Software Development Best Practices

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Root Causes</th>
<th>Best Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Needs not met</td>
<td>Insufficient requirements</td>
<td>Develop Iteratively</td>
</tr>
<tr>
<td>Requirements churn</td>
<td>Ambiguous communications</td>
<td>Manage Requirements</td>
</tr>
<tr>
<td>Modules don't fit</td>
<td>Brittle architectures</td>
<td>Use Component Architectures</td>
</tr>
<tr>
<td>Hard to maintain</td>
<td>Overwhelming complexity</td>
<td>Model Visually (UML)</td>
</tr>
<tr>
<td>Late discovery</td>
<td>Undetected inconsistencies</td>
<td>Continuously Verify Quality</td>
</tr>
<tr>
<td>Poor quality</td>
<td>Poor testing</td>
<td></td>
</tr>
<tr>
<td>Poor performance</td>
<td>Subjective assessment</td>
<td></td>
</tr>
<tr>
<td>Colliding developers</td>
<td>waterfall development</td>
<td></td>
</tr>
<tr>
<td>Build-and-release</td>
<td>Uncontrolled change</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Insufficient automation</td>
<td>Manage Change</td>
</tr>
</tbody>
</table>
Practice 1: Develop Iteratively

Best Practices
Process Made Practical

Develop Iteratively
Manage Requirements
Use Component Architectures
Model Visually (UML)
Continuously Verify Quality
Manage Change

Waterfall Development Characteristics

Waterfall Process

- Delays confirmation of critical risk resolution
- Measures progress by assessing work-products that are poor predictors of time-to-completion
- Delays and aggregates integration and testing
- Precludes early deployment
- Frequently results in major unplanned iterations
Iterative Development Produces Executables

Each iteration results in an executable release.

Practice 2: Manage Requirements

Best Practices
Process Made Practical

Develop Iteratively
Manage Requirements
Use Component Architectures
Model Visually (UML)
Continuously Verify Quality
Manage Change
Requirements Management Means…

Making sure you
- solve the right problem
- build the right system
by taking a systematic approach to
- eliciting
- organizing
- documenting
- managing
the changing requirements of a software application.

Map of the Territory
Use Cases as a Basis for Iteration Planning

During Elaboration, use cases are implemented to validate the architecture.

Use Cases as a Basis for System Modeling

- **Use-Case Model** (requirements)
  - realization
  - influence
  - verification

- **Design Model** (classes and objects)
- **Implementation Model** (source code)
- **Test Scripts**
Use Cases as a Basis for Test Planning

The complete behavior of a use case is tested using Test Scripts and Test Suites.

Practice 3: Use Component Architectures

Best Practices
Process Made Practical

Develop Iteratively
Manage Requirements
**Use Component Architectures**
Model Visually (UML)
Continuously Verify Quality
Manage Change
Purpose of a Component-Based Architecture

- Basis for reuse
  - Component reuse
  - Architecture reuse
- Basis for project management
  - Planning
  - Staffing
  - Delivery
- Intellectual control
  - Manage complexity
  - Maintain integrity

Practice 4: Model Visually (UML)

Best Practices
Process Made Practical

Develop Iteratively
Manage Requirements
Use Component Architectures
Model Visually (UML)
Continuously Verify Quality
Manage Change
Role of UML in RUP

- Rational Unified Process was developed hand-in-hand with the UML.
- Many artifacts in Rational Unified Process have a UML representation.
- Rational Unified Process also includes guidelines for UML concepts.
Practice 5: Continuously Verify Quality

**Best Practices**
*Process Made Practical*

- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually (UML)
- **Continuously Verify Quality**
- Manage Change

---

**Continuously Verify Your Software’s Quality**

Software problems are 100 to 1000 times more costly to find and repair after deployment

- Cost to Repair Software
- Cost to Lost Opportunities
- Cost of Lost Customers

---

Page 9
Testing Dimensions of Quality

- **Usability**: Test application from the perspective of convenience to end-user.
- **Functionality**: Test the accurate workings of each usage scenario.
- **Reliability**: Test that the application behaves consistently and predictably.
- **Supportability**: Test the ability to maintain and support application under production use.
- **Performance**: Test online response under average and peak loading.

Test Each Iteration

- **UML Model and Implementation**
  - Iteration 1
  - Iteration 2
  - Iteration 3
  - Iteration 4

- **Tests**
  - Test Suite 1
  - Test Suite 2
  - Test Suite 3
  - Test Suite 4

Need for automated testing increases.
Practice 6: Manage Change

Best Practices
Process Made Practical

Develop Iteratively
Manage Requirements
Use Component Architectures
Model Visually (UML)
Continuously Verify Quality
Manage Change

Muutostenhallinta - Change Request Management

Change requests come from many sources throughout each iteration of the product lifecycle

Single Channel for Approval
Approved Decision Process (CCB)

New Feature
New Requirement
Bug
Change Request (CR)

Req
Design
Code
Test
Maint

Customer and User inputs
Marketing
Coders inputs Testers inputs
Help Desk User inputs
Unified Change Management

Change Request Management (CRM)

Measurement

Configuration Management (CM)

Best Practices Reinforce Each Other: An Example

Best Practices

- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually (UML)
- Continuously Verify Quality
- Manage Change

Ensures users involved as requirements evolve
Validates architectural decisions early on
Addresses complexity of design/implementation incrementally
Measures quality early and often
Evolves baselines incrementally
RUP Implements Best Practices

Best Practices
Process Made Practical
- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually (UML)
- Continuously Verify Quality
- Manage Change

Bringing It All Together: The Iterative Approach

In an iteration, you walk through all disciplines.

Disciplines group related activities.
### Changing Focus of Phases Over Time

#### Planned (Business) Decision Points

<table>
<thead>
<tr>
<th>Event</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scope and Business Case agreement</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Architecture baselined</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Product sufficiently mature for customers to use</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acceptance or end of life</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Understand problem**
- Inception

**Understand solution**
- Elaboration

**Have a solution**
- Construction

**Efficient use of solution**
- Transition

#### Planned (Technical) Visibility Points

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Typical Effort and Time Percentages by Phase

#### Effort

<table>
<thead>
<tr>
<th>Phase</th>
<th>Inc</th>
<th>Elab</th>
<th>Const</th>
<th>Trans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort</td>
<td>5%</td>
<td>20%</td>
<td>65%</td>
<td>10%</td>
</tr>
</tbody>
</table>

#### Time/Schedule

<table>
<thead>
<tr>
<th>Phase</th>
<th>Inc</th>
<th>Elab</th>
<th>Const</th>
<th>Trans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time/Schedule</td>
<td>10%</td>
<td>30%</td>
<td>50%</td>
<td>10%</td>
</tr>
</tbody>
</table>