Software Processes

- What are they?
- Why study them?
- What are the activities involved?
- What are some specific models?
  - Waterfall
  - Evolutionary
  - Spiral
  - The "Unified Process"
- How can CASE tools support them?

What are software processes?

- Software processes are the activities involved in producing and evolving a software system. They are represented in a software process model
  - Generic process models describe the organisation of software processes
  - Iterative process models describe the software process as a cycle of activities

What are the activities involved?

- General activities are specification, design and implementation
  - Requirements engineering is the process of developing a software specification
  - Design and implementation processes transform the specification to an executable program
  - Validation involves checking that the system meets to its specification and user needs
  - Evolution is concerned with modifying the system after it is in use

Why study software processes?

- "This topic is a waste of time. If you want to build a system, just build it already. Why waste all this time talking about it?"

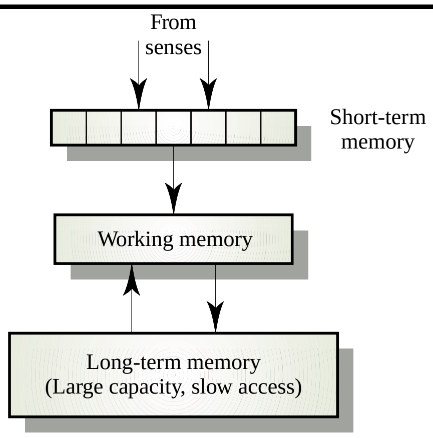
Why study software processes?

- ... besides many other good answers ...
- So that your plans will be accurate models of how time is spent.
- So that you have a high level understanding of the different ways to run a software project, and have fast access to the various activities and issues involved in running a project throughout the entire process.
- How do you maintain fast access? Get the ideas into your long term memory.

Limits to thinking

- People don't all think the same way but everyone is subject to some basic constraints on their thinking due to
  - Memory organisation
  - Knowledge representation
  - Motivation influences
- If we understand these constraints, we can understand how they affect people participating in the software process

Memory organisation



Short-term memory

- Fast access, limited capacity
- 5-7 locations
- Holds 'chunks' of information where the size of a chunk may vary depending on its familiarity
- Fast decay time

Working memory

- Larger capacity, longer access time
- Memory area used to integrate information from short-term memory and long-term memory.
- Relatively fast decay time.

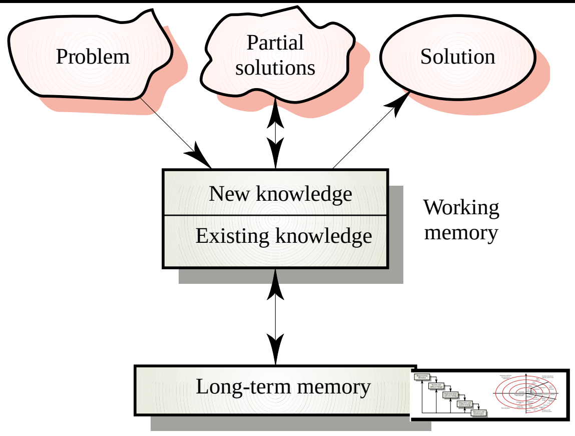
Long-term memory

- Slow access, very large capacity
- Unreliable retrieval mechanism
- Slow but finite decay time - information needs reinforced
- Relatively high threshold - work has to be done to get information into long-term memory.

Information transfer

- Problem solving usually requires transfer between short-term memory and working memory
- Information may be lost or corrupted during this transfer
- Information processing occurs in the transfer from short-term to long-term memory

Problem solving



The software process

- A structured set of activities required to develop a software system
  - Requirements Analysis
  - 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

Software specification

- The process of establishing what services are required and the constraints on the system's operation and development
- Requirements engineering process
  - Feasibility study
  - Requirements elicitation and analysis
  - Requirements specification
  - Requirements validation

Requirements Analysis

- Produce specification of what the software must do
  - User requirements; may be divided into problem analysis and solution analysis
  - Suppress the "how" until design phase
  - Must be understandable to the client (if they must sign off on the analysis), which in practice means it is necessarily somewhat informal
  - To the extent possible, should be precise, complete, unambiguous, and modifiable; Should include object acceptance tests and a system test plan

Question

- (Sommerville 3.4) Why is it important to make a distinction between the developing the user requirements and developing the system requirements in the requirements engineering process?

Software design and implementation

- The process of converting the system specification into an executable system
- Software design
  - Design a software structure that realises the specification
- Implementation
  - Translate this structure into an executable program
- The activities of design and implementation are closely related and may be inter-leaved

Software validation

- Validation and verification are intended to show that a system conforms to its specification and meets the requirements of the system customer
  - Validation: Are we building the right product?
  - Verification: Are we building the product right?
- Involves checking and review processes and system testing
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system

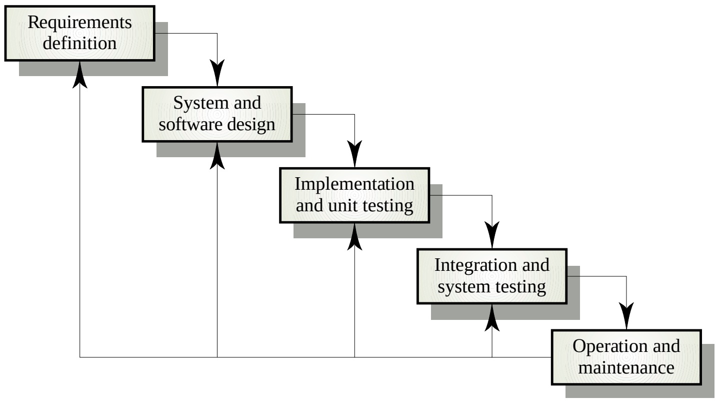
Software evolution

- (Not the same thing as evolutionary development)
- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new

Software process models

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

Waterfall model


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

Waterfall model phases

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance
- The drawback of the waterfall model is the difficulty of accommodating change after the process is underway

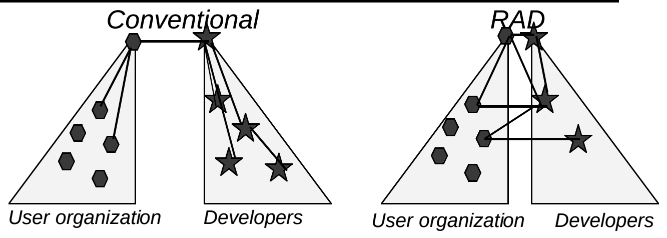
Waterfall model problems

- 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

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
- Then iteratively build the product with intense user involvement to negotiate requirements and test deliverables
- Two specific methodologies: Rapid Appliation Development (RAD) and Joint Application Development (JAD).

RAD communication structure



- Peer-to-peer communication between users and developers
- Intense user involvement (and commitment) in negotiating requirements and testing prototypes
- Emphasis on rapid delivery and change, not on preserving information for a longer period. Hence, reduced paper documentation
  - Active, intense user participation among fixed personnel (including user representatives) reduces need for documents as orientation and communication.

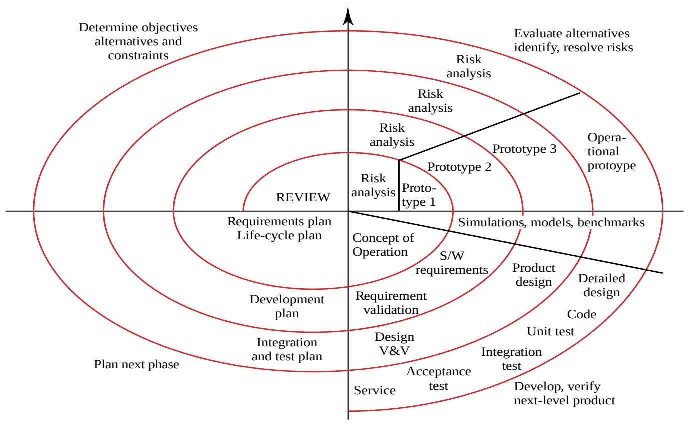
Evolutionary development

- 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

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

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

Question

- (Sommerville 3.3) How can both the waterfall and prototyping models of software development be accommodated in the spiral model process?

The Unified Software Development Process

- A process framework for transforming user requirements into a software system.
  - Use-case driven: All analyses are derived from specific user-system interactions. The requirements analysis asks “What is the system supposed to do *for each user*?”
  - Architecture-centric: Emphasis on the entire design of the system, with important characteristics made visible, and details hidden inside.
  - Iterative and incremental: In every iteration: Identify and specify relevant use cases, create a design using the chosen architecture as a guide, implement the design in components, and verify that the components satisfy the use cases.
- Object-oriented, Uses the Unified Modeling Language (UML)

Automated process support (CASE)

- Computer-aided software engineering (CASE) is software to support software development and evolution processes
- Activity automation
  - Graphical editors for system model development
  - Data dictionary to manage design entities
  - Graphical UI builder for user interface construction
  - Debuggers to support program fault finding
  - Automated translators to generate new versions of a program

Case technology

- Case technology has led to significant improvements in the software process though not the order of magnitude improvements that were once predicted
  - Software engineering requires creative thought - this is not readily automatable
  - Software engineering is a team activity and, for large projects, much time is spent in team interactions. CASE technology does not really support these

CASE classification

- Classification helps us understand the different types of CASE tools and their support for process activities
- Functional perspective
  - Tools are classified according to their specific function
- Process perspective
  - Tools are classified according to process activities that are supported
- Integration perspective
  - Tools are classified according to their organisation into integrated units

Functional tool classification

Tool type	**Examples**
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

Summary

- Introduced software process models
- Outlined requirements engineering, software development, testing and evolution
- Described a number of different process models and how and when they may be used
- Introduced CASE technology, which supports software process activities