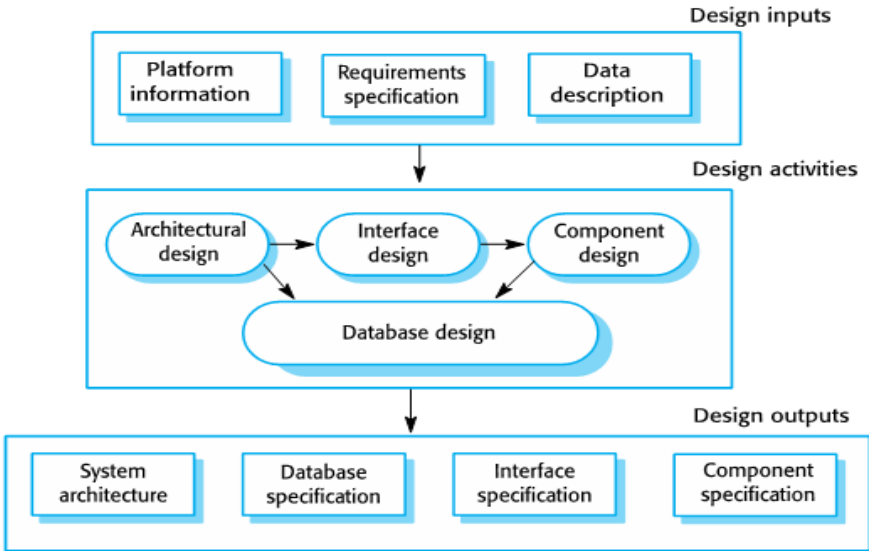
  The **integration and configuration** process model attempts to **integrate** existing, reusable components into a system or software, rather than developing them from scratch
  - Reusing software using existing components, known as Commercial Off the Shelf systems
  - Reusable elements can easily be configured to meet user needs hence used nowadays.
  - Types:
    - Stand – alone applications
    - Packages for component frameworks as .Net
    - Web services
- Stages:
  - Requirements Specification – analyze user requirements
  - Software discovery and evaluation – looking for reusable components
  - Requirements Refinement – validating models with user requirements
  - Application system configuration
  - Component adaptation and integration



- Advantages:
  - cost and risk reduced due reusing components
  - rapid delivery and deployment
- Disadvantages:
  - may not meet user requirements
  - inability to evolve reusable components

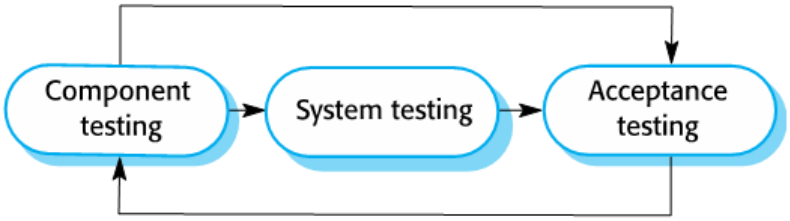
- Process Activities:
  - These do not include technical, collaboration or managing activities.
  - These include:
    - Specification – what system does
    - Design and Implementation
    - Validation – testing it
    - Evolution – changing to customer needs
  - Software Specification:
    - what services are required and system constraints (detailed form of user specification)
    - Involves requirements engineering:
      - Requirements Elicitation:
        - stakeholders requirement and expectation from the system
      - Requirements Specification:
        - Defining the requirements in detail
      - Requirements validation:
        - validity of these requirements
  - Design and implementation

- converting specification to a working system
- Design: structure that realizes the specification
- Implementation: translate this structure into working program
- Design Process:

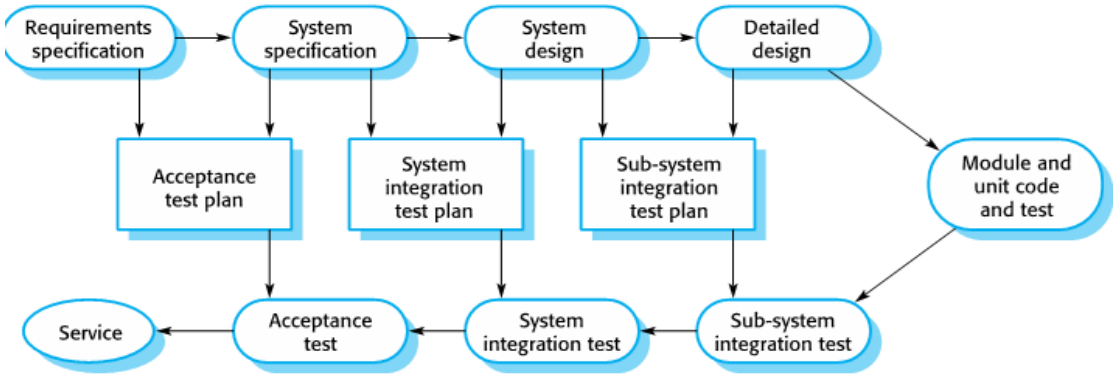


  - Architectural Design: understanding structure of systems including modules' relationship and distribution
  - Database Design: schema design
  - Interface Design: interfaces between components
  - Component Design: operation and reusability of components
- Design and implementation are mostly interleaved

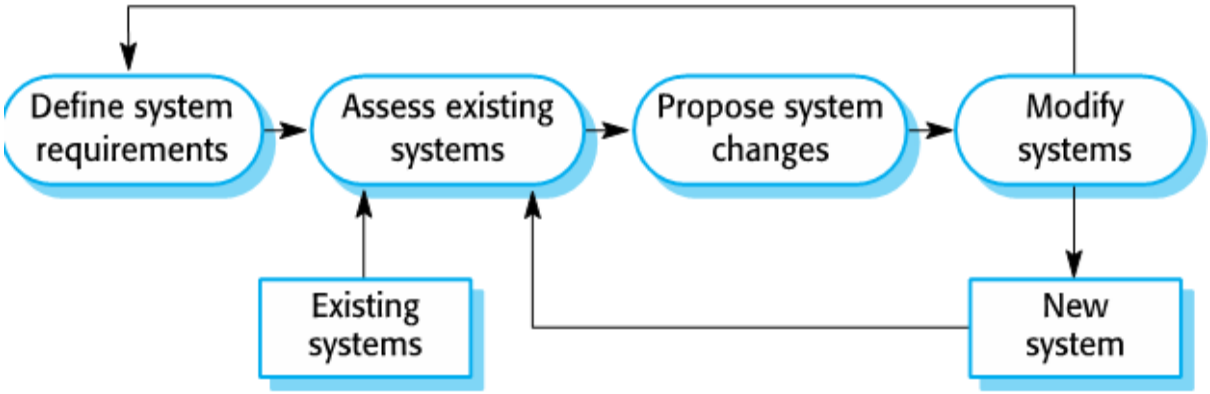
- Software validation:
  - (V & V) of the system
  - Verification: identification that system is error / bug free
  - Validation: verification + the system gives desired output
  - Done using test cases made from real data to be processed
  - Stages:



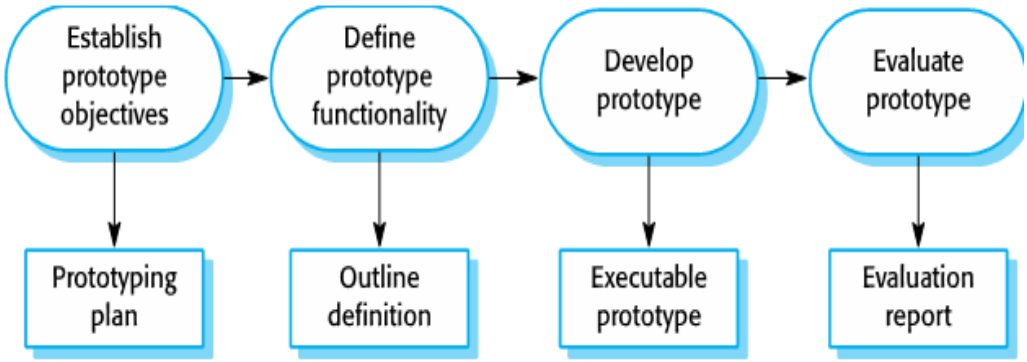
- Component:
  - made and tested independently
- System:
  - testing the recombined system as a whole
- Acceptance:
  - test with customer data for validation



- Evolution:
  - maintenance and changing system with changing requirements



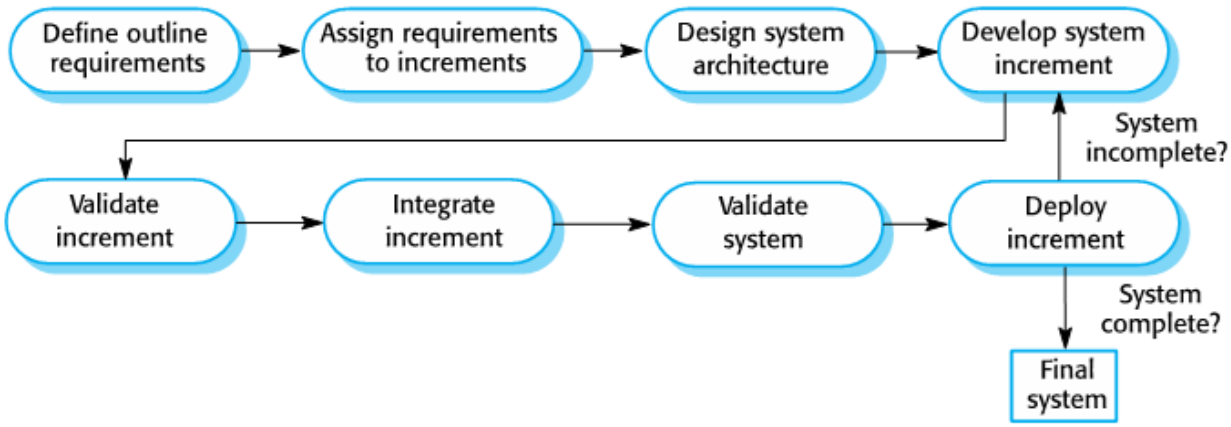
- Coping with change:
  - Change is obvious due to changing user requirements, new technologies and hence need rework (re-analysis) which implies some cost
- How to reduce this rework cost?
  - Change anticipation:
    - anticipating possible changes to avoid rework. Involves software prototyping
  - Change tolerance:
    - system designed in such a way so that adding changes incur a low cost. Involves incremental development
- Software Prototyping:
  - part of the system is developed quickly to check the customer's requirements and design decisions
  - These are for areas not well understood
  - used for testing, exploring options in UI design etc.
  - Benefits:
    - improves maintainability
    - reduces effort
    - increases quality
  - Prototype dev process:



    - Focus on functional rather than non-functional requirements
    - No error checking
    - Should be discarded as not valid base for production

- Incremental Development and Delivery:

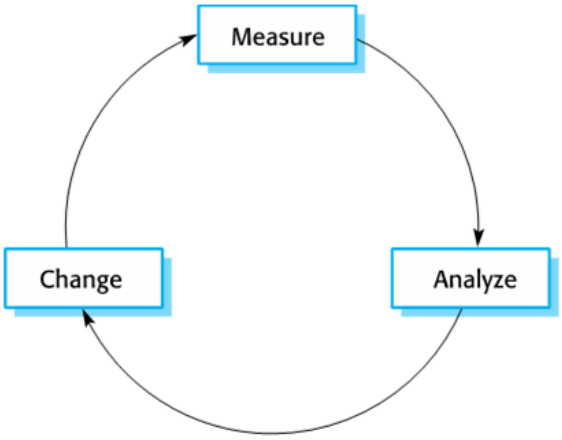
- Development:
  - developing in increments which is checked before proceeding to next
  - good for agile!
  - evaluation using customer feedback
- Delivery:
  - system is delivered in increments are delivered to the customer for feedback
  - high priority functionalities in user requirements are sent in early increments



- Advantages:
  - increased customer collaboration
  - less risk of overall failure
  - high priority requirements are tested first
- Problems:
  - hard to identify common functionalities in all increments
  - complete system specification is not provided

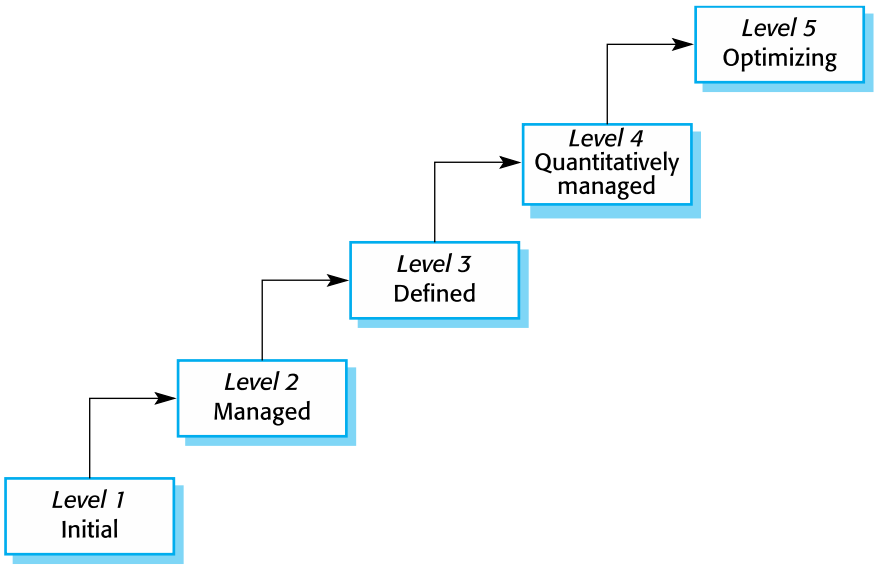
- Process Improvement:
  - understanding current processes and using these to increase quality, reduce costs and dev times etc.
  - Approaches:
    - Process maturity
      - focus on improving project management, project technicality
    - Agile

        - focus on iterative development and reduction of documentation overheads (rapid and responsive)
    - Improvement Cycle:



    - Measurement:
        - collecting data to measure effectiveness of the change
        - Some include:
            - time taken to complete
            - resources taken
            - errors occured
    - Analysis:
        - process weaknesses and bottlenecks are identified along with making a process model
    - Change:
        - changes are made to overcome weaknesses

SEI capability maturity model:



- Level 1:
    - Processes followed are adhoc and immature and are not well defined.
    - Unstable environment for software dvelopment.
- Level 2:
    - Focuses on establishing basic project management policies.
    - Experience with earlier projects is used for managing new similar natured projects.
- Level 3:
    - At this level, documentation of the standard guidelines and procedures takes place.
    - It is a well defined integrated set of project specific software engineering and management processes.
- Level 4:
    - At this stage, quantitative quality goals are set for the organization for software products as well as software processes.
- Level 5:
    - This is the highest level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback. Using new tools, techniques to prevent recurrent errors