COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

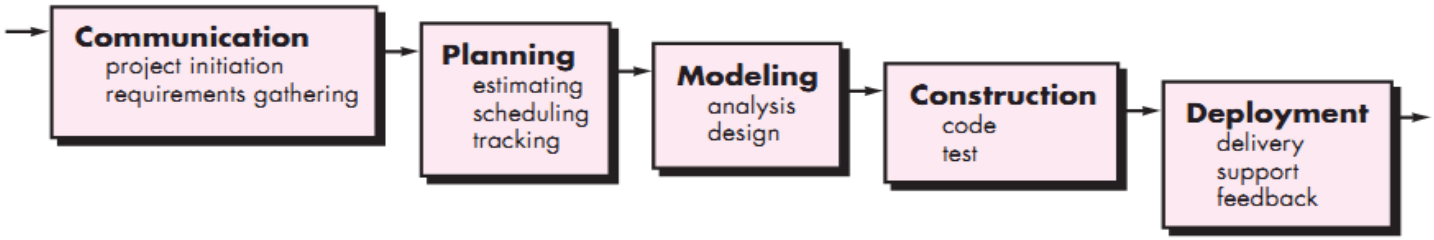
CHAPTER 2

SOFTWARE DEVELOPMENT PROCESS MODEL

SOFTWARE PROCESS

- A structured set of activities required to develop a software system
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective

WATERFALL MODEL



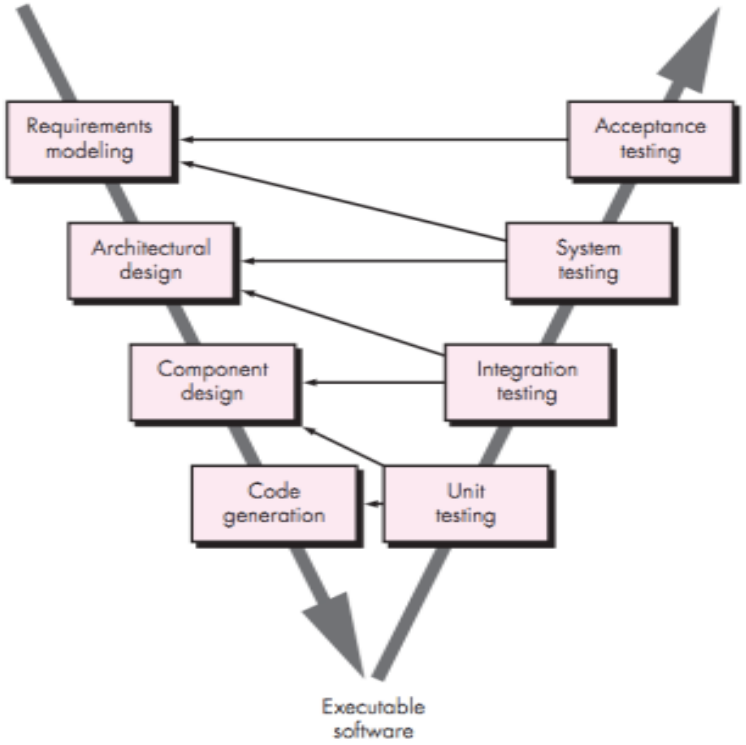
- **The waterfall or linear sequential model**

Problems of Waterfall Model

- Inflexible partitioning of the project into distinct stages where next phase starts only after completion of the previous phase
- This makes it difficult to respond to changing customer requirements (no backtracking)
- Therefore, this model is only appropriate when the requirements are well-understood

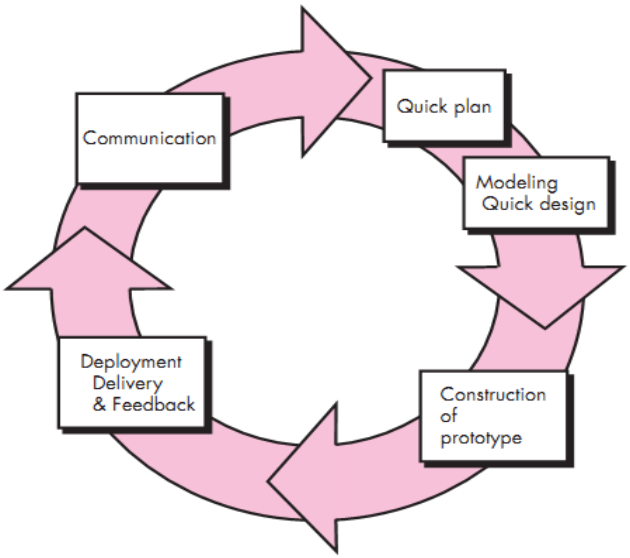
V - MODEL

- The V-model is a SDLC model where execution of processes happens in a sequential manner in V-shape. It is also known as Verification and Validation model.
- V-Model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle there is a directly associated testing phase.
- This is a highly disciplined model and next phase starts only after completion of the previous phase.



PROTOTYPING MODEL

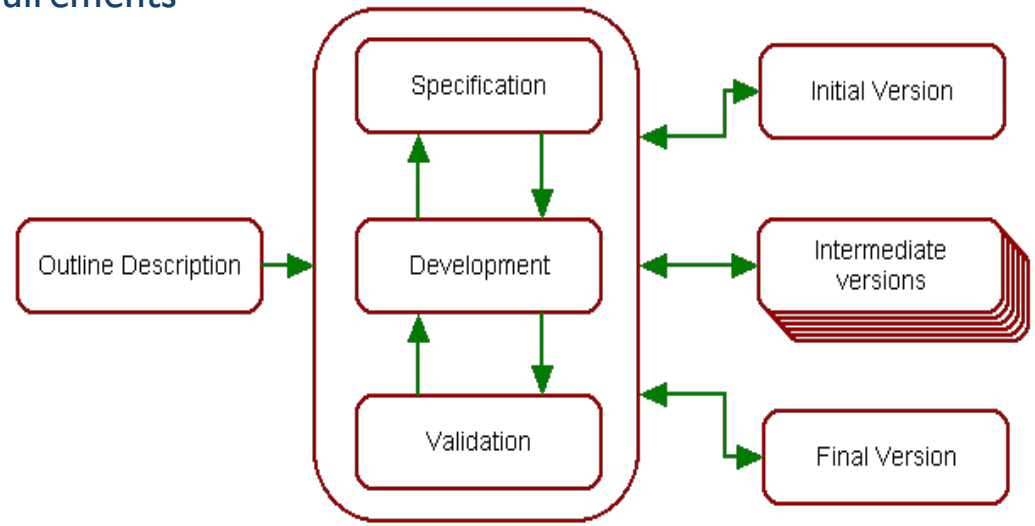
- Requirements are not clear and prototype serves as a mechanism for identifying software requirements
- Iteration occurs as the prototype is tuned to satisfy the needs of the customer



- System requirements ALWAYS evolve in the course of a project, so process iteration is useful where earlier stages are reworked is always part of the process for large systems

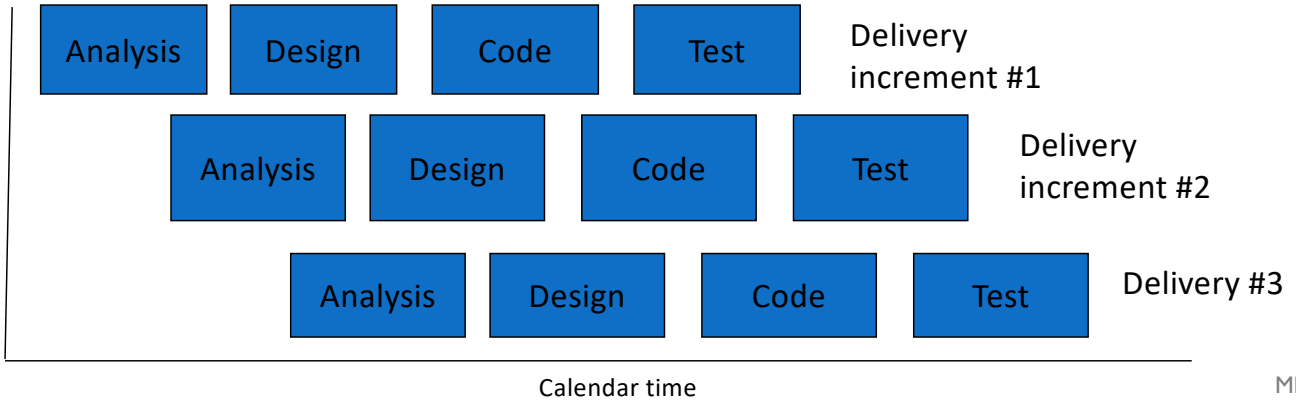
EVOLUTIONARY DEVELOPMENT

- **Exploratory development:** Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements
- **Throw-away prototyping:** Objective is to understand the system requirements. Should start with poorly understood requirements



INCREMENTAL DEVELOPMENT

- 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 (SPIRAL).
  The requirements are relatively certain but there are many complexities that leads to frequent changes.
- User requirements are prioritised 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 DEVELOPMENT

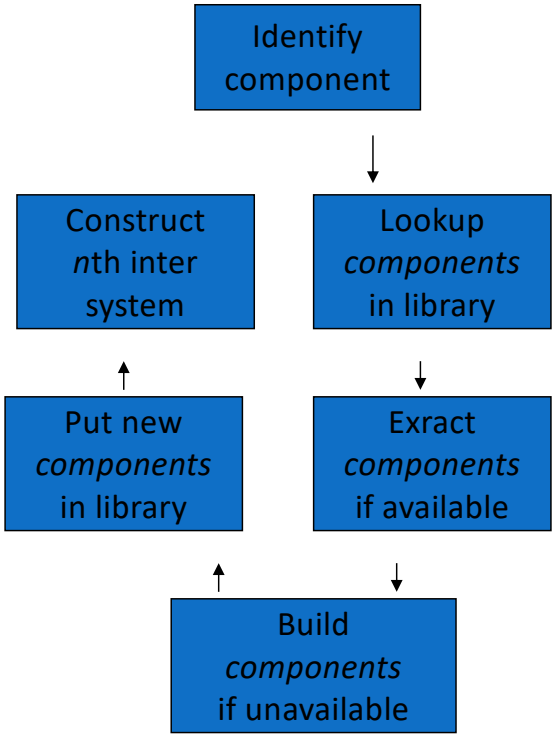
- **Advantages of Incremental Development**
  - Customer value can be delivered with each increment so system functionality is available earlier
  - Deliver the core product first
  - Early increments act as a prototype to help elicit requirements for later increments
  - Lower risk of overall project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

RAPID APPLICATION DEVELOPMENT (RAD)

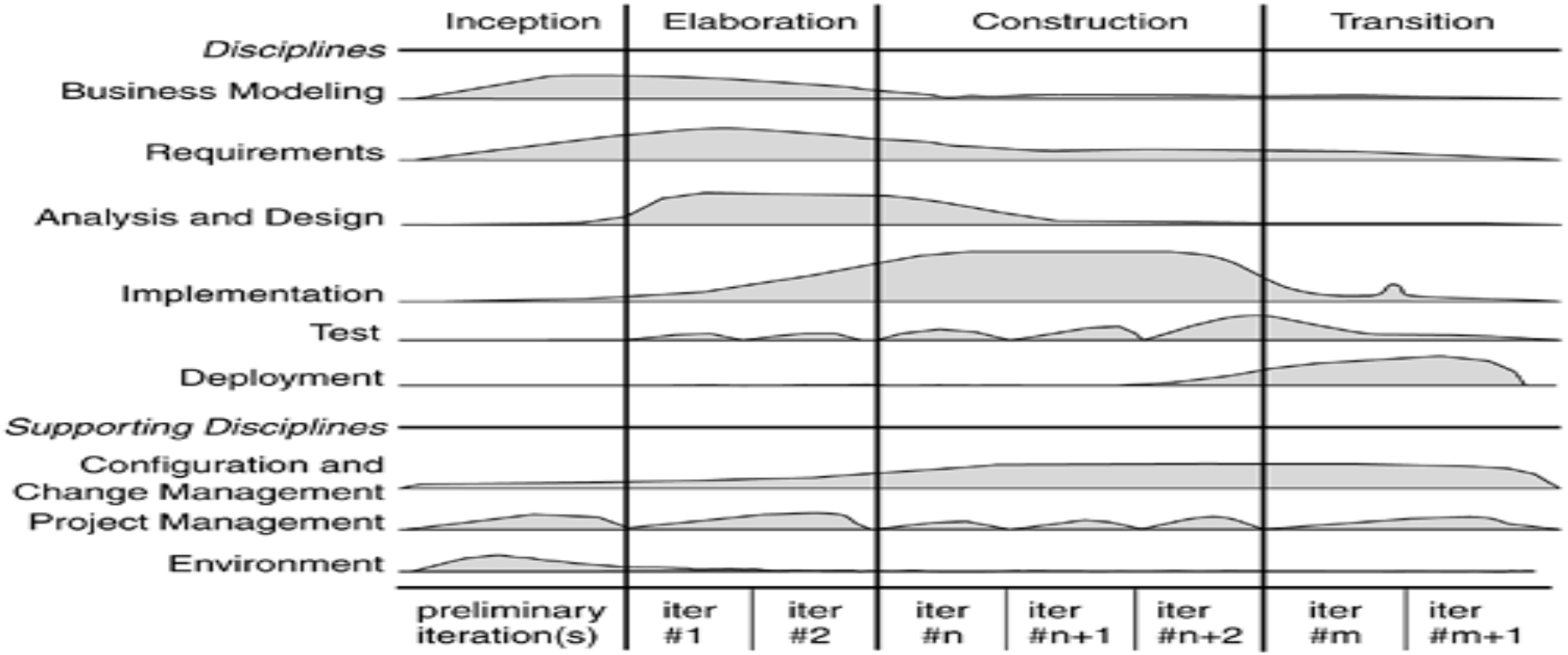
- It is a type of incremental model. The developments are time boxed, delivered and then assembled into a working prototype
- In RAD model the components or functions are developed in parallel as if they were mini projects (frozen requirements in each increments)
- This can quickly give the customer something to see and use and to provide feedback
- Delivers a fully functional system in 90 days, give or take 30 days
- Phases of RAD are:
  - Requirements Planning
  - User Design (user interact with the system analysts)
  - Construction (program and application development)
  - Cutover (testing, changeover to new system, user training)

COMPONENT BASED DEVELOPMENT MODEL

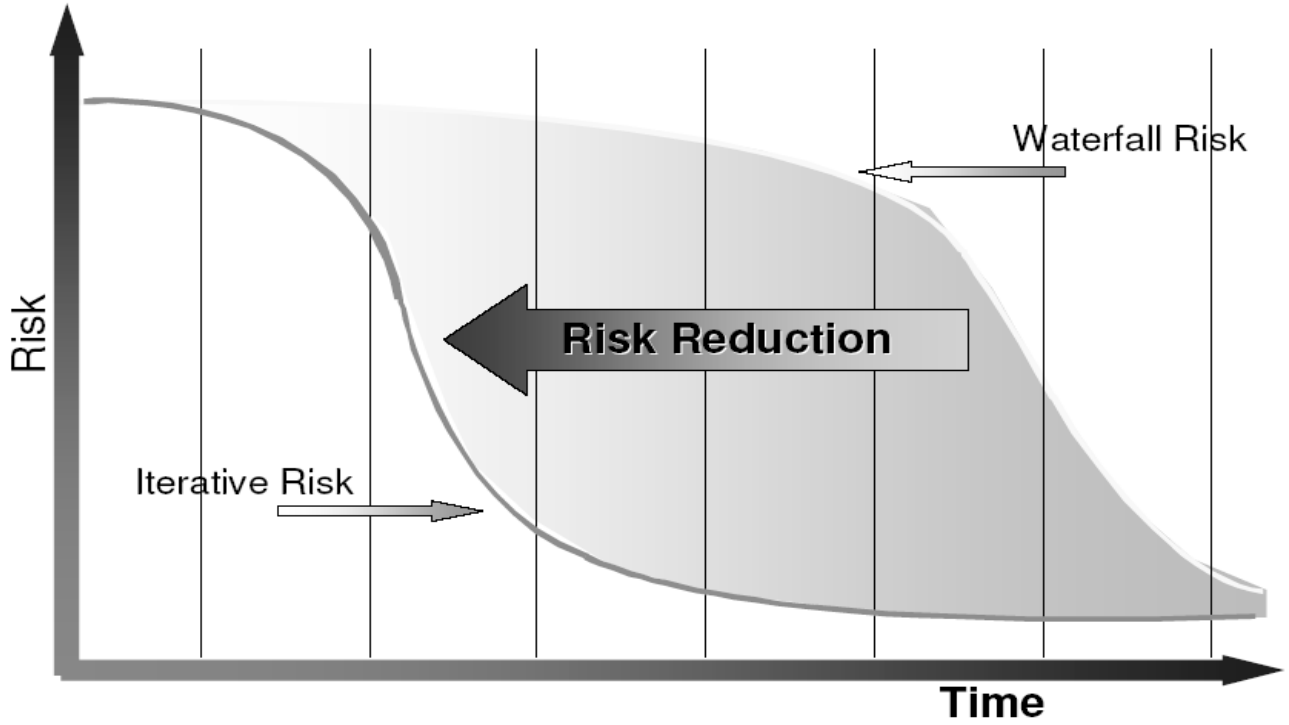
- Suitable for re-usable object oriented classes
- Apply characteristics of spiral development



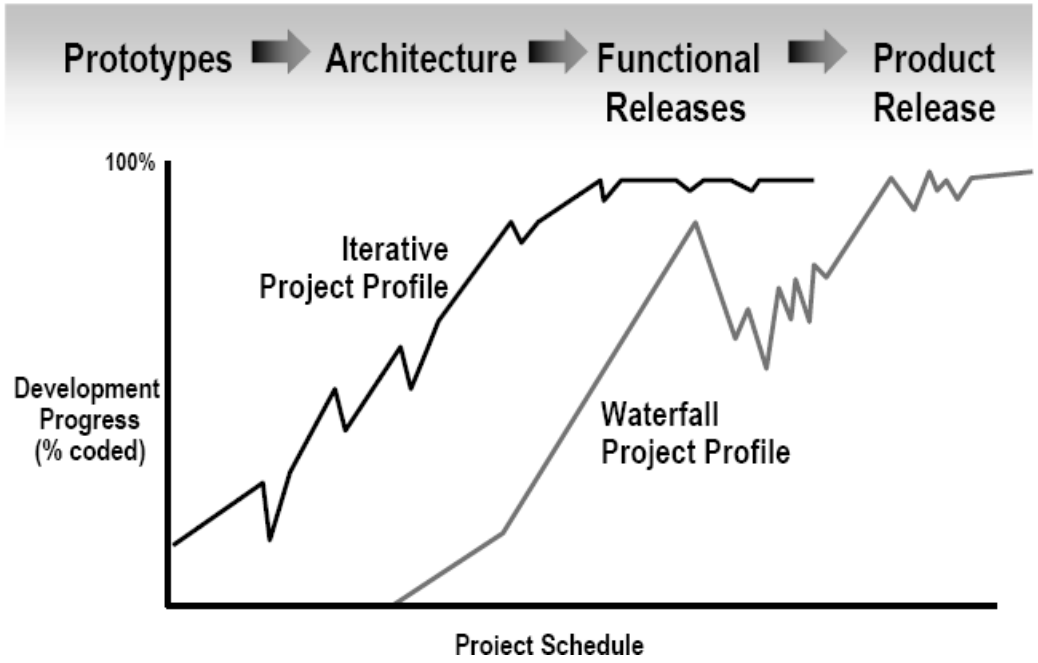
RATIONAL UNIFIED PROCESS (RUP)



RISK PROFILE



REDUCE SCRAP/REWORK: USE AN ITERATIVE PROCESS



- Iterative Development
  - Products are visible at an early stages of development
  - Low probability of rework in case of defects in the deliverable product

REFERENCES

- R.S. Pressman & Associates, Inc. (2010). *Software Engineering: A Practitioner's Approach.*
- Kelly, J. C., Sherif, J. S., & Hops, J. (1992). An analysis of defect densities found during software inspections. *Journal of Systems and Software*, *17*(2), 111-117.
- Bhandari, I., Halliday, M. J., Chaar, J., Chillarege, R., Jones, K., Atkinson, J. S., & Yonezawa, M. (1994). In-process improvement through defect data interpretation. *IBM Systems Journal*, *33*(1), 182-214.