Rational unified process

Introduction

Unified Process—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML) to model and develop object-oriented system iteratively and incrementally.

Introduction

- RUP was originally developed by Rational Software (now part of IBM)
  - It is a Software engineering process
  - It is a process product
  - It enhances team productivity
  - It creates and maintains models
  - It is a guide to effectively use the Unified Modeling Language
- Its goal is to delivery a high quality product that the customer actually wants

Introduction

- The RUP is not a concrete development model, but rather is intended to be adaptive and tailored to the specific needs of your project, team, or organization.
- The RUP is based on a few fundamental ideas, such as the phases of development and the building blocks, which define who, what, when, and how development will take place.

RUP six fundamental best practices

- The Rational Unified Process is structured around six fundamental best practices, which are so-named due to their common use throughout the industry:

RUP six fundamental best practices

- Develop Software Iteratively:
  - Encourages iterative development by locating and working on the high-risk elements within every phase of the software development life cycle.

RUP six fundamental best practices

- Manage Requirements:
  - Describes how to organize and keep track of functionality requirements, documentation, tradeoffs and decisions, and business requirements.

RUP six fundamental best practices

- Use Component-Based Architectures: Emphasizes development that focuses on software components which are reusable through this project and, most importantly, within future projects.

RUP six fundamental best practices

- Visually Model Software:
  - Based on the Unified Modeling Language (UML), the Rational Unified Process provides the means to visually model software, including the components and their relationships with one another.

RUP six fundamental best practices

- **Verify Software Quality**:
  - Assists with design, implementation, and evaluation of all manner of tests throughout the software development life cycle.

RUP six fundamental best practices

- **Control Changes to Software**:
  - Describes how to track and manage all forms of change that will inevitably occur throughout development, in order to produce successful iterations from one build to the next.

The Building Blocks

- All aspects of the Rational Unified Process are based on a set of building blocks, which are used to describe
  - what should be produced,
  - who is in charge of producing it,
  - how production will take place, and
  - when production is complete.
- There are four building blocks.

The Building Blocks

- Workers, the 'Who':
  - The behavior and responsibilities of an individual, or a group of individuals together as a team, working on any activity in order to produce artifacts.

The Building Blocks

- Activities, the 'How':
  - A unit of work that a worker is to perform.
  - Activities should have a clear purpose, typically by creating or updating artifacts.

The Building Blocks

- Artifacts, the 'What':
  - An artifact represents any tangible output that emerges from the process; anything from new code functions to documents to additional life cycle models.
- Workflows, the 'When':
  - Represents a diagrammed sequence of activities, in order to produce observable value and artifacts.

Workflows, the 'When

- Workflows are further divided up in the Rational Unified Process into six core engineering workflows

Workflows, the 'When

- Business Modeling Workflow:
  - During this workflow, the business context (scope) of the project should be outlined.
- Requirements Workflow:
  - Used to define all potential requirements of the project, throughout the software development life cycle.

Workflows, the 'When

- Analysis & Design Workflow:
  - Once the requirements workflow is complete, the analysis and design phase takes those requirements and transforms them into a design that can be properly implemented.
- Implementation Workflow:
  - This is where the majority of actual coding takes place, implementing and organizing all the code into layers that make up the whole of the system.

Workflows, the 'When

- Test Workflow:
  - Testing of all kinds takes place within this workflow.
- Deployment Workflow:
  - Finally, the deployment workflow constitutes the entire delivery and release process, ensuring that the software reaches the customer as expected.

supporting workflows

- There are also three core supporting workflows defined in the Rational Unified Process:
- Project Management Workflow:
  - Where all activities dealing with project management take place, from pushing design objectives to managing risk to overcoming delivery constraints.

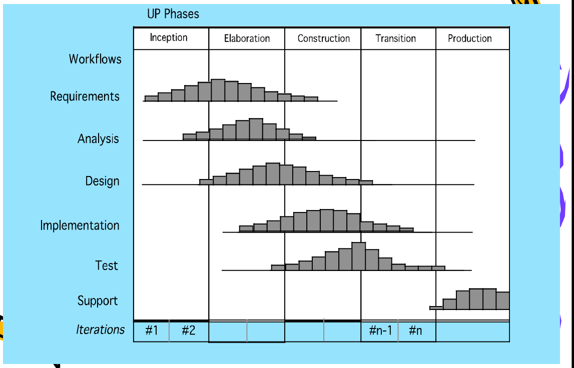
supporting workflows

- Configuration & Change Management Workflow:
  - Used to describe the various artifacts produced by the development team, ideally ensuring that there is minimal overlap or wasted efforts performing similar activities that result in identical or conflicting artifacts.

supporting workflows

- Environment Workflow:
  - this workflow handles the setup and management of all software development environments throughout the team, including the processes, as well as the tools, that are to be used throughout the software development life cycle.

UP Phases



The Four Life Cycle Phases

- Inception Phase
  - During the inception phase, the basic idea and structure of the project is determined.
  - The team will sit down and determine if the project is worth pursuing at all, based on the proposed purpose of the project, the estimated costs (monetary and time), and what resources will be required to complete the project once the green light is given.

Inception Phase

- The conclusion of the inception phase is the Lifecycle Objectives Milestone, which consists of the following evaluation criteria:
  - Stakeholder concurrence on scope definition and cost/schedule estimates.
  - Requirements understanding as evidenced by the fidelity of the primary use cases.

Inception Phase

- Credibility of the cost/schedule estimates, priorities, risks, and development process.
- Depth and breadth of any architectural prototype that was developed.
- Actual expenditures versus planned expenditures.

Elaboration Phase

- The purpose of the elaboration phase is to analyze the requirements and necessary architecture of the system.
- The success of this phase is particularly critical, as the final milestone of this phase signifies the transition of the project from low-risk to high-risk, since the actual development and coding will take place in the following phase.

Elaboration Phase

- The Lifecycle Architecture Milestone signifies the end of the elaboration phase, and is evaluated using these criteria:
  - Is the vision of the product stable?
  - Is the architecture stable?
  - Does the executable demonstration show that the major risk elements have been addressed and credibly resolved?

Elaboration Phase

- Is the plan for the construction phase sufficiently detailed and accurate? Is it backed up with a credible basis of estimates?
- Do all stakeholders agree that the current vision can be achieved if the current plan is executed to develop the complete system, in the context of the current architecture?
- Is the actual resource expenditure versus planned expenditure acceptable?

Construction Phase

- As the meat and potatoes of the software development life cycle, the construction phase is when the coding and implementation of all application features will take place.
- This period is also where integrations with other services or existing software should occur.

Construction Phase

- The end of the construction phase is measured by the completion of the Initial Operational Capability Milestone, which is based on these criteria:
  - Is this product release stable and mature enough to be deployed in the user community?

Construction Phase

- Are all stakeholders ready for the transition into the user community?
- Are the actual resource expenditures versus planned expenditures still acceptable?

Transition Phase

- Easier thought of as deployment, the transition phase is when the finished product is finally released and delivered to customers.
- However, the transition phase is more than just the process of deployment; it must also handle all post-release support, bug fixes, patches, and so forth.

Transition Phase

- The Product Release Milestone signals the end of the transition phase, and is based on a few simple questions:
  - Is the user satisfied?
  - Are the actual resources expenditures versus planned expenditures still acceptable?

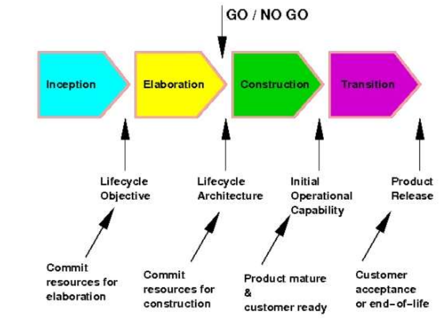
Iterations

- The Rational Unified Process also recommends that each of the four above phases be further broken down into iterations, a concept taken from agile and other common iterative development models.

Iterations

- Just as with other models, in the context of the Rational Unified Process, an iteration simply represents a full cycle of the aforementioned core phases, until a product is released in some form
- From this baseline, the next iteration can be modified as necessary until, finally, a full and complete product is released to customers.

Phases of Rup


GO / NO GO
Inception
Elaboration
Construction
Transition
Lifecycle Objective
Lifecycle Architecture
Initial Operational Capability
Product Release
Commit resources for elaboration
Commit resources for construction
Product mature & customer ready
Customer acceptance or end-of-life

Advantages of RUP

- Allows for the adaptive capability to deal with changing requirements throughout the development life cycle, whether they be from customers or from within the project itself.
- Emphasizes the need (and proper implementation of) accurate documentation.

Advantages of RUP

- Diffuses potential integration headaches by forcing integration to occur throughout development, specifically within the construction phase where all other coding and development is taking place.

Disadvantages of RUP

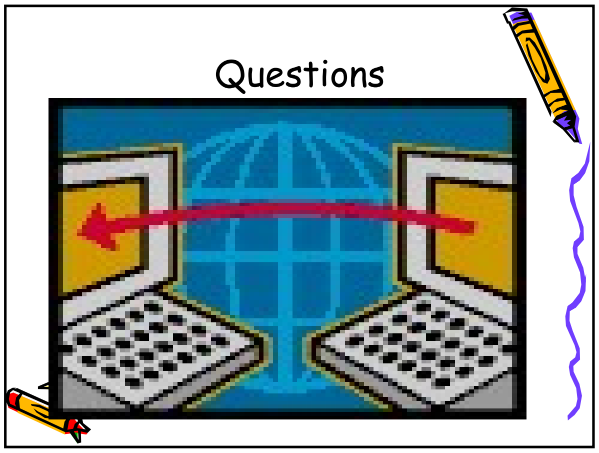
- Heavily relies on proficient and expert team members, since assignment of activities to individual workers should produce tangible, pre-planned results in the form of artifacts.

Disadvantages of RUP

- Given the emphasis on integration throughout the development process, this can also be detrimental during testing or other phases, where integrations are conflicting and getting in the way of other, more fundamental activities.

DisAdvantages of RUP

- Arguably, rup is a fairly complicated model.
- Given the assortment of the components involved, including best practices, phases, building blocks, milestone criteria, iterations, and workflows, often proper implementation and use of the RUP can be challenging for many organizations, particularly for smaller teams or projects.



Questions