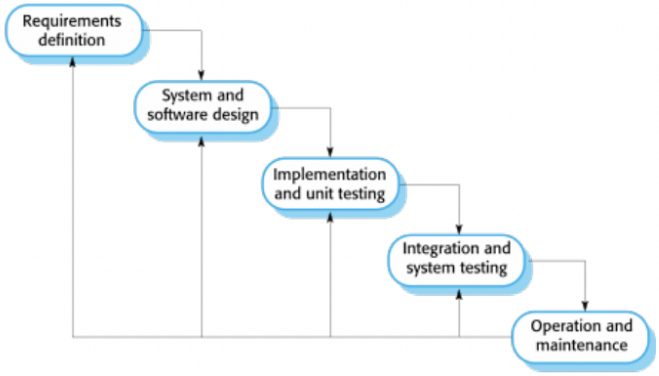
CAS CS 411
Lec 2

SDLC

1. The software process
    a. A structured set of activities required to develop a software system.
    b. Many different software processes but all involve
        i. Specification – defining what the system should do
        ii. Design and implementation – defining the organization of the system and implementing the system
        iii. Validation – checking that it does what the customer wants
        iv. Evolution – changing the system in response to changing customer needs.
    c. A software process model is an abstract representation of a process. It presents a description of a process from some
2. Software process descriptions
    a. When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
    b. Process descriptions may also include
        i. Products, which are the outcomes of a process activity
        ii. Roles, which reflect the responsibilities of the people involved in the process
        iii. Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.
3. Plan-driven vs agile processes
    a. Plan-driven, or prescriptive, processes are processes where all of the process activities are planned in advance and progress is measured against this plan — they strive for an orderly approach to development.
    b. In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
    c. In practice, most practical processes include elements of both plan-driven and agile approaches.
    d. There are no right or wrong software processes
4. Software process models
    a. The waterfall model
        i. Plan-driven model. Separate and distinct phases of specification and development.

b. Incremental development
    i. Specification, development and validation are interleaved. May be plan-driven or agile.
c. Integration and configuration
    i. The system is assembled from existing configurable components. May be plan-driven or agile.
d. In practice, most large systems are developed using a process that incorporates elements from all of these models.

5. Waterfall model
    a. Activity-centered view of the software lifecycle
        i. Define detailed upfront requirements
        ii. Come up with the design that will support the required behavior.
        iii. Implement the required system.
        iv. Integrate and test the components.
    b. Activities are performed in sequence
    c. Described in 1970 by Royce.



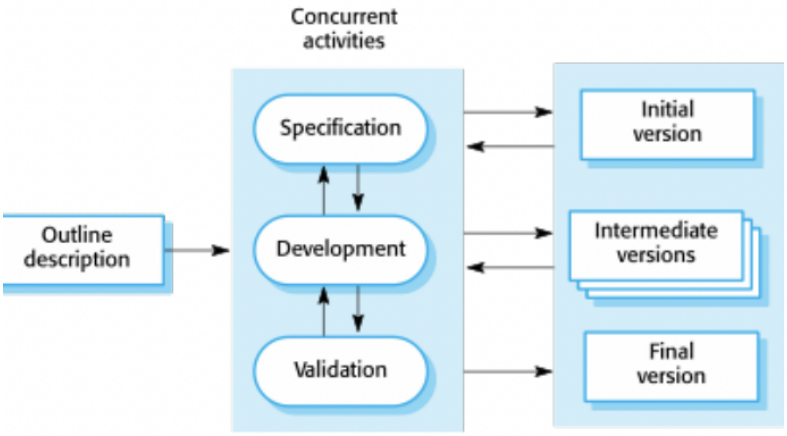
6. Waterfall model phases
    a. There are separate identified phases in the waterfall model
        i. Requirements analysis and definition
        ii. System and software design
        iii. Implementation and unit testing
        iv. Integration and system testing
        v. Operation and maintenance
    b. The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.
7. Problems with the Waterfall
    a. Complete up-front specifications with sign-off
        i. Research showed that 45% of features created from early specifications were never used—with an additional 19% rarely used [Johnson02].

ii. Over-engineering, a study of 400 projects spanning 15 years showed that less than 50% of the code was actually useful or used [CLW01].

b. Late Integration and Test
   i. The waterfall pushes this high-risk and difficult issues toward the end of the project. Waterfall is called fail-late lifecycle.

c. Reliable Up-front Estimates and Schedules
   i. Can not be done when the full requirements and risks are not reliably known at the start, and high rates of change are the norm.

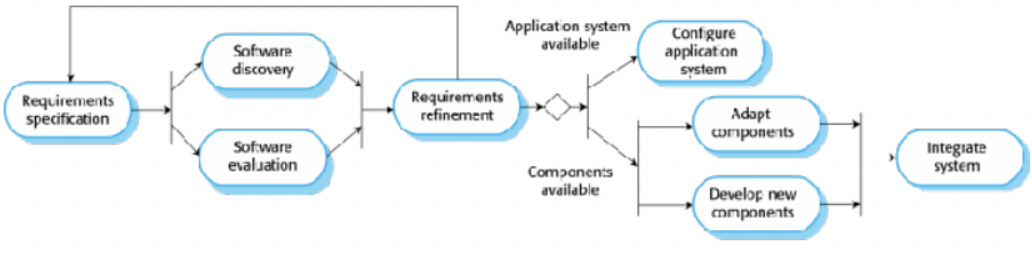
d. "Plan the work, work the plan" values
   i. Limited value for high change, novel, innovative domains such as software development.

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

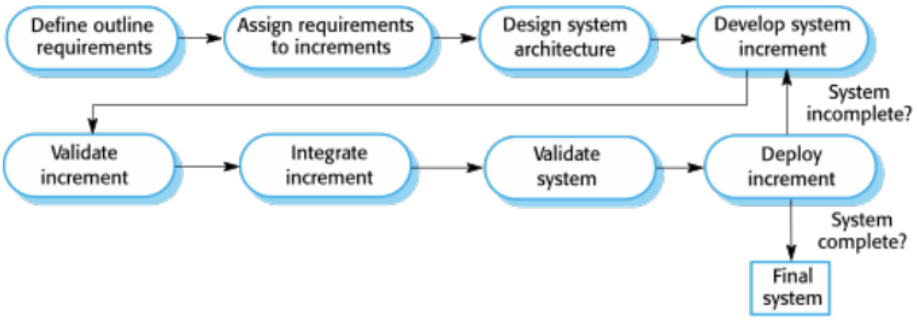
10. Incremental development benefits
    a. The cost of accommodating changing customer requirements is reduced.
       i. The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.

b. It is easier to get customer feedback on the development work that has been done.
    i. Customers can comment on demonstrations of the software and see how much has been implemented.
c. More rapid delivery and deployment of useful software to the customer is possible.
    i. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

11. Incremental development problems
    a. The process is not visible.
        i. Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
    b. System structure tends to degrade as new increments are added.
        i. Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.
12. Integration and configuration
    a. Based on software reuse where systems are integrated from existing components or application systems (sometimes called COTS -Commercial-off-the-shelf) systems).
    b. Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
    c. Reuse is now the standard approach for building many types of business system
13. Types of reusable software
    a. Stand-alone application systems (sometimes called COTS) that are configured for use in a particular environment.
    b. Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
    c. Web services that are developed according to service standards and which are available for remote invocation.
14. Reuse-oriented software engineering
    a.



15. Advantages and disadvantages
    a. Reduced costs and risks as less software is developed from scratch
    b. Faster delivery and deployment of system
    c. But requirements compromises are inevitable so system may not meet real needs of users
    d. Loss of control over evolution of reused system elements
16. Incremental delivery
    a. 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.
    b. User requirements are prioritised and the highest priority requirements are included in early increments.
    c. Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.
17. Incremental development and delivery
    a. Incremental development
        i. Develop the system in increments and evaluate each increment before proceeding to the development of the next increment
        ii. Normal approach used in agile methods
        iii. Evaluation done by user/customer proxy.
    b. Incremental delivery
        i. Deploy an increment for use by end-users
        ii. More realistic evaluation about practical use of software
        iii. Difficult to implement for replacement systems as increments have less functionality than the system being replaced.
18. Incremental delivery

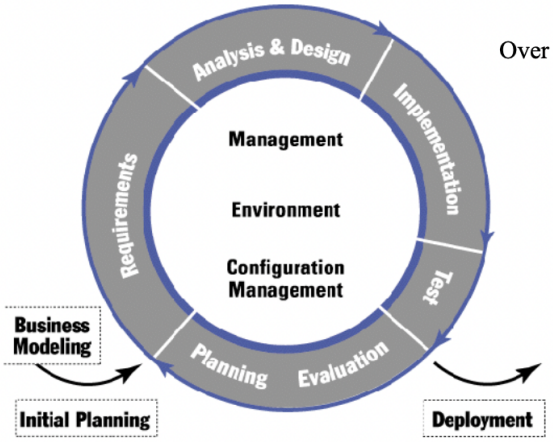


    a.
19. Incremental delivery advantages
    a. Customer value can be delivered with each increment so system functionality is available earlier.
    b. Early increments act as a prototype to help elicit requirements for later increments.

c. Lower risk of overall project failure.
d. The highest priority system services tend to receive the most testing

20. Incremental delivery problems
    a. Most systems require a set of basic facilities that are used by different parts of the system.
        i. As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
    b. The essence of iterative processes is that the specification is developed in conjunction with the software.
        i. However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract
21. RUP: Rational Unified Process
    a. Derived from the work on the UML at Rational
        i. Booch, Jacobson, and Rumbaugh (1999)
    b. 4 Phases
        i. Inception
        ii. Elaboration
        iii. Construction
        iv. Transition
    c. Several Iterations in each phase
    d. For each iteration, several parallel activities (workflows)
22. RUP Phases
    a. Inception
        i. Establish the business case for the system
        ii. Identify actors and initial use cases
        iii. Initial planning, estimation and schedule
    b. Elaboration
        i. Understand problem domain
        ii. Requirements model (use case model)
        iii. Overall system architecture
    c. Construction
        i. System design, object design, programming and testing
        ii. Develop a working software system ready to deliver to users
    d. Transition
        i. Deploy the system in its operating environment.

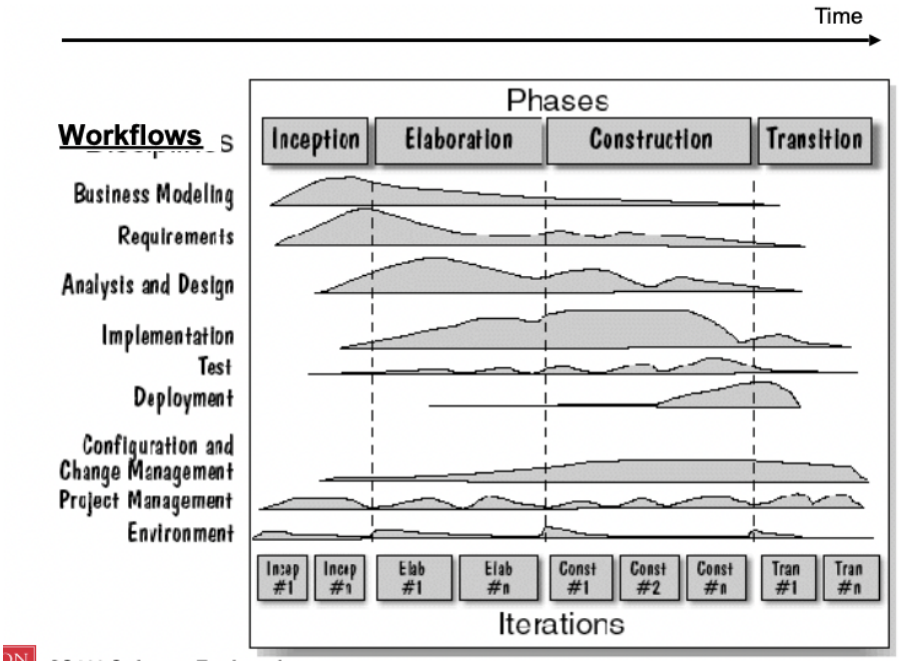
e.



Over time…

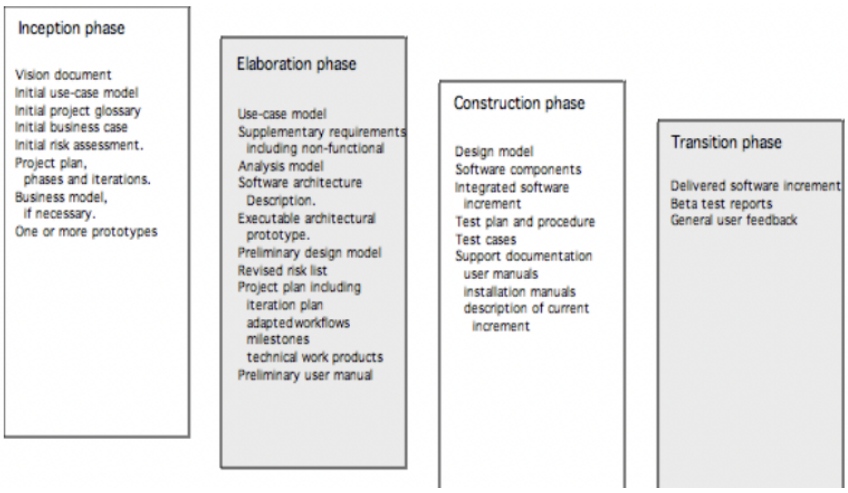
23. RUP illustrated

a.



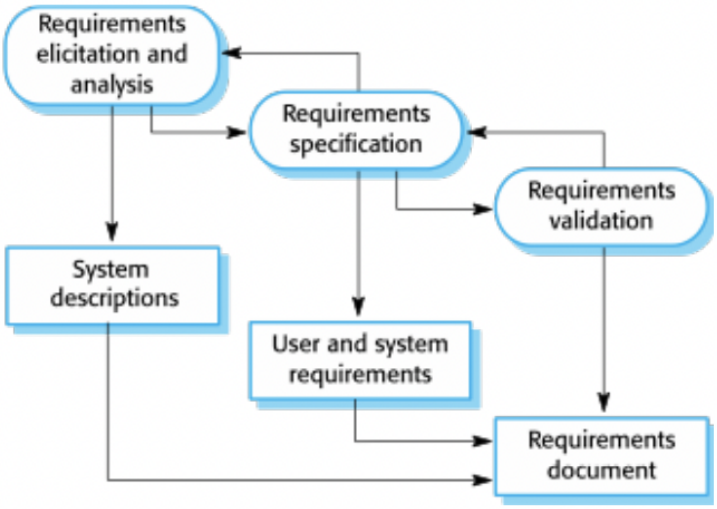
24. RUP Work Products

a.

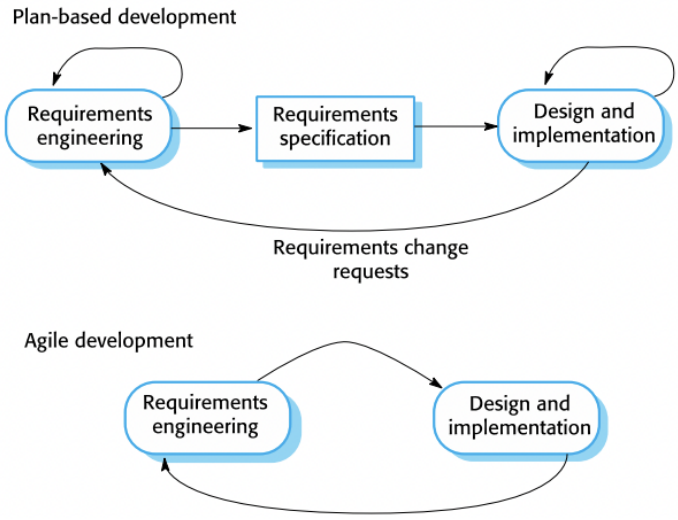


25. RUP Good Practice
    a. Develop software iteratively
        i. Plan increments based on customer priorities
    b. Manage requirements
        i. Explicitly document requirements and track requirement changes - Analyze impact of changes before accepting them
    c. Use component-based architectures
    d. Visually model software (UML)
    e. Control changes to software
        i. Manage changes to software using a change management system and configuration management procedures and tools

26. The requirements engineering process
    a.



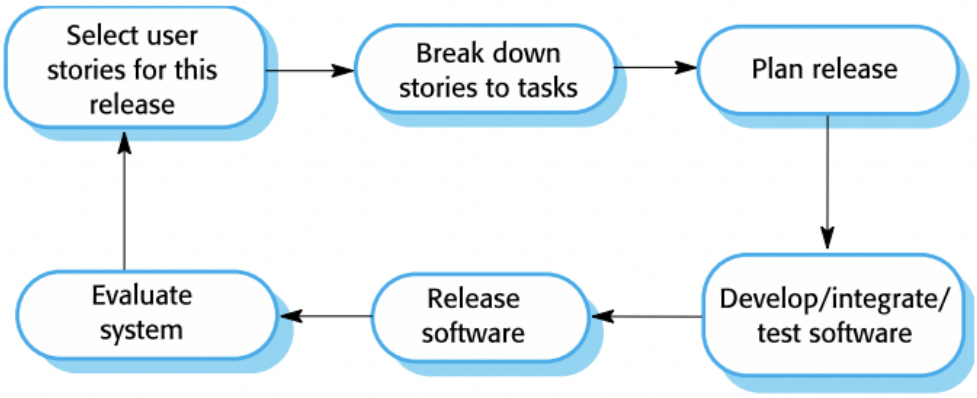
27. Plan-driven vs agile development
    a.



28. Agile methods
    a. Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
        i. Focus on the code rather than the design
        ii. Are based on an iterative approach to software development
        iii. Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
    b. The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.
29. The principles of agile methods
    a.

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

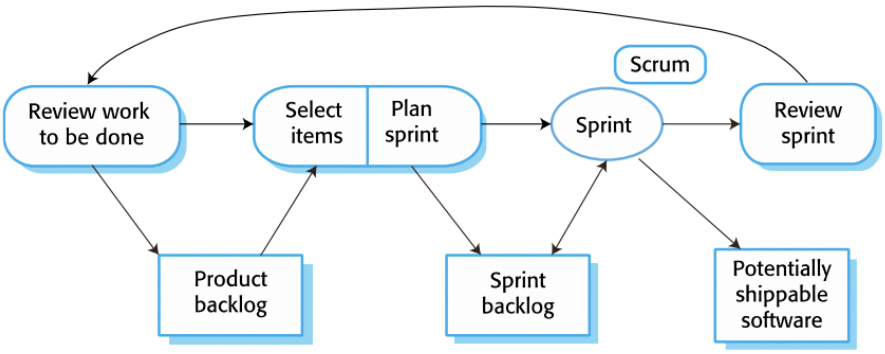
30. The extreme programming (XP) release cycle
    a.



31. Scrum
    a. Scrum is an agile method that focuses on managing iterative development rather than specific agile practices.

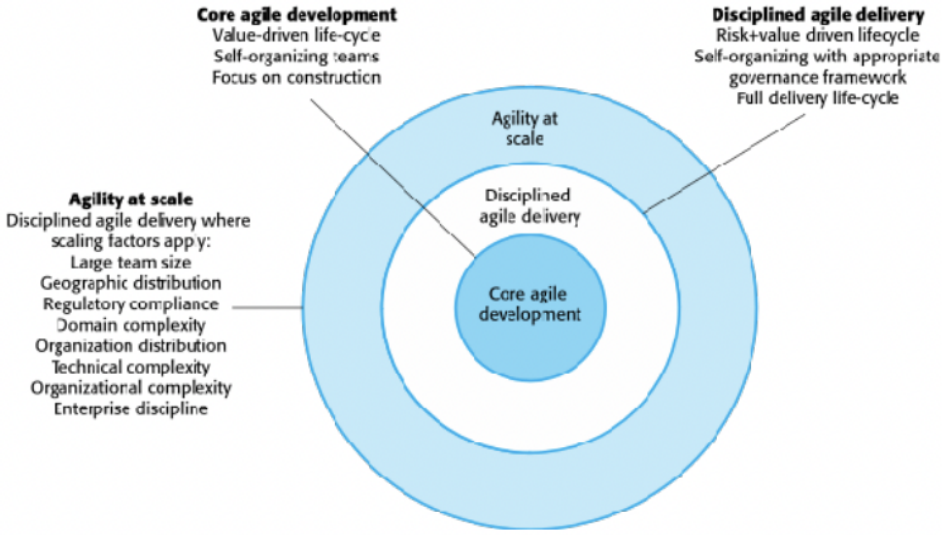
b. There are three phases in Scrum.
    i. The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
    ii. This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
    iii. The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

32. Scrum sprint cycle



a.

33. IBM's agility at scale model



a.

34. Bottom line
    a. Software development is a planned activity
    b. We use process to both manage the software development lifecycle (SDLC) activities and to improve the process itself
    c. Each company uses its own SDLC model but all of them are essentially Define, Design, Develop, Deliver
    d. The differences are often of scale and/or concurrency of activities