Questions

• What is the life cycle of a software product?
• Why do we need software process models?
• What are the goals of a software process and what makes it different from other industrial processes?
Software Processes

• Coherent sets of activities for specifying, designing, implementing and testing software systems
## Product stages

<table>
<thead>
<tr>
<th>Products</th>
<th>Measurement/Evaluation</th>
<th>Doc</th>
<th>Team work</th>
<th>Mgmt Of Artifacts</th>
<th>Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arch/Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment/Maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Product, process stages

<table>
<thead>
<tr>
<th>Processes</th>
<th>Measurement/Evaluation</th>
<th>Doc</th>
<th>Teamwork</th>
<th>Mgmt Of Artifacts</th>
<th>Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arch/Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment/Maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is a software process model?

- A simplified representation of a software process, presented from a specific perspective
- Examples of process perspectives are
  - Workflow perspective - sequence of activities
  - Data-flow perspective - information flow
  - Role/action perspective - who does what
- Generic process models
  - Waterfall
  - Evolutionary development
  - Formal transformation
  - Integration from reusable components
**Software process model**

- **Attempt to organize the software life cycle by**
  - defining activities involved in software production
  - order of activities and their relationships
- **Goals of a software process**
  - standardization, predictability, productivity, high product quality, ability to plan time and budget requirements
The earliest approach

- Write code
- Fix it to eliminate any errors that have been detected, to enhance existing functionality, or to add new features
- Source of difficulties and deficiencies
  - impossible to predict
  - impossible to manage
Models are needed

- Symptoms of inadequacy: the software crisis
  - scheduled time and cost exceeded
  - user expectations not met
  - poor quality
- The size and economic value of software applications required appropriate "process models"
"determine the order of stages involved in software development and evolution, and to establish the transition criteria for progressing from one stage to the next. These include completion criteria for the current stage plus choice criteria and entrance criteria for the next stage. Thus a process model addresses the following software project questions:
   What shall we do next?
   How long shall we continue to do it?"
Process as a "black box"
Problems

• The assumption is that requirements can be fully understood prior to development

• Interaction with the customer occurs only at the beginning (requirements) and end (after delivery)

• Unfortunately the assumption almost never holds
Process as a "white box"
Advantages

• Reduce risks by improving visibility
• Allow project changes as the project progresses
  • based on feedback from the customer
Feasibility study

• Why a new project?
  • cost/benefits tradeoffs
  • buy vs make
    • Requires to perform preliminary requirements analysis
  • Produces a Feasibility Study Document
    1. Definition of the problem
    2. Alternative solutions and their expected benefits
    3. Required resources, costs, and delivery dates in each proposed alternative solution
Little-JIL notation

- Pre-requisite badge
- Interface badge
- Step name
- Post-requisite badge
- Step bar
- Sequencing badge
- Reactions badge
- Handlers badge
Example process

Preorder

Out: customer ID, service types
Out: order
Out: bundle update

Process Request
Form Order
Provide Service
Assign Customer To Bundle
Update Package
Assign Package To Bundle

ID Customer
Receive Service Request
Add Feature
Update Services
Input example (BOOD process)
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
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
Rapid application development model

- Requirements are well understood
- Fourth generation techniques are used
- The system must be modularizable
- High performance very difficult to obtain
- New technologies are high risk
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

Concurrent activities

Outline description

Specification

Development

Validation

Initial version

Intermediate versions

Final version
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
Formal systems development

• Based on the transformation of a mathematical specification through different representations to an executable program

• Transformations are ‘correctness-preserving’ so it is straightforward to show that the program conforms to its specification

• Embodied in the ‘Cleanroom’ approach to software development
Formal systems development

- Requirements definition
- Formal specification
- Formal transformation
- Integration and system testing
Formal transformations

Proofs of transformation correctness
Formal systems development

• Problems
  • Need for specialised skills and training to apply the technique
  • Difficult to formally specify some aspects of the system such as the user interface

• Applicability
  • Critical systems especially those where a safety or security case must be made before the system is put into operation
Reuse-oriented development

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems
- Process stages
  - Component analysis
  - Requirements modification
  - System design with reuse
  - Development and integration
- This approach is becoming more important but still limited experience with it
Reuse-oriented development

1. Requirements specification
2. Component analysis
3. Requirements modification
4. System design with reuse
5. Development and integration
6. System validation
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
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.
Incremental development

Define outline requirements → Assign requirements to increments → Design system architecture

Develop system increment → Validate increment → Integrate increment → Validate system

System incomplete → Final system
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
Extreme programming

• New approach to development based on the development and delivery of very small increments of functionality
• Relies on constant code improvement, user involvement in the development team and pairwise programming
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

- Determine objectives
- Identify, resolve risks
- Plan next phase
- Evaluate alternatives
- Concept of Operation
- Simulations, models, benchmarks
- Develop, verify next-level product
- Detailed design
- Code
- Integrations test
- Unit test
- Acceptance test
- Service
- Design V&V
- Requirements plan
- Life-cycle plan
- Development plan
- Integration and test plan
- REVIEW
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