Universitas Esa Unggul

Smart, Creative and Entrepreneurial

CPL230-PENGEMBANGAN PERANGKAT LUNAK (PERTEMUAN-2)

www.esaunggul.ac.id

Dosen Pengampu :

5165-Kundang K Juman

Prodi Teknik Informatika Fakultas Ilmu Komputer

NJIT

New Jersey's Science & Technology University

THE EDGE IN KNOWLEDGE

George Blank
University Lecturer

Iterative, Evolutionary, and Agile

Introduction to the Rational Unified Process

Grady Booch speaks

- “People are more important than any process.
- Good people with a good process will outperform good people with no process any time.”

The Unified Process

- The Unified Process has emerged as a popular and effective software development process.
- In particular, the Rational Unified Process, as modified at Rational Software, is widely practiced and adopted by industry.

The Most Important Concept

- The critical idea in the Rational Unified Process is *Iterative Development*.
- Iterative Development is successively enlarging and refining a system through multiple iterations, using feedback and adaptation.
- Each iteration will include requirements, analysis, design, and implementation.
- Iterations are *timeboxed*.

What is Rational Unified Process (RUP)?

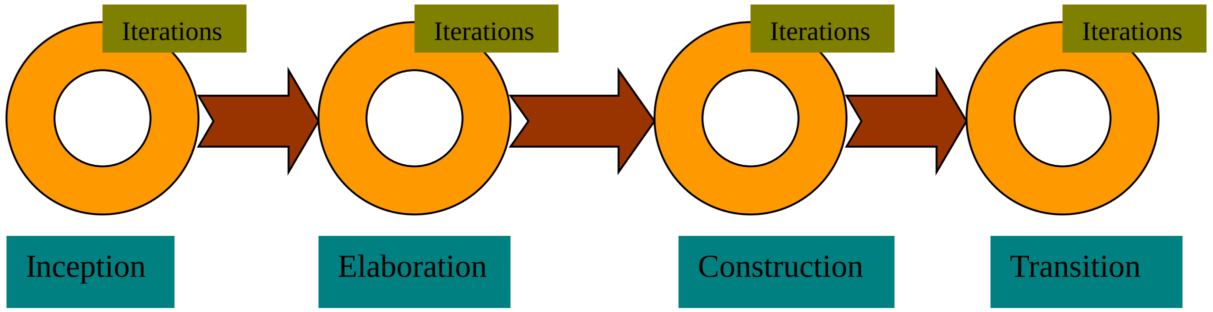
- RUP is a complete software-development process framework , developed by Rational Corporation.
- It’s an iterative development methodology based upon six industry-proven best practices.
- Processes derived from RUP vary from lightweight—addressing the needs of small projects —to more comprehensive processes addressing the needs of large, possibly distributed project teams.

Phases in RUP

- RUP is divided into four phases, named:
- Inception
- Elaboration
- Construction
- Transition

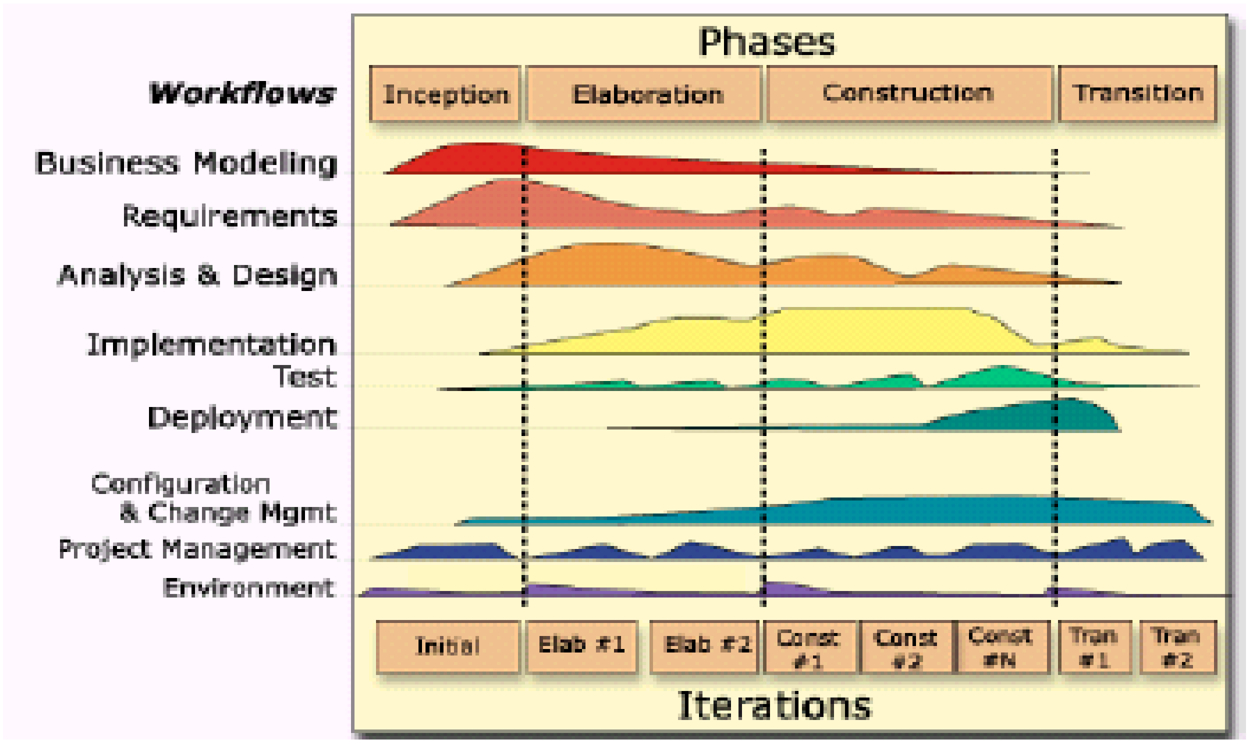
Iterations

EEach phase has iterations, each having the purpose of producing a demonstrable piece of software. The duration of iteration may vary from two weeks or less up to six months.



The iterations and the phases fig 1

Resource Histogram



RUP Overview Diagram

The iterations and the phases fig 2

The Agile Manifesto

- Individuals and interactions
  - Over processes and tools
- Working software
  - Over comprehensive documentation
- Customer collaboration
  - Over contract negotiation
- Responding to change
  - Over following a plan

Unified Process best practices

- Get high risk and high value first
- Constant user feedback and engagement
- Early cohesive core architecture
- Test early, often, and realistically
- Apply use cases where needed
- Do some visual modeling with UML
- Manage requirements and scope creep
- Manage change requests and configuration

Inception

- The life-cycle objectives of the project are stated, so that the needs of every stakeholder are considered. Scope and boundary conditions, acceptance criteria and some requirements are established.

Inception - Entry criteria

- The expression of a need, which can take any of the following forms:
  - an original vision
  - a legacy system
  - an RFP (request for proposal)
  - the previous generation and a list of enhancements
  - some assets (software, know-how, financial assets)
  - a conceptual prototype, or a mock-up

Inception - Activities

- **Formulate the scope of the project.**
- Needs of every stakeholder, scope, boundary conditions and acceptance criteria established.
- **Plan and prepare the business case.**
- Define risk mitigation strategy, develop an initial project plan and identify known cost, schedule, and profitability trade-offs.
- **Synthesize candidate architecture.**
- Candidate architecture is picked from various potential architectures
- **Prepare the project environment.**

Inception - Exit criteria

- An initial business case containing at least a clear formulation of the product vision - the core requirements - in terms of functionality, scope, performance, capacity, technology base.
- Success criteria (example: revenue projection).
- An initial risk assessment.
- An estimate of the resources required to complete the elaboration phase.

Elaboration

- An analysis is done to determine the risks, stability of vision of what the product is to become, stability of architecture and expenditure of resources.

Elaboration - Entry criteria

- The products and artifacts described in the exit criteria of the previous phase.
- The plan approved by the project management, and funding authority, and the resources required for the elaboration phase have been allocated.

Elaboration - Activities

- **Define the architecture.**
- Project plan is defined. The process, infrastructure and development environment are described.
- **Validate the architecture.**
- **Baseline the architecture.**
- To provide a stable basis for the bulk of the design and implementation effort in the construction phase.

Elaboration - Exit criteria

- A detailed software development plan, with an updated risk assessment, a management plan, a staffing plan, a phase plan showing the number and contents of the iteration , an iteration plan, and a test plan
- The development environment and other tools
- A baseline vision, in the form of a set of evaluation criteria for the final product
- A domain analysis model, sufficient to be able to call the corresponding architecture ‘complete’.
- An executable architecture baseline.

Construction

- The Construction phase is a manufacturing process. It emphasizes managing resources and controlling operations to optimize costs, schedules and quality. This phase is broken into several iterations.

Construction - Entry criteria

- The product and artifacts of the previous iteration. The iteration plan must state the iteration specific goals
- Risks being mitigated during this iteration.
- Defects being fixed during the iteration.

Construction - Activities

- **Develop and test components**.
- Components required satisfying the use cases, scenarios, and other functionality for the iteration are built. Unit and integration tests are done on Components.
- **Manage resources and control process**.
- **Assess the iteration**
- Satisfaction of the goal of iteration is determined.

Construction - Exit Criteria

- The same products and artifacts, updated, plus:
- A release description document, which captures the results of an iteration
- Test cases and results of the tests conducted on the products,
- An iteration plan, detailing the next iteration
- Objective measurable evaluation criteria for assessing the results of the next iteration(s).

Transition

- The transition phase is the phase where the product is put in the hands of its end users. It involves issues of marketing, packaging, installing, configuring, supporting the user-community, making corrections, etc.

Transition - Entry criteria

- The product and artifacts of the previous iteration, and in particular a software product sufficiently mature to be put into the hands of its users.

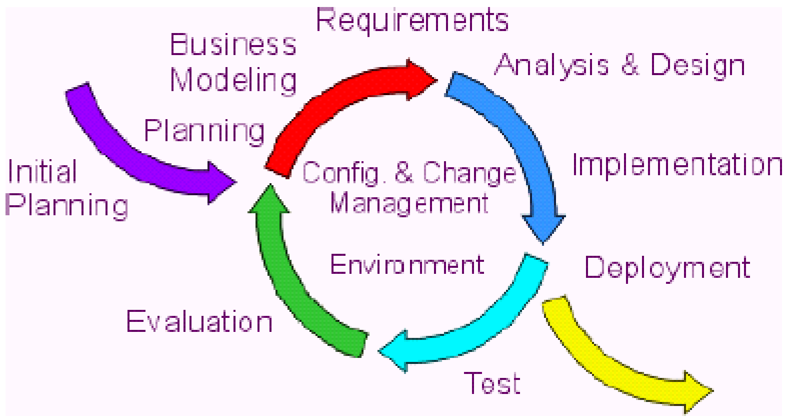
Transition - Activities

- Test the product deliverable in a customer environment.
- Fine tune the product based upon customer feedback
- Deliver the final product to the end user
- Finalize end-user support material

Transition - Exit criteria

- An update of some of the previous documents, as necessary, the plan being replaced by a “post-mortem” analysis of the performance of the project relative to its original and revised success criteria;
- A brief inventory of the organization’s new assets as a result this cycle.

Modeling Disciplines of RUP



Typical Iteration Flow

1. **Business Modeling**

- The purpose of this discipline is to model the business context and the scope of your system. This workflow is done usually in Inception and Elaboration phase.

Activities include the development of:
-A context model showing how the system fits into its overall environment
-A high-level business requirements model eg. use case model
-A domain model eg. class diagram or data diagram depicting major business classes or entities
-A business process model

2. **Requirements**

   The purpose of this discipline is to engineer the requirements for the project, including their identification, modeling, and documentation. The main deliverable of this discipline is the Software Requirements Specification (SRS), also referred to as the Requirements Model, which encompasses the captured requirements.

3. **Analysis & Design**
   The purpose of this discipline is to evolve a robust architecture for the system based on the requirements, to transform the requirements into a design, and to ensure that implementation environment issues are reflected in the design.

4. Environment

The purpose of this discipline is to support development work. Practically all the work in this workflow are done in Inception phase.The activities include

- implementing and configuring RUP , selecting and acquiring required tools, developing in-house tools
- providing software and hardware maintenance and training.

5. Implementation
6. Test
7. Configuration and Change Management
8. Project Management

PS: See appendix for flowcharts

Practices

- ***Develop software iteratively***
- Software must be developed in small increments and short iterations
- ***Manage requirements***
- Requirements that change over time and those requirements that have greater impact on project goals must be identified
- ***Use component-based architecture*** Components that are most likely to change and components that can be re-used must be identified and built

- ***Visually model software***

Models must be built using visualization methods like UML, to understand the complexity of the system

- ***Verify software quality***

Testing must be done to remove defects at early stages, thus reducing the costs at later stages

- ***Control software changes***

Any changes to requirements must be managed and their effect on software should be traceable.

Life-Cycle Artifacts

- Rational suggests the following typical set of
- documents.
- **Management artifacts:**
- **A**rtifacts used to drive or monitor the progress of the project, estimate the risks, adjust the resources, give visibility to the customer or the investors.
- Technical artifacts:
- Artifacts that are either the delivered goods, executable software and manuals, or the blueprints that were used to manufacture the delivered goods.

Management artifacts:

- An *organizational policy* document
- A *Vision* document
- A *Business Case* document
- A *Development Plan* document
- An *Evaluation Criteria* document
- *Release Description* documents for each release
- *Deployment* document
- *Status Assessment* documents

Technical artifacts:

- *User's Manual*
- *Software documentation*, preferably in the form of self-documenting source code, and models (use cases, class diagrams, process diagrams, etc.) captured and maintained with appropriate CASE tools.
- A *Software Architecture* document, describing the overall structure of the software: class categories, classes, processes, subsystems, the definition of critical interfaces, and rationale for the key design decisions.

Roles and Responsibilities

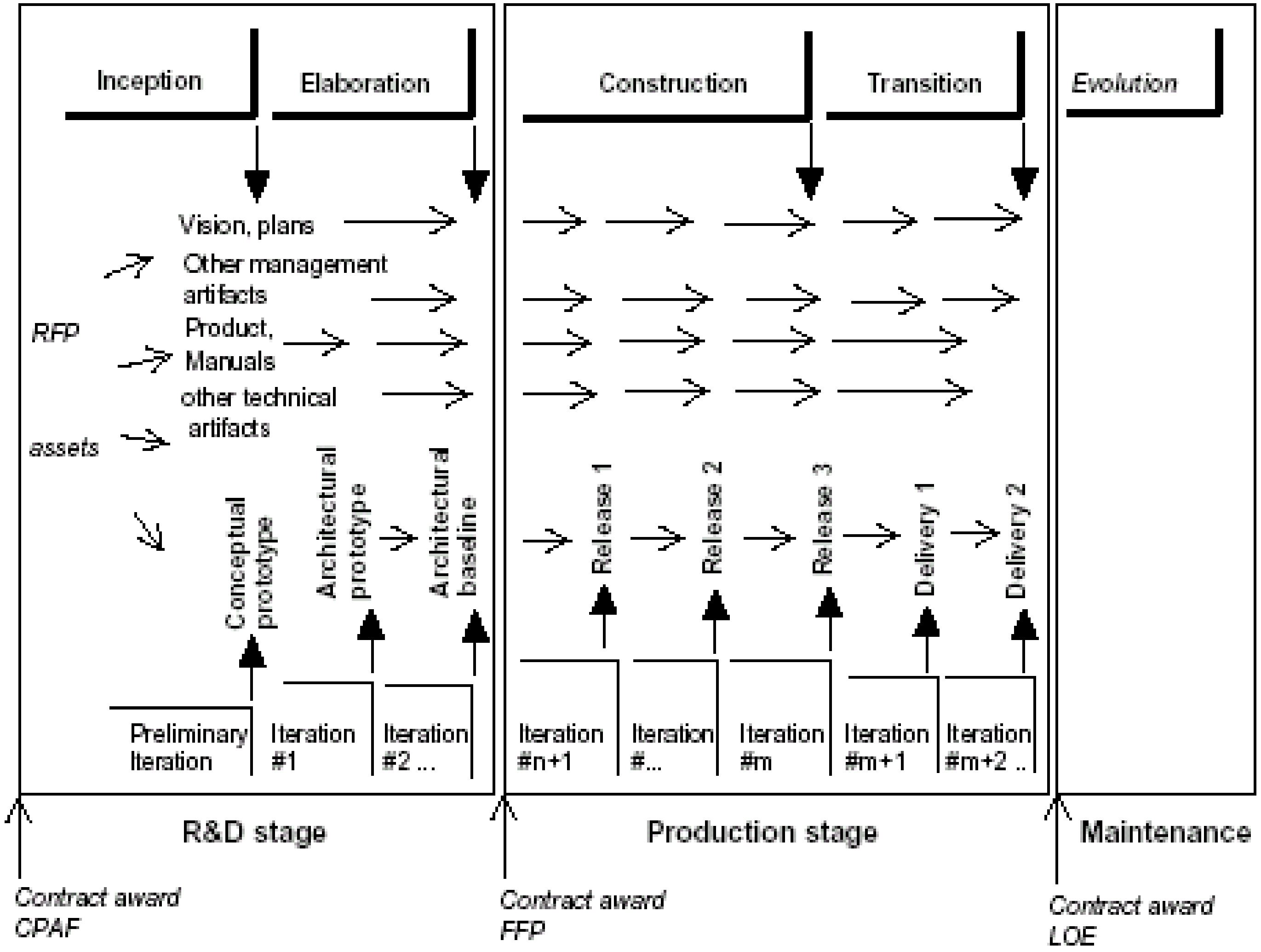
- RUP defines thirty roles called ***workers***. Roles are assigned for each activity. Besides the conventional roles (Architect, Designer, Design Reviewer, Configuration Manager etc), specific roles are assigned in Business Modeling and Environment workflows:
- Business-Process Analyst
- Business designer
- Business Model Reviewer
- Course developer
- Tool smith

Examples of RUP

- **RUP for Large Contractual Software**
- **development:**
- Rational proposes the procurement of large
- software in 3 stages, associated with 3 different
- kinds of contract.
- An **R&D** stage, comprising the inception and elaboration phase, typically bid in a risk sharing manner, e.g., as a cost plus award fee contract (CPAF).

- A **production** stage, comprising the construction and transition phases, typically bid as a firm, fixed price contract (FFP).
- A **maintenance** stage if any, corresponding to the evolution phase, typically bid as a level of effort contract (LOE).

Illustration



RUP for a small commercial software product

- A small commercial development would see
- a more fluid process, with only limited
- amount of formalism at the major
- milestones and a more limited set of
- documents:
  - a product *vision*
  - a *development plan*

- *Release description* documents, specifying the goal of an iteration at the beginning of the iteration, and updated to serve as release notes at the end.
- *User documentation*, as necessary
- Software architecture, software design, development process and procedures

Advantages of RUP

- The RUP puts an emphasis on addressing very early high risks areas.
- It does not assume a fixed set of firm requirements at the inception of the project, but allows to refine the requirements as the project evolves.
- It does not put either a strong focus on documents or ‘ceremonies’
- The main focus remains the software product itself, and its quality.

Drawbacks of RUP

- RUP is not considered particularly “agile” However, recent studies have shown that by adopting the right essential artifacts RUP is agile.
- It fails to provide any clear implementation guidelines.
- RUP leaves the tailoring to the user entirely.

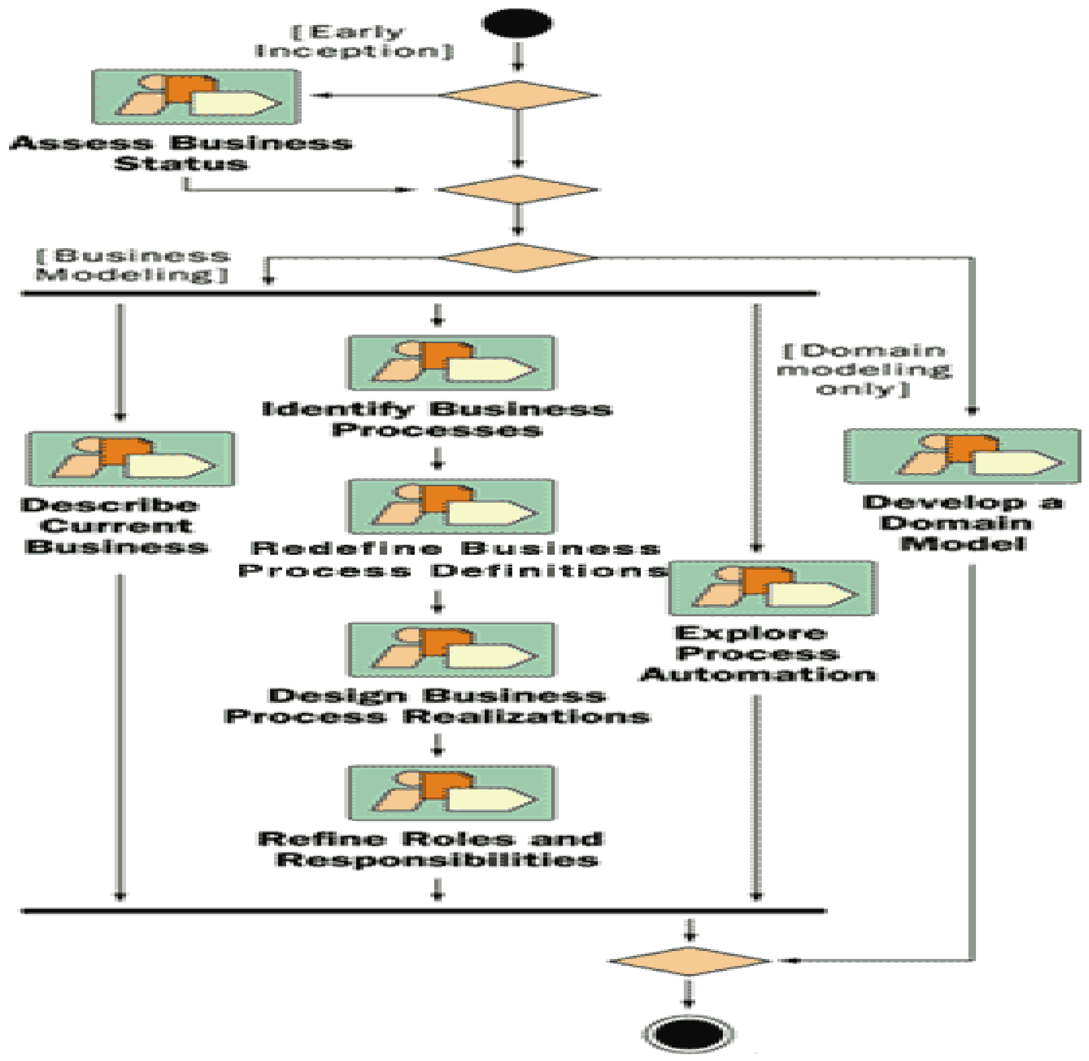
References

- *Agile software development methods* – review and analysis by Pekka Abrahamsson, Outi Salo & Jussi Ronkainen
- http://www.rational.com
- *Using the Rational unified Process for small projects* – Gary Pollice, Rational software

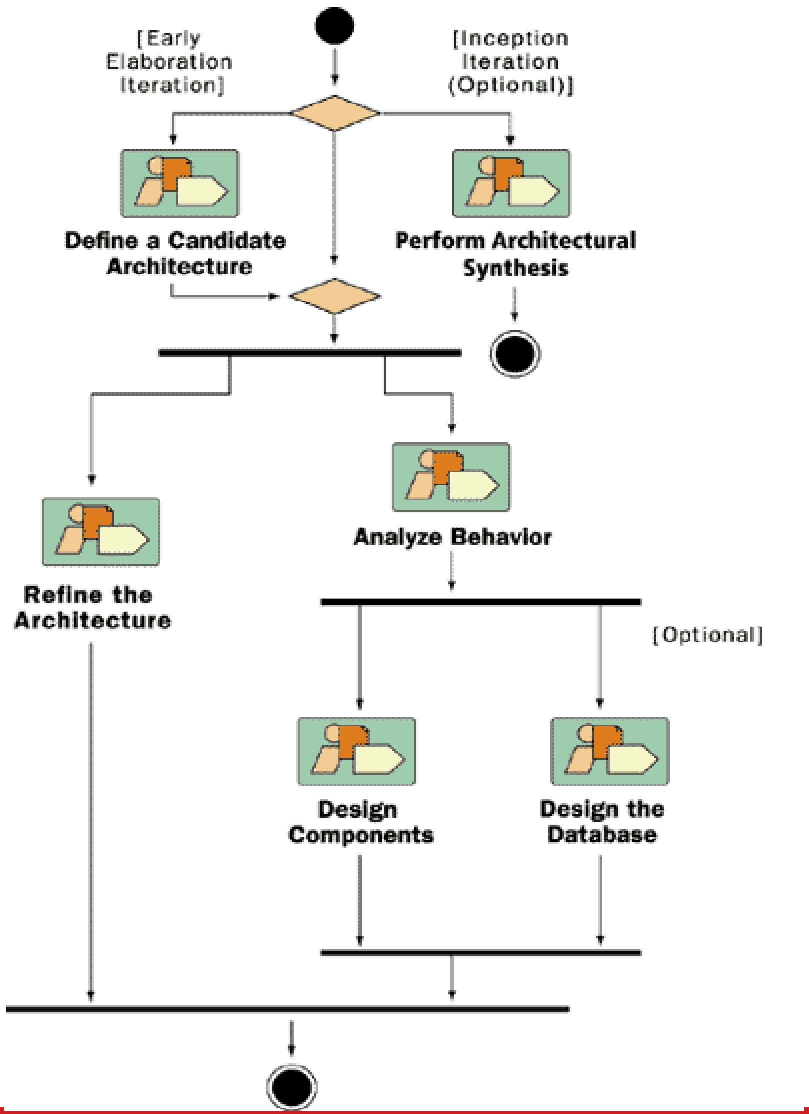
Appendix

- **Flow-charts for the workflows of RUP**

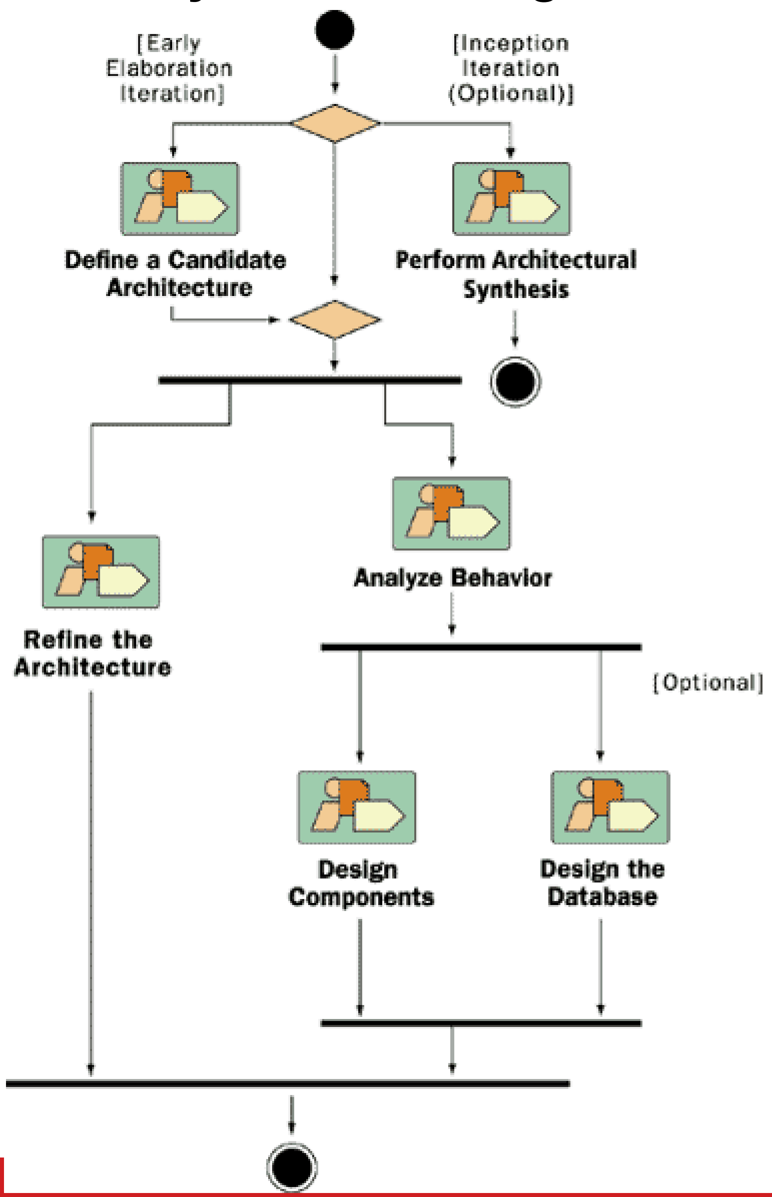
Business Modeling


[Early Inception]
Assess Business Status
[Business Modeling]
Identify Business Processes
[Domain modeling only]
Describe Current Business
Develop a Domain Model
Redefine Business Process Definitions
Explore Process Automation
Design Business Process Realizations
Refine Roles and Responsibilities

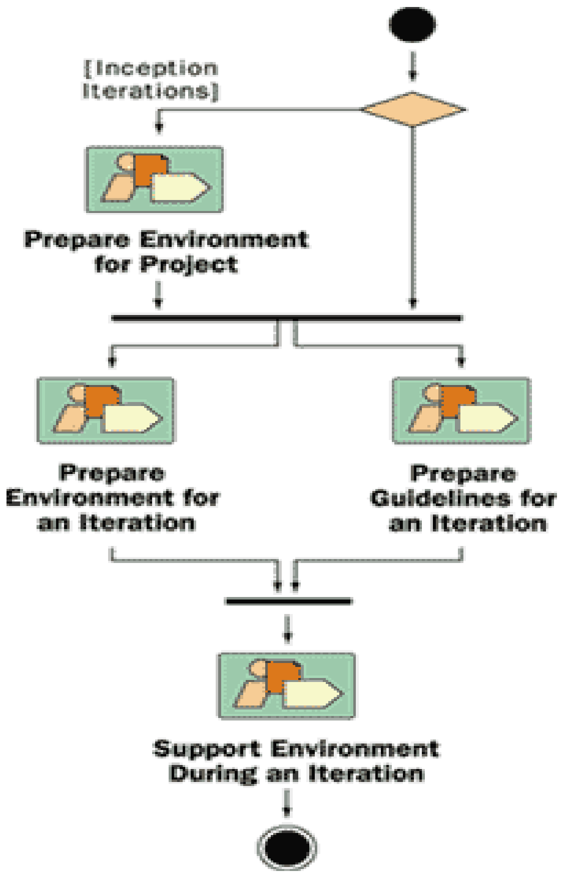
Requirements


[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Refine the Architecture
Analyze Behavior
[Optional]
Design Components
Design the Database

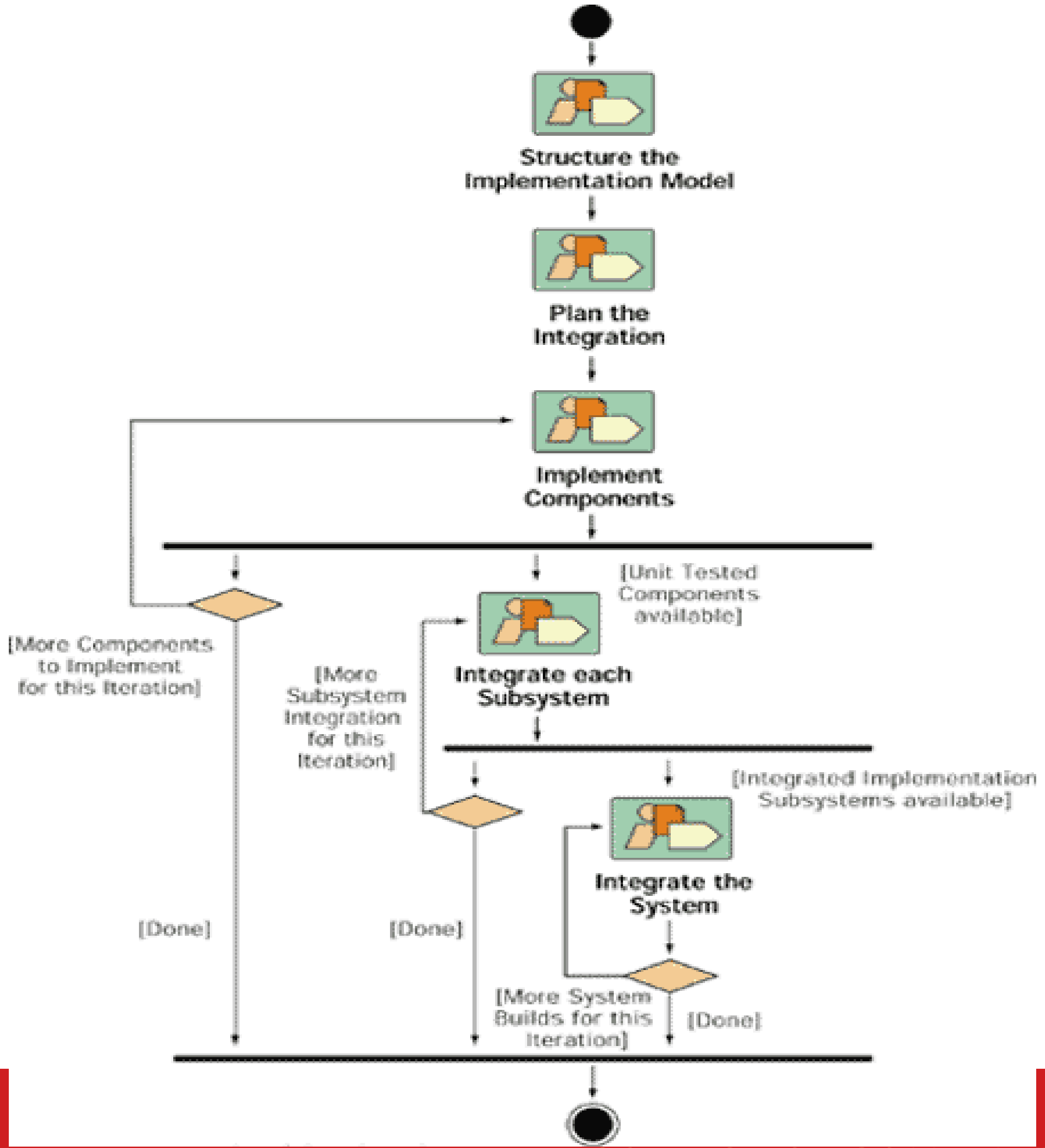
Analysis and Design


[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Refine the Architecture
[Optional]
Design Components
Design the Database

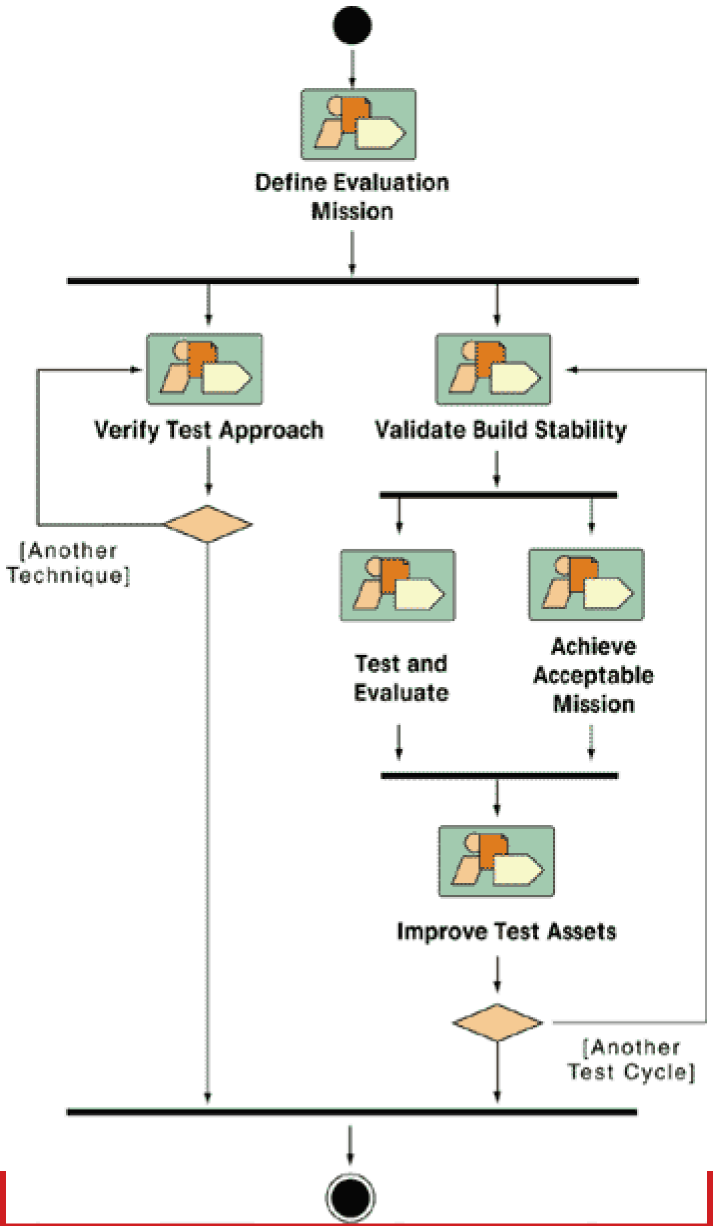
Environment


[Inception Iterations]
Prepare Environment for Project
Prepare Environment for an Iteration
Prepare Guidelines for an Iteration
Support Environment During an Iteration

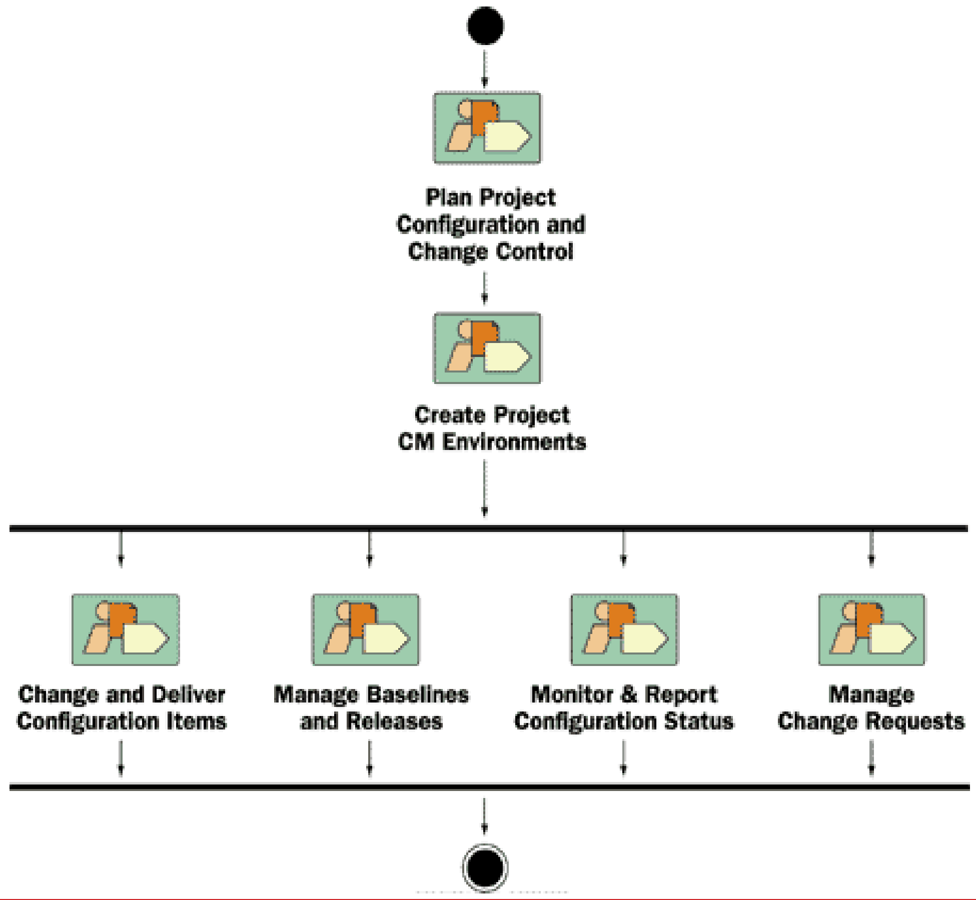
Implementation


Structure the Implementation Model
Plan the Integration
Implement Components
[Unit Tested Components available]
[More Components to Implement for this Iteration]
[More Subsystem Integration for this Iteration]
Integrate each Subsystem
[Integrated Implementation Subsystems available]
Integrate the System
[Done]
[Done]
[More System Builds for this Iteration]
[Done]

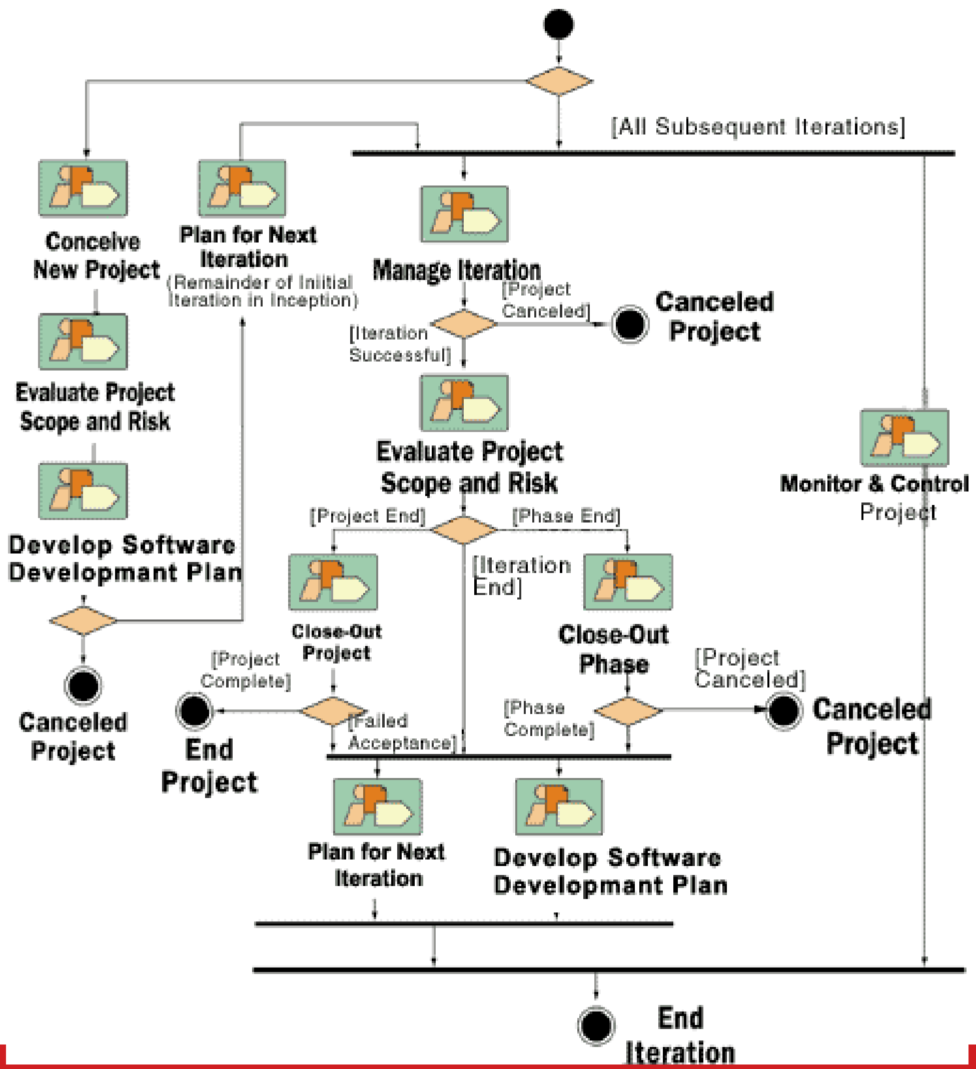
Test


Define Evaluation Mission
Verify Test Approach
Validate Build Stability
[Another Technique]
Test and Evaluate
Achieve Acceptable Mission
Improve Test Assets
[Another Test Cycle]

Configuration and change management


Plan Project Configuration and Change Control
Create Project CM Environments
Change and Deliver Configuration Items
Manage Baselines and Releases
Monitor & Report Configuration Status
Manage Change Requests

Project Management


[All Subsequent Iterations]
Conceive New Project
Plan for Next Iteration
(Remainder of Initial Iteration in Inception)
Manage Iteration
[Project Canceled]
Canceled Project
[Iteration Successful]
Evaluate Project Scope and Risk
Develop Software Developmant Plan
Evaluate Project Scope and Risk
Monitor & Control Project
[Project End]
[Phase End]
[Iteration End]
Close-Out Project
Close-Out Phase
[Project Complete]
[Project Canceled]
Canceled Project
End Project
[Failed Acceptance]
[Phase Complete]
Canceled Project
Plan for Next Iteration
Develop Software Developmant Plan
End Iteration