Unit 2

Software Processes

Objectives

- To introduce software process models
- To describe a number of different process models and when they may be used
- To describe outline process models for requirements engineering, software development, testing and evolution

Key points

- Software processes are the activities involved in producing and evolving a software system. They are represented in a software process model
- General activities are specification, design and implementation, validation and evolution
- Generic process models describe the organisation of software processes
- Iterative process models describe the software process as a cycle of activities

Software Processes

- Coherent sets of activities for specifying, designing, implementing and testing software systems

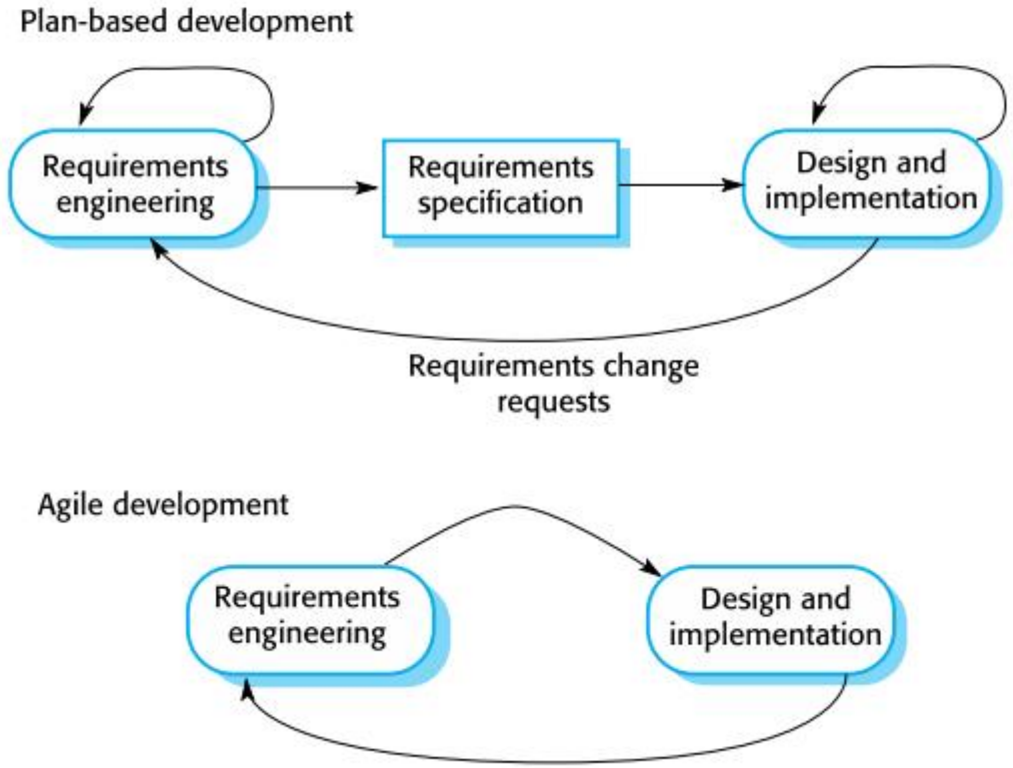
The software process

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

Plan-driven and agile development

- Plan-driven development
  - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
  - Not necessarily waterfall model – plan-driven, incremental development is possible
  - Iteration occurs within activities.
- Agile development
  - Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Plan-driven and agile specification


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

Scaling agile methods

- Agile methods have proved to be successful for small and medium sized projects that can be developed by a small co-located team.
- It is sometimes argued that the success of these methods comes because of improved communications which is possible when everyone is working together.
- Scaling up agile methods involves changing these to cope with larger, longer projects where there are multiple development teams, perhaps working in different locations.

Rapid software development

- Rapid development and delivery is now often the most important requirement for software systems
  - Businesses operate in a fast –changing requirement and it is practically impossible to produce a set of stable software requirements
  - Software has to evolve quickly to reflect changing business needs.
- Rapid software development
  - Specification, design and implementation are inter-leaved
  - System is developed as a series of versions with stakeholders involved in version evaluation
  - User interfaces are often developed using an IDE and graphical toolset.

Technical, human, organizational issues

- Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on:
  - Is it important to have a very detailed specification and design before moving to implementation? If so, you probably need to use a plan-driven approach.
  - Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic? If so, consider using agile methods.
  - How large is the system that is being developed? Agile methods are most effective when the system can be developed with a small co-located team who can communicate informally. This may not be possible for large systems that require larger development teams so a plan-driven approach may have to be used.

Technical, human, organizational issues

- What type of system is being developed?
  - Plan-driven approaches may be required for systems that require a lot of analysis before implementation (e.g. real-time system with complex timing requirements).
- What is the expected system lifetime?
  - Long-lifetime systems may require more design documentation to communicate the original intentions of the system developers to the support team.
- What technologies are available to support system development?
  - Agile methods rely on good tools to keep track of an evolving design
- How is the development team organized?
  - If the development team is distributed or if part of the development is being outsourced, then you may need to develop design documents to communicate across the development teams.

Technical, human, organizational issues

- Are there cultural or organizational issues that may affect the system development?
  - Traditional engineering organizations have a culture of plan-based development, as this is the norm in engineering.
- How good are the designers and programmers in the development team?
  - It is sometimes argued that agile methods require higher skill levels than plan-based approaches in which programmers simply translate a detailed design into code
- Is the system subject to external regulation?
  - If a system has to be approved by an external regulator (e.g. approve software that is critical to the operation of an aircraft) then you will probably be required to produce detailed documentation as part of the system safety case.

Agile methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
  - Focus on the code rather than the design
  - Are based on an iterative approach to software development
  - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
- 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.

The principles of agile methods

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.

Agile method applicability

- Product development where a software company is developing a small or medium-sized product for sale.
- Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software.
- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems.

Problems with agile methods

- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterises agile methods.
- Prioritising changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.

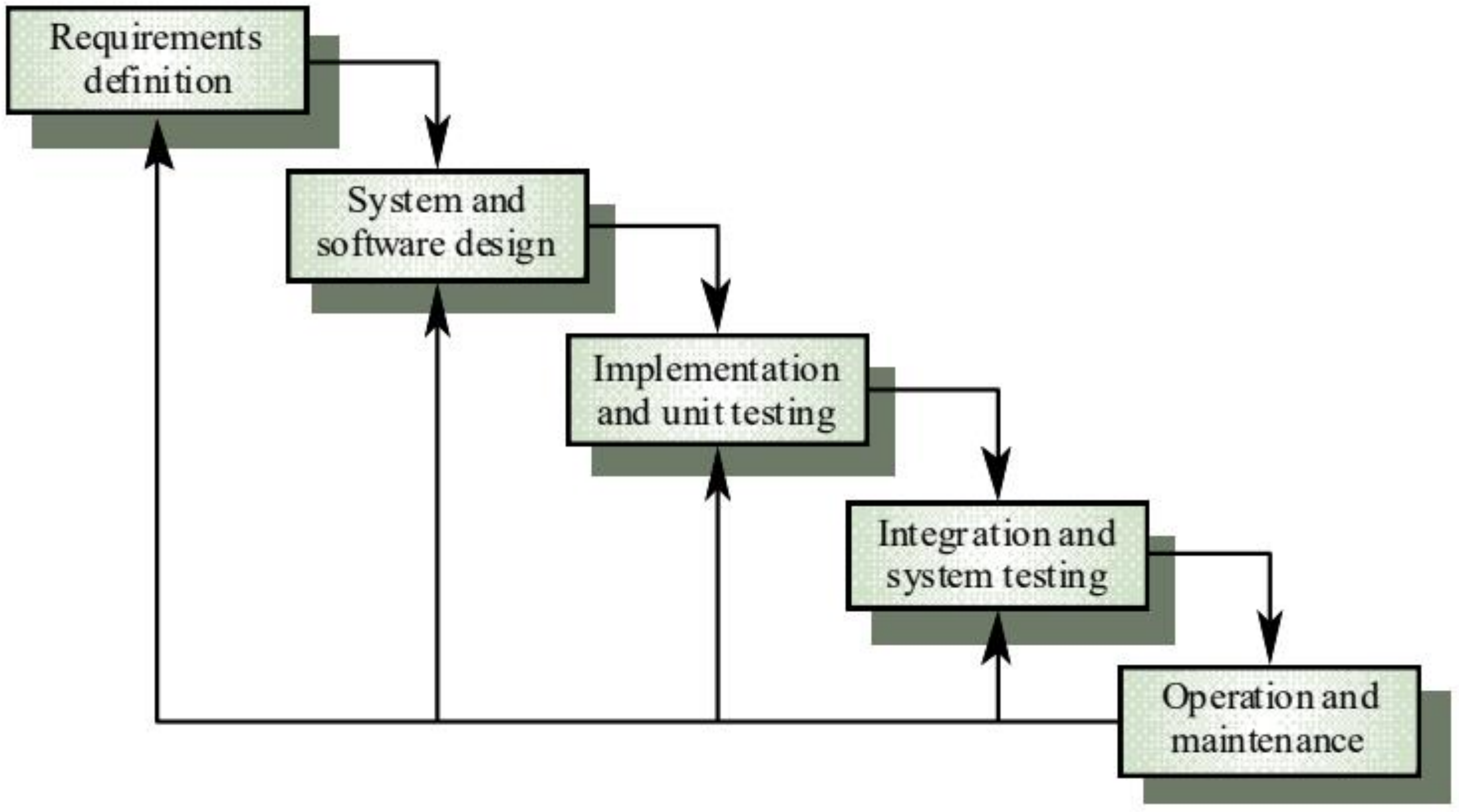
Agile methods and software maintenance

- Most organizations spend more on maintaining existing software than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development.
- Two key issues:
  - Are systems that are developed using an agile approach maintainable, given the emphasis in the development process of minimizing formal documentation?
  - Can agile methods be used effectively for evolving a system in response to customer change requests?
- Problems may arise if original development team cannot be maintained.

Generic software process models

- The waterfall model
  - Separate and distinct phases of specification and development
- Evolutionary development
  - Specification and development are interleaved
- Formal systems development
  - A mathematical system model is formally transformed to an implementation
- Reuse-based development
  - The system is assembled from existing components

1.Waterfall model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Software Process Models: .Waterfall.

- Main characteristics:
  - Also called classic software life cycle or sequential model
  - Process activities (phases/stages) are clearly separated
  - After a number of iterations, phases of the life cycle (such as specification and design).

Software Process Models: ..Waterfall

- Advantages:
  - Organized approach, provides robust separation of phases
  - Reflects common engineering practice
- Disadvantages:
  - Doesn’t cope well with changes required by the client
  - Development teams might wait for each other
  - A working version of the product is available only late
  - The drawback of the waterfall model is the difficulty of accommodating change after the process is underway
- Applicability:
  - When requirements are well known and few changes are likely to be needed
  - Can be used also for parts of larger software systems

Waterfall Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

Waterfall Deficiencies

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

When to use the Waterfall Model

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

Waterfall #1

- Jump to next phase only if the prior one is completed

- PROs
  - Detailed early analysis cause huge advantages at later phases
  - If a bug found earlier, it is much cheaper (and more effective) to fix than bugs found in a later phase
  - Requirement should be set before design starts
  - Points to importance of documentation (minimized “broken leg” issue)
  - Disciplined and well-structured approach
  - Effective for stable software projects
  - Easy to plan from project management point of view

Waterfall #2

- CONs
  - Changes are expensive
  - Client does not explicitly know what he or she wants
  - Client does not explicitly know what is possible to have
  - Need to finish every phase fully
  - Long projects, difficult to keep the plan
  - Designers may not know in advance how complex a feature's implementation

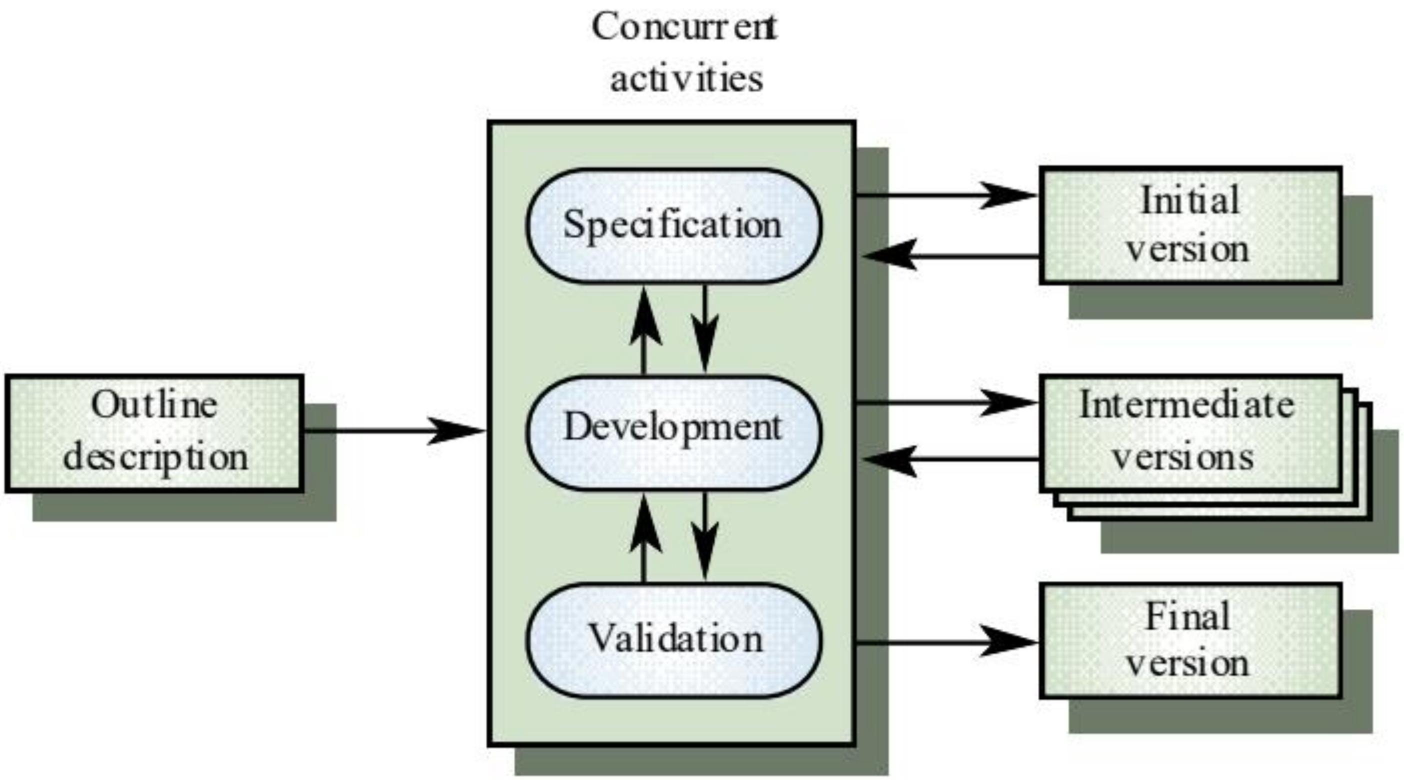
Waterfall model problems

- Inflexible partitioning of the project into distinct stages
- This makes it difficult to respond to changing customer requirements
- Therefore, this model is only appropriate when the requirements are well-understood

2.Evolutionary Development..

- Main characteristics:
  - The phases of the software construction are interleaved
  - Feedback from the user is used throughout the entire process
  - The software product is refined through many versions
- Types of 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

Evolutionary development



Software Process Models: ..Evolutionary Development.

- Advantages:
  - Deals constantly with changes
  - Provides quickly an initial version of the system
  - Involves all development teams
- Disadvantages:
  - Quick fixes may be involved
  - “Invisible” process, not well-supported by documentation
  - The system’s structure can be corrupted by continuous change
  - Special tools and techniques may be necessary
  - The client may have the impression the first version is very close to the final product and thus be less patient

Software Process Models: …Evolutionary Development

- Applicability:
  - When requirements are not well understood
  - When the client and the developer agree on a "rapid prototype" that will be thrown away
  - Good for small and medium-sized software systems

Evolutionary development

- Problems
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g. in languages for rapid prototyping) may be required
- Applicability
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

Process iteration

- System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems
- Iteration can be applied to any of the generic process models
- Two (related) approaches
  - Incremental development
  - Spiral development

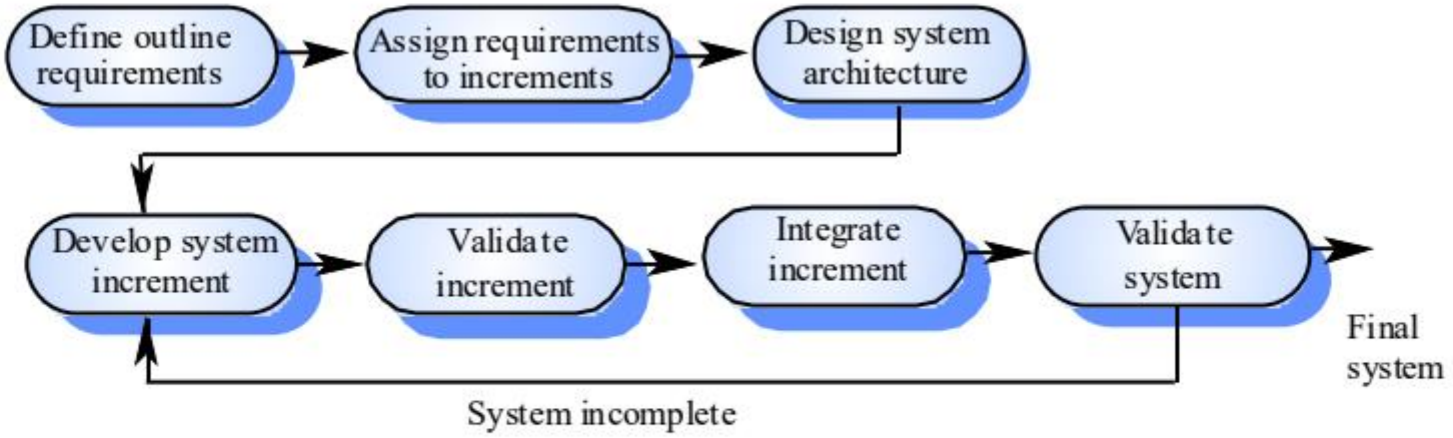
4.Incremental development

- 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
- 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

Software Process Models: .Incremental..

- Main characteristics:
  - Hybrid model that combines elements of the waterfall and evolutionary paradigms
  - The specification, design, and implementation phases are broken in smaller increments

Incremental development



Software Process Models: ..Incremental.

- Advantages:
  - Provides better support for process iteration
  - Reduces rework in the software construction process
  - Some decisions on requirements may be delayed
  - Allows early delivery of parts of the system
  - Supports easier integration of sub-systems
  - Lower risk of project failure
  - Delivery priorities can be more easily set

Software Process Models: ...Incremental

- Disadvantages:
  - Increments need be relatively small
  - Mapping requirements to increments may not be easy
  - Common software facilities may be difficult to identify
- Applicability:
  - When it is possible to deliver the system “part-by-part”

Software Process Models: ..Incremental.

- Advantages:
  - Provides better support for process iteration
  - Reduces rework in the software construction process
  - Some decisions on requirements may be delayed
  - Allows early delivery of parts of the system
  - Supports easier integration of sub-systems
  - Lower risk of project failure
  - Delivery priorities can be more easily set

Incremental development advantages

- Customer value can be delivered with each increment so system functionality is available earlier
- 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

Software Process Models: ...Incremental

- Disadvantages:
  - Increments need be relatively small
  - Mapping requirements to increments may not be easy
  - Common software facilities may be difficult to identify
- Applicability:
  - When it is possible to deliver the system “part-by-part”

Incremental Model Strengths

- Develop high-risk or major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses “divide and conquer” breakdown of tasks
- Lowers initial delivery cost
- Initial product delivery is faster
- Customers get important functionality early
- Risk of changing requirements is reduced

Incremental Model Weaknesses

- Requires good planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required (some will be developed long before others)
- Total cost of the complete system is not lower

When to use the Incremental Model

- Risk, funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology

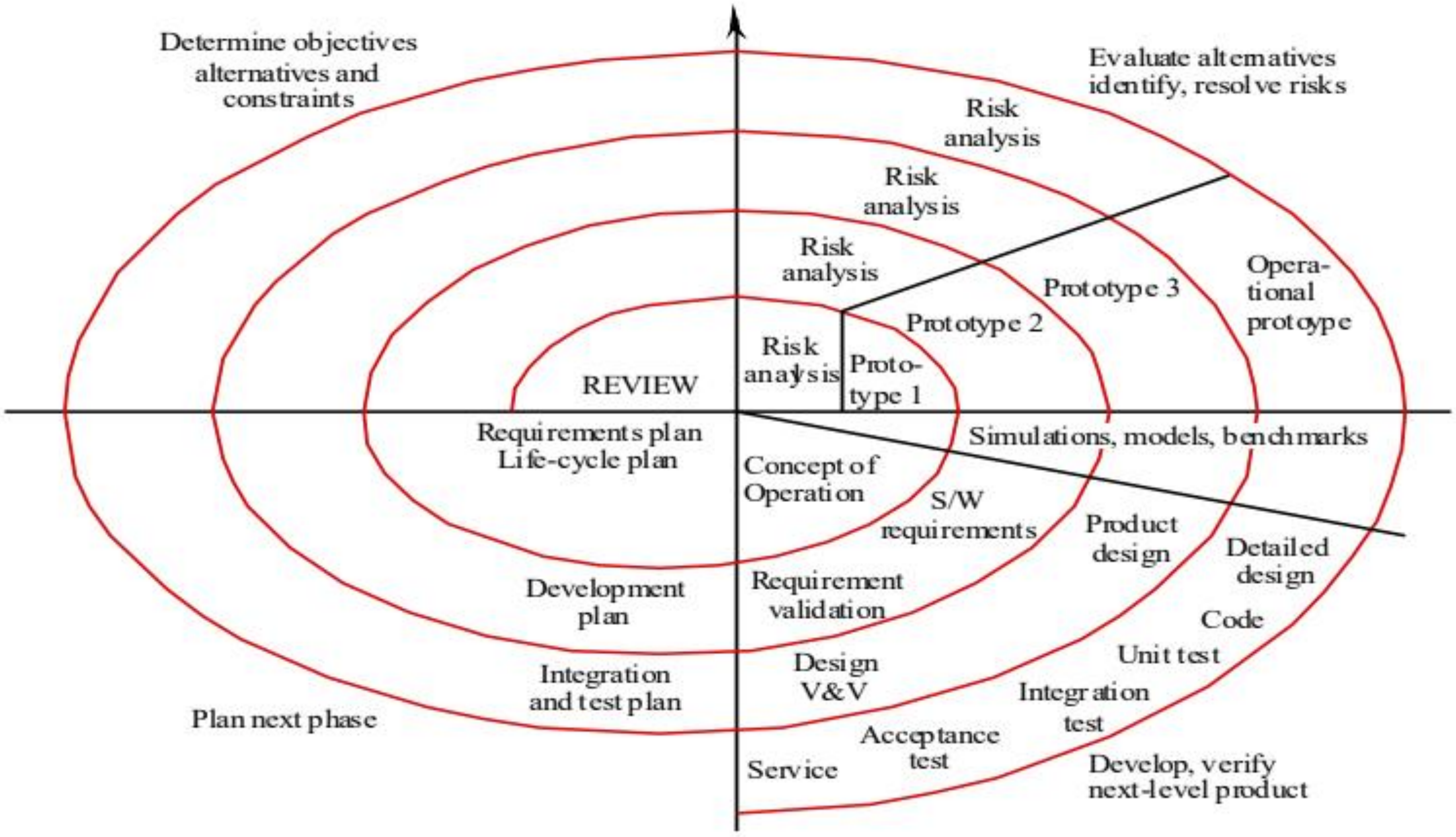
Spiral development

- Process is represented as a spiral rather than as a sequence of activities with backtracking
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process

Software Process Models: .Spiral Model.

- Main characteristics:
    - Also a hybrid model that support process iteration
    - The process is represented as a spiral, each loop in the spiral representing a process phase
    - Four sectors per loop: objective setting, risk assessment and reduction, development and validation, planning
    - Risk is explicitly taken into consideration

Spiral model of the software process



Spiral model sectors

- Objective setting
  - Specific objectives for the phase are identified
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks
- Development and validation
  - A development model for the system is chosen which can be any of the generic models
- Planning
  - The project is reviewed and the next phase of the spiral is planned

Spiral Quadrant

Determine objectives, alternatives and constraints

- Objectives: functionality, performance, hardware/software interface, critical success factors, etc.
- Alternatives: build, reuse, buy, sub-contract, etc.
- Constraints: cost, schedule, interface, etc.

Evaluate alternatives, identify and resolve risks

- Study alternatives relative to objectives and constraints
- Identify risks (lack of experience, new technology, tight schedules, poor process, etc.
- Resolve risks (evaluate if money could be lost by continuing system development

Develop next-level product

- Typical activites:
  - Create a design
  - Review design
  - Develop code
  - Inspect code
  - Test product

Plan next phase

- Typical activities
  - Develop project plan
  - Develop configuration management plan
  - Develop a test plan
  - Develop an installation plan

Software Process Models: ..Spiral Model

- Advantages:
  - Risk reduction mechanisms are in place
  - Supports iteration and reflects real-world practices
  - Systematic approach
- Disadvantages:
  - Requires expertise in risk evaluation and reduction
  - Complex, relatively difficult to follow strictly
  - Applicable only to large systems
- Applicability:
  - Internal development of large systems

Spiral Model Strengths

- Provides early indication of undefeatable risks, without much cost
- Users see the system early because of rapid prototyping tools
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- Early and frequent feedback from users
- Cumulative costs assessed frequently

Spiral Model Weaknesses

- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required
- Spiral may continue indefinitely
- Developers must be reassigned during non-development phase activities
- May be hard to define objective, verifiable milestones that indicate readiness to proceed through the next iteration

When to use Spiral Model

- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

Prototyping

- Prototyping is an information-gathering technique
- Prototypes are useful in seeking user reactions, suggestions, innovations, and revision plans
- Prototyping may be used as an alternative to the systems development life cycle

Prototype Development Guidelines

- Guidelines for developing a prototype are
  - Work in manageable modules
  - Build the prototype rapidly
  - Modify the prototype in successive iterations
  - Stress the user interface

Prototype Advantages

- Potential for changing the system early in its development
- Opportunity to stop development on an unworkable system
- Possibility of developing a system that closely addresses users' needs and expectations

Prototype Disadvantages

- Managing the prototyping process is difficult because of its rapid, iterative nature
- Requires feedback on the prototype
- Incomplete prototypes may be regarded as complete systems

Agile

- Group of software development methods
- Based on iterative and incremental development
- Most important phrases
  - self-organizing, cross-functional teams
  - adaptive planning,
  - evolutionary development and delivery,
  - a time-boxed iterative approach,
  - rapid and flexible response to change.

Agile SDLC’s

- Speed up or bypass one or more life cycle phases
- Usually less formal and reduced scope
- Used for time-critical applications
- Used in organizations that employ disciplined methods

Some Agile Methods

- Rapid Application Development (RAD)
- Extreme Programming (XP)
- Dynamic Software Development Method (DSDM)

What is RAD?

- Rapid Application Development (RAD) is any method of software engineering that leads to faster application development.
- There are a number of different approaches that can be considered Rapid Application Development

Methods of RAD

Use Existing Software Applications

- Some software applications have the flexibility to meet user requirements. They allow the user to:
  - add templates
  - customise the user interface
  - write macros
  - write extra code to add extra functionality to the program (e.g. using languages like VBA, SQL)
- Examples: MS Word, Access, Excel

Methods of RAD

Object-Oriented Programming Languages

- The translation process in OOP languages allow very quick generation of a program
- The user interface can be created quickly
- Library modules may be reused
- Examples: Visual Basic, Hyperscript, Smalltalk

Methods of RAD

Restructure an Existing Program

- Use a software development package to access the code of an existing program. The code can then be changed to meet the requirements of the user
- This can only be done if the developer or user owns the copyright of the existing program or is able to gain the permission of the copyright owner. For this reason, “Open Source” software is becoming very popular.

Personnel Required

- A very small team.
- Might only require the user and the designer or developer
- Might not require a programmer

Rapid Application Model (RAD)

- Requirements planning phase (a workshop utilizing structured discussion of business problems)
- User description phase – automated tools capture information from users
- Construction phase – productivity tools, such as code generators, screen generators, etc. inside a time-box. (“Do until done”)
- Cutover phase -- installation of the system, user acceptance testing and user training

RAD Strengths

- Reduced cycle time and improved productivity with fewer people means lower costs
- Time-box approach mitigates cost and schedule risk
- Customer involved throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs
- Focus moves from documentation to code
- Uses modeling concepts to capture information about business, data, and processes.

RAD Weaknesses

- Accelerated development process must give quick responses to the user
- Risk of never achieving closure
- Hard to use with legacy systems
- Requires a system that can be modularized
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

When to use RAD

- Reasonably well-known requirements
- User involved throughout the life cycle
- Project can be time-boxed
- Functionality delivered in increments
  - There are pressing reasons for speeding up application development
- Low technical risks

Methods of RAD

CASE Tools

Computer-Aided Software Engineering

- There are CASE tools that can:
  - Generate code
  - Analyse code
  - Reverse engineer software
- These tools save time and reduce errors by automating a part of the development process

Self-study and try to explore each type of tools!

CASE Tools Types