(Traditional) Software Development Activities
Goals of the Unit

• A gentle and high-level introduction to software development activities

• Understanding what are the building blocks for producing software

• Remarks:
  – This is no substitute for a software engineering course
  – The activities need to be integrated in a coherent process, to make sense
  – Software development projects range from the very small to the very large ... not all activities equally useful or relevant in any context
Overview

• Software development is a progressive refinement which moves from concept to operations through the following phases:
  – Requirements and User Experience Design
  – Design
  – Implementation
  – Verification and Validation
  – Deployment
  – Operations and Maintenance

• As we move along these phases, we make and commit to specific choices; the cost of changes increases accordingly

• Different processes put different emphasis on each activity or define the order in which these activities can be performed
The software process

• Software Process: a coherent sets of activities for specifying, designing, implementing and testing software systems

• The software process manages the transition from concept to product

• A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective

• Some characteristics of a process:
  – the building blocks
  – the order in which the building blocks are executed
  – inputs/outputs to each activity
  – level of formality (when and how you move to the next activity; what you really need to trace)
# The Building Blocks: the “what”

<table>
<thead>
<tr>
<th>Business/Environment Modeling</th>
<th>Describes the environment (people, organization, other systems) in which the software in going to be run</th>
<th>A textual description, diagrams (BPMN, UML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements (the what)</td>
<td>Describes the external qualities of the software (what the users sees) and the constraints (for instance: execution environment)</td>
<td>A textual description, GUI layouts, use cases, user stories, ...</td>
</tr>
</tbody>
</table>
Example

• Business Modeling
  1. Unit Head receives expense request
  2. Unit Head authorizes and forward to administration
  3. Administration verifies budget availability and forwards to procurement
  4. Procurement proceeds
  5. ....

• Requirements
  1. The system shall allow a unit head to authorize an expense request
  2. ....

FURPS
- Functionality
- Usability
- Reliability
- Performance
- Supportability
# The Building Blocks: the “how”

<table>
<thead>
<tr>
<th>Architectural Design (the how)</th>
<th>Describes how the system is structured in components</th>
<th>Diagrams (class, components, ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation (the actual stuff!)</td>
<td>Implementation of the requirements according to the structure defined in the architectural design</td>
<td>Code</td>
</tr>
</tbody>
</table>
Example (architectural diagrams)

• Several notations:
  – Data flow diagrams
  – Hierarchical decomposition
  – State diagrams
  – Class diagrams
  – ...

• Common goals:
  – What components a system is made of (logical/physical)
  – How they interact
  – How they behave
  – ...

Control Logic

Actuator

Driver

Show <<view>>
display

Person <<model>>
name
surname
get_fullname

PersonLogic <<controller>>
change_name
change_surname

Show

Driver

Control Logic
## The Building Blocks: “v&v”

<table>
<thead>
<tr>
<th>Verification (does it work right?)</th>
<th>It verifies that the implementation works as expected</th>
<th>A report (and the specification of the tests - code or text)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation (is it the right thing?)</td>
<td>It verifies whether the system does what was intended in the requirements</td>
<td>A report (and the specification of the tests - typically: text)</td>
</tr>
</tbody>
</table>

- **Different scopes/representations:**
  - **unit** (test a piece of code): executable code
  - **component/system** (test a component): textual specifications/executable scripts

- **Different types:**
  - **white box testing** (you know the source code)
  - **black box testing** (you don’t know the internals)
Examples

• Textual/Executable lists of tests

• Often in the form:
  – with the system in a given state
  – if some events occur (e.g. user interaction)
  – then some output is expected

• Example:
  – after successful login as a “simple user”
  – if the user presses the “admin” button it is taken to an error page displaying (“user is not authorized”)

• It can be automated (even with Graphical User Interfaces)
# The Building Blocks: “going live”

<table>
<thead>
<tr>
<th>User Documentation</th>
<th>User Manual and other information to support users about the functions and usage of the system</th>
<th>Manual!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution and Deployment</td>
<td>Activities related to going “live” and reaching the customer</td>
<td>Software Installation package, Appstore, Website, Web Application, ...</td>
</tr>
<tr>
<td>Maintenance</td>
<td>What you do after the software has been released (sometimes mentioned; modern software development approaches treat maintenance as a new project)</td>
<td></td>
</tr>
</tbody>
</table>
(Some) “Transversal Concerns”

<table>
<thead>
<tr>
<th><strong>Business Plan Definition</strong></th>
<th>How you are going to make money out of your system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration Management</strong></td>
<td>The activities related to maintaining the coherency of a system when it evolves</td>
</tr>
<tr>
<td><strong>Project Management</strong></td>
<td>The activities to keep the project under control</td>
</tr>
<tr>
<td><strong>Process Improvement</strong></td>
<td>Measuring your performances and making sure you get better at building software</td>
</tr>
</tbody>
</table>