Rational Unified Process (RUP)

Process Meta-model
Inception & Elaboration

These notes adopted and slightly modified from "RUP Made Easy" provided by the IBM Academic Initiative

Agenda

• Iterative Development with RUP
• Inception Phase & Objectives
• Elaboration Phase & Objectives
• Project and Iteration Planning
RUP: Pros and Cons

Pros:
- Popular language (UML) and process model
- Good tool support
- Core workflow definitions and incorporation of BP
- Explicit recognition of deployment activities
- Architecture-centric / Component-based

Cons
- Some “gaps” or disconnects in the model
- Pitched as a “all-or-nothing” process commercially
  - IBM now owns Rational – is this a good or bad thing?
  - Tries to be “all things to all people”

When to use?
- Suggested for medium to large scale projects

Iterative Development and RUP

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
</table>

**Inception:** Understand what to build
- Vision, high-level requirements, business case
- Not detailed requirements

**Elaboration:** Understand how to build it
- Baseline architecture, most requirements detailed
- Not detailed design

**Construction:** Build the product
- Working product, system test complete

**Transition:** Validate solution
- Stakeholder acceptance

*An iteration is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an executable release.*

Inception: Know What to Build

Prepare vision document and initial business case
- Include risk assessment and resource estimate

Develop high-level project requirements
- Initial use-case and domain models (10-20% complete)

Manage project scope
- Reduce risk by identifying all key requirements
- Acknowledge that requirements will change
  - Manage change, use iterative process
Objectives with Inception

1. Understand what to build
   - Vision, including who wants the system and it’s value
   - The scope of the system
   - Preliminary use-case model
2. Identify key requirements
   - Critical use cases and non-functional requirements
3. Determine at least one potential solution
   - Identify candidate architecture
4. Understand costs, schedule and risk
   - Business case, Software Development Plan, and Risk List
5. Understand what process to follow and tools to use
   - RUP configuration, development case, and customized tools

Objective 1: Understand What to Build

Agree on a high-level vision
Provide a “mile-wide, inch-deep” description
   - Identify as many actors as you can
   - Associate each of the actors with use cases
   - Find additional actors for each use case
   - Briefly describe each actor (1-2 sentences) and use case (1-2 paragraphs)
   - Create a Glossary
   - Do a lifecycle analysis of key glossary items
   - Identify most essential and critical use cases (<20% of use cases)

Detail key actors and use cases
   - Done for 10-20% most critical use cases
   - Outline main flow of events
   - Identify alternative flow of events
   - Complement textual descriptions with use-case prototypes
   - Time-box the writing, you never get “done”
   - Use less detail
     - Small projects
     - Analyst and developer

Detail key non-functional requirements
**Obj. 2: Identify Key System Functionality**

Functionality is the core of the application
- Exercises **key interfaces**
- Deals with risks related to performance, redundancy, data security, …
- Example: “Check Out” for e-commerce application

**Functionality must be delivered**
- Captures the **essence** of the system
- Example: “Book conference room” for conference room booking system

**Functionality covers an otherwise untouched area of the system**
- May **conceal** unexpected technical difficulties

**Obj. 3: Determine at Least 1 Potential Solution**

Should answer questions providing major impact on
- Can you build the application with a **sensible amount** of risk at a reasonable cost?
  - Have you built similar systems?
  - With what architecture at what cost?
  - Will current architecture work, or will rework be required?
- Staffing profile – Will you be able to acquire personnel with the right competency to succeed?
- Required target environment – If it will have major impact on the **cost** profile of the overall project
- Required software components? Can they be purchased? At a reasonable **cost**?

**Obj. 4: Understand Cost, Schedule & Risk**

**Business case**
- Ultimately answers the question: Should we fund the project?
- **Cost**
- **Return of Investment**

**Software Development Plan**
- Coarse project plan
- Resource needs
Objective 5: Decide on Process & Tools

Decide on RUP configuration
- Select plug-ins and process components
- Produce process views

Decide on development case
- What artifacts should be produced with what formality

Tool deployment plan
- Required customizations
- Reusable assets

Common Pitfalls in Inception

Too much formality / too many artifacts
- Only produce the artifacts that add value, minimize formality if possible
- When in doubt of value, don’t do it

Analysis Paralysis
- You can improve upon things later on – move on
- Focus on objectives with Inception
- Do NOT describe all requirements in detail

Too long initial iteration
- Cut scope rapidly
- You fail with first iteration, project likely to fail

Project Review: Lifecycle Objective Milestone

Do you have agreement on:
- Scope definition
- Key requirements have been captured
- Cost and schedule estimates
- Priorities understood
- Development process and tools defined
- Initial risks identified and risk mitigation strategy exists
Elaboration: Know How to Build It

Detail requirements as necessary (~80% complete)
- Less essential requirements may not be fleshed out
- Produce an executable and stable architecture
- Define, implement and test interfaces of major components
- Identify dependencies on external components and systems.
- Some key components will be partially implemented
- Roughly 10% of code is implemented.

Drive architecture with key use cases
- 20% of use cases drive 80% of the architecture
- Design, implement and test key scenarios for use cases
- Verify architectural qualities
- Reliability (Stress test), Scalability and Performance (Load test)

Continuously assess business case, risk profile & dev plan

Objectives with Elaboration

1. Get a more detailed understanding of requirements
   - Move from 20% to 80% of requirements captured
2. Design, implement, validate and baseline the architecture
   - Make critical design decisions: buy vs. build, patterns, ...
   - Baseline a skeleton structure of your system
   - Perform initial load and performance test
3. Mitigate essential risks, and produce more accurate schedule and cost estimates
   - You now know what to build, and have reduced risk => More accurate schedule
4. Fine-tune and deploy development environment
   - Harvest experiences from Inception
   - Rollout development environment

Obj. 1: Get a Detailed Understanding of Reqs.

By end of elaboration
- Detail ~80% of use cases
- Produce prototypes of graphically oriented use cases (at least mock-up screens)
- Walk through use cases with stakeholders
- For use case with partial implementations—demo
- Detail non-functional requirements (all that have an impact on arch!)
- Time box! You will never be done, you can fix issues in later iterations

What is not done
- Use cases with no or very limited associated risk (if you have done one “print” use case, do you need to do more?)
- Use cases that are expected to be volatile, and have little impact on end solution or stakeholder satisfaction
Objective 2: Design, Implement, Validate and Baseline the Architecture

The most important building blocks of the systems
- Build, buy, or reuse?

Interaction between these building blocks to provide key scenarios
- Required to verify the architecture

Run-time architecture
- Processes, threads, nodes, ...

Architectural patterns
- Dealing with persistence, inter-process communication, recovery, authentication, garbage collection, ...

Test framework allowing testing of key capabilities
- Performance, scalability, reliability, load, protocols, and other key non-functional requirements

Obj. 3: Mitigate Essential Risks, & Produce More Accurate Schedule & Cost Estimates

Most key risks addressed
- Technical risks by implementing and testing the architecture
- Business risks by implementing and testing key functionality
- Team- and tool-oriented risks by having the team going through the full software lifecycle implementing real code, using the tools at hand

Schedule & cost estimates can be radically improved as we:
- Have mitigated key risks
- Understand a vast majority of the requirements
- Understand which building blocks needs to be implemented and tested
- Understand which building blocks can be acquired, and to what cost
- Understand how effective our team is

Objective 4: Fine-tune & Deploy Dev Env

Fine-tune your development case based on the experiences so far

Do customizations and improvements of the tool environment as required

Make it easy for all team members to find and use reusable assets, including the architectural patterns you developed in Elaboration

Do training of your team and deploy tools
Proj. Review: Lifecycle Architecture Milestone

- Are the product Vision and requirements **stable**?
- Is the architecture **stable**?
- Are the key approaches to be used in testing and evaluation proven?
- Have testing and evaluation of executable prototypes demonstrated that the **major risk elements have been addressed and resolved**?
- Are the iteration plans for Construction of sufficient detail and fidelity to allow the work to proceed?
- Are the iteration plans for the Construction phase supported by credible estimates?
- Are actual resource expenditures versus planned expenditures acceptable?

Do all stakeholders agree that the current Vision, as defined in the Vision Document, can be met if the current plan is executed to develop the complete system in the context of the current architecture?