RATIONAL UNIFIED PROCESS

Fuente:
Rational Unified Process (Patricio Letelier, UPV-España),
Descripción actividades RUP

CONCEPTOS PRELIMINARES

Proceso

"Defines Who is doing What, When to do it and How to reach a certain goal" Ivar Jacobson (Rational SW)

Método

Proceso + Notación + Heurísticas

Modelo

"A model is an abstraction that describes one or more aspects of a problem or a potential solution addressing a problem" *Scott Ambler (AmbySoft)*

RUP

Rational Unified Process (RUP) es un marco de referencia de procesos

- Influenciado por patrones de Proceso/Análisis
- Bien documentado

RUP es una enorme base de conocimiento en Ingeniería de Software (RUP Homepage (uhcl.edu))

Desarrollado por Rational (IBM)

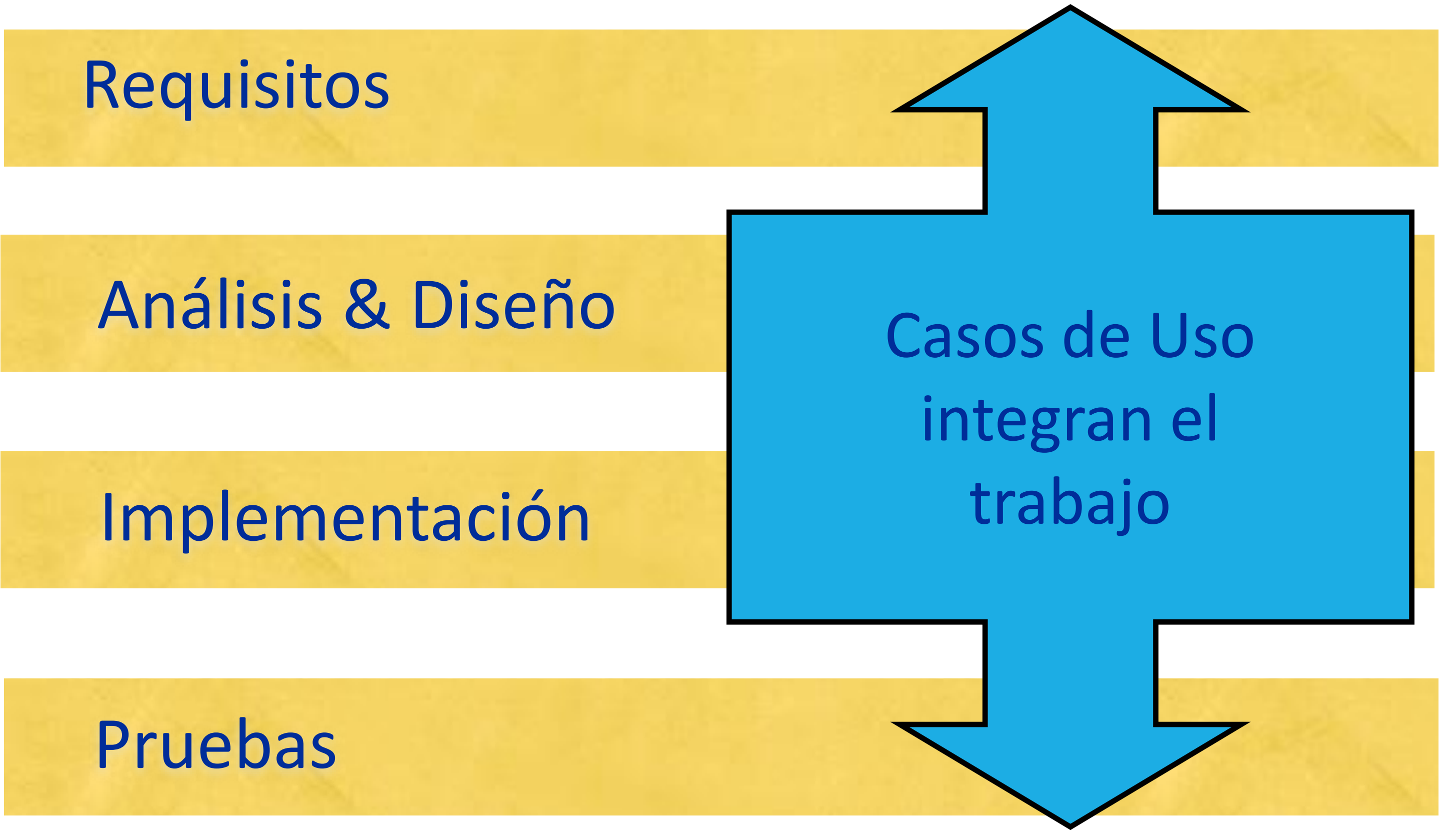
CARACTERÍSTICAS ESENCIALES DE RUP

Proceso Dirigido por los Casos de Uso

Proceso Centrado en la Arquitectura

Proceso Iterativo e Incremental

1. PROCESO DIRIGIDO POR CASOS DE USO

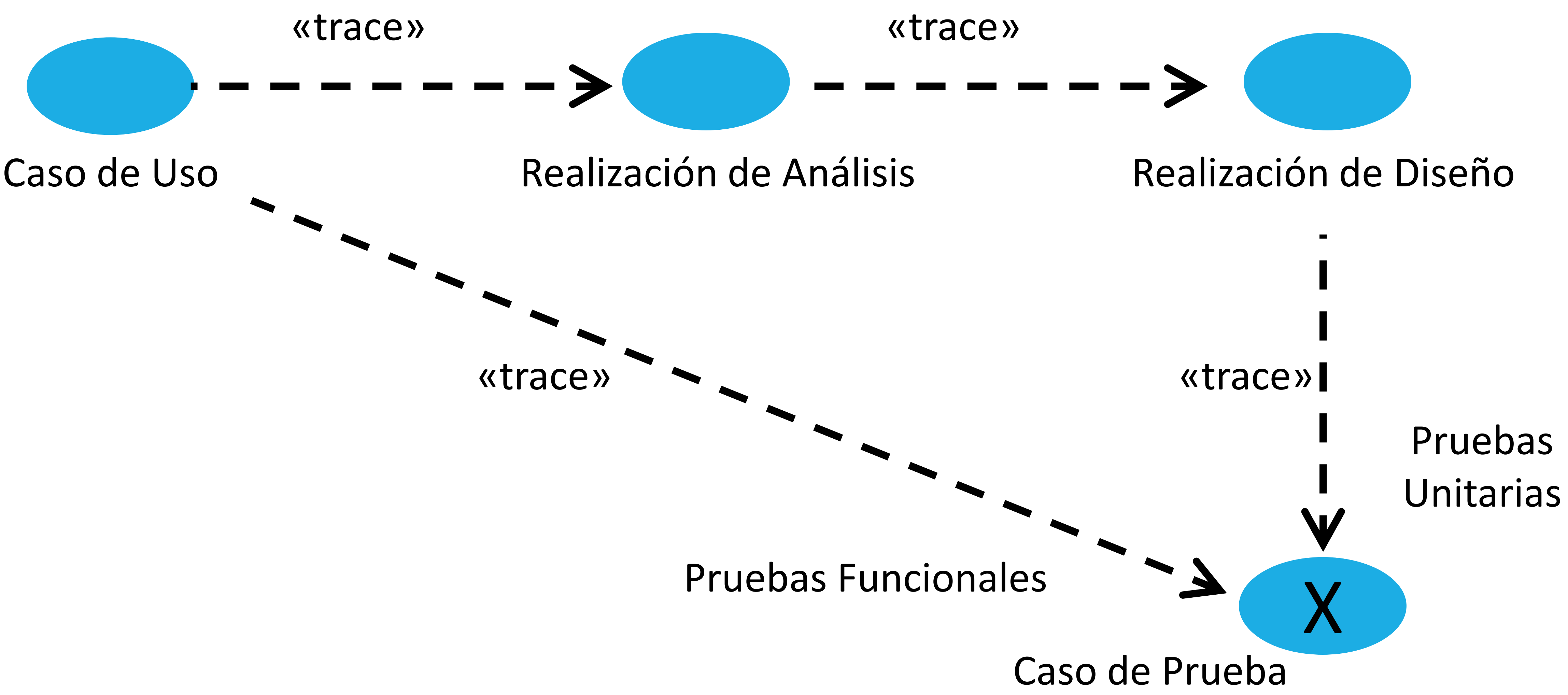


Capturar, definir y validar los casos de uso

Realizar los casos de uso

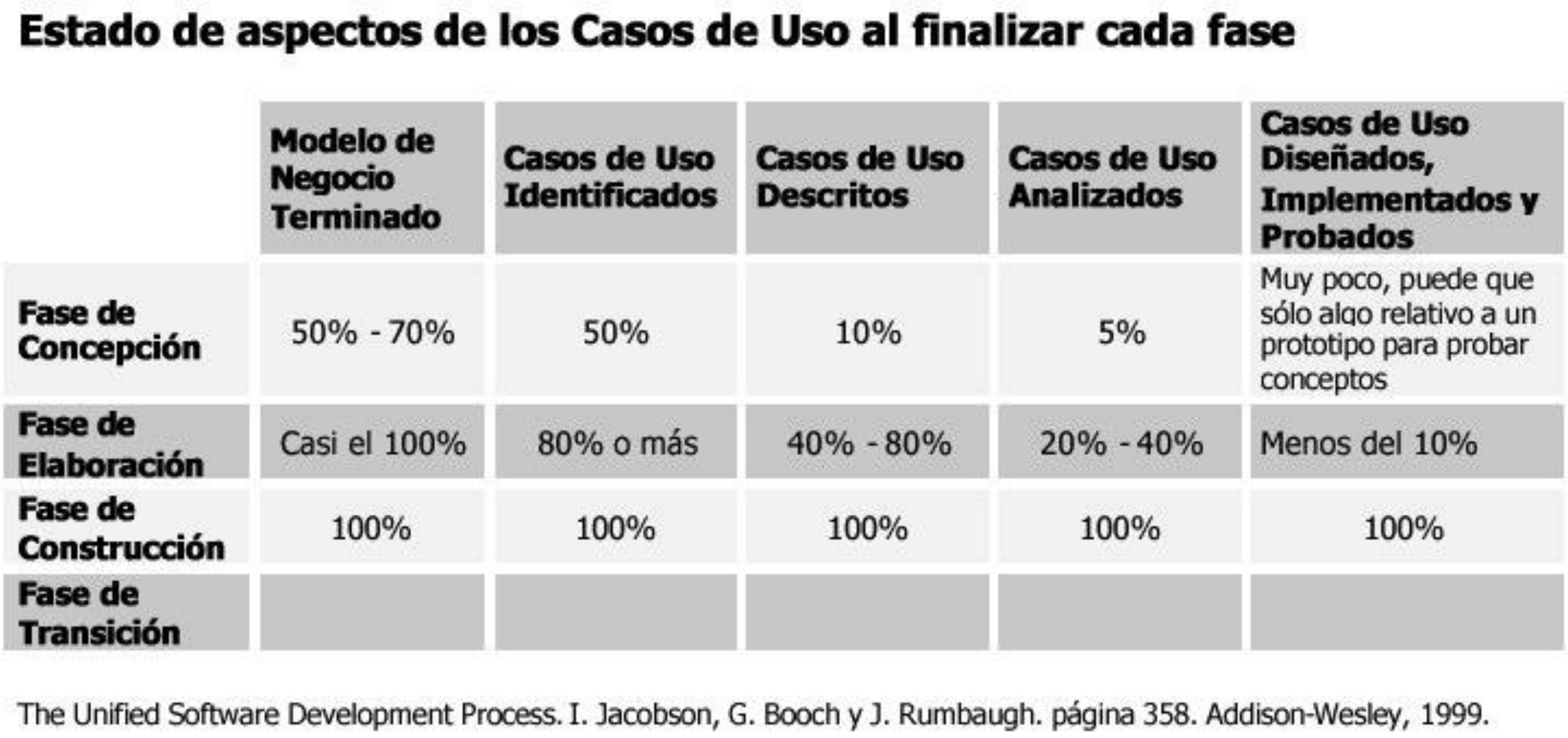
Verificar que se satisfacen los casos de uso

1. PROCESO DIRIGIDO POR CASOS DE USO



[The Unified Software Development Process. I. Jacobson, G. Booch and J. Rumbaugh. Addison-Wesley, 1999]

1. PROCESO DIRIGIDO POR CASOS DE USO



Estado de aspectos de los Casos de Uso al finalizar cada fase

	Modelo de Negocio Terminado	Casos de Uso Identificados	Casos de Uso Descritos	Casos de Uso Analizados	Casos de Uso Diseñados, Implementados y Probados
Fase de Concepción	50% - 70%	50%	10%	5%	Muy poco, puede que sólo algo relativo a un prototipo para probar conceptos
Fase de Elaboración	Casi el 100%	80% o más	40% - 80%	20% - 40%	Menos del 10%
Fase de Construcción	100%	100%	100%	100%	100%
Fase de Transición					

The Unified Software Development Process. I. Jacobson, G. Booch y J. Rumbaugh. página 358. Addison-Wesley, 1999.

2. PROCESO CENTRADO EN LA ARQUITECTURA

"The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them".

- *Bass et al. SW Architecture in Practice (2003)*

2. PROCESO CENTRADO EN LA ARQUITECTURA

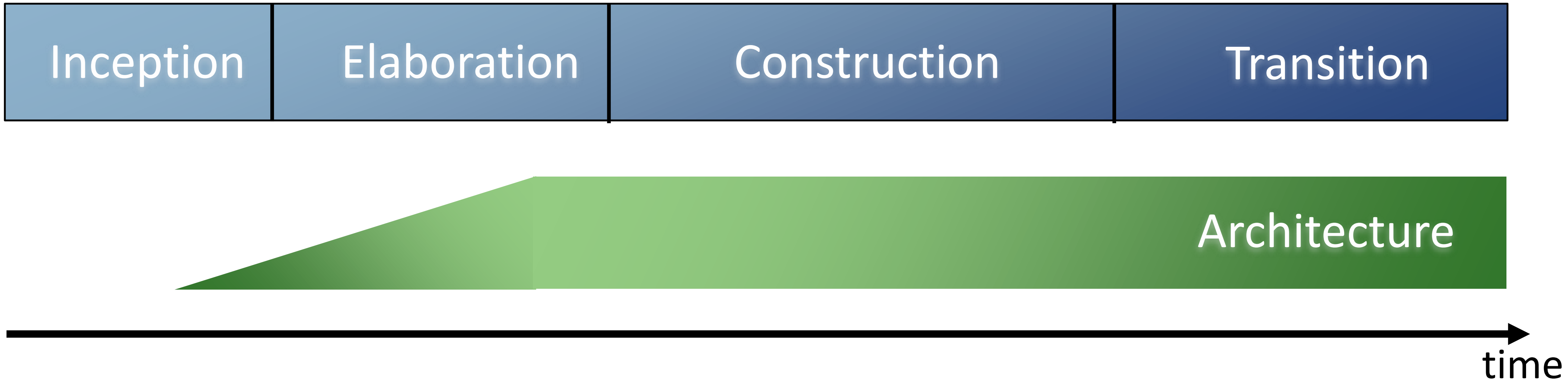
Arquitectura de un sistema es la organización o estructura de sus partes más relevantes

- Producto: Función y Forma
- Función: Casos de Uso; Forma: Arquitectura

Un arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades

2. PROCESO CENTRADO EN LA ARQUITECTURA

RUP establece refinamientos sucesivos de una arquitectura ejecutable, construida como un prototipo evolutivo



Modificación y Consolidación de la Arquitectura

3. PROCESO ITERATIVO E INCREMENTAL

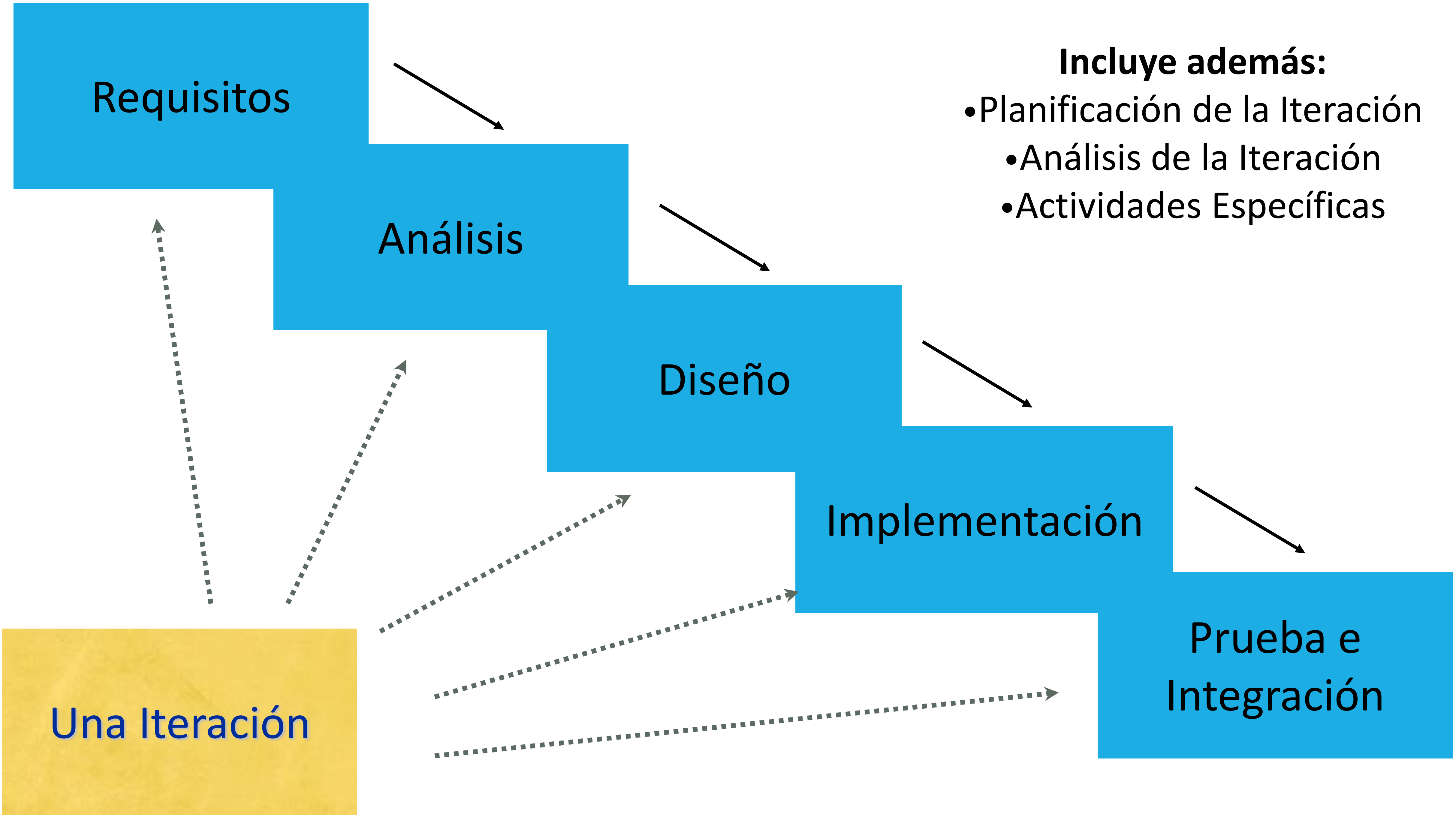
El trabajo se divide en partes más pequeñas o mini proyectos

Cada mini proyecto es una iteración que produce un crecimiento en el producto

En cada iteración se determina si han aparecido nuevos requerimientos o se han modificado los existentes

Se captura la retroalimentación de iteraciones anteriores

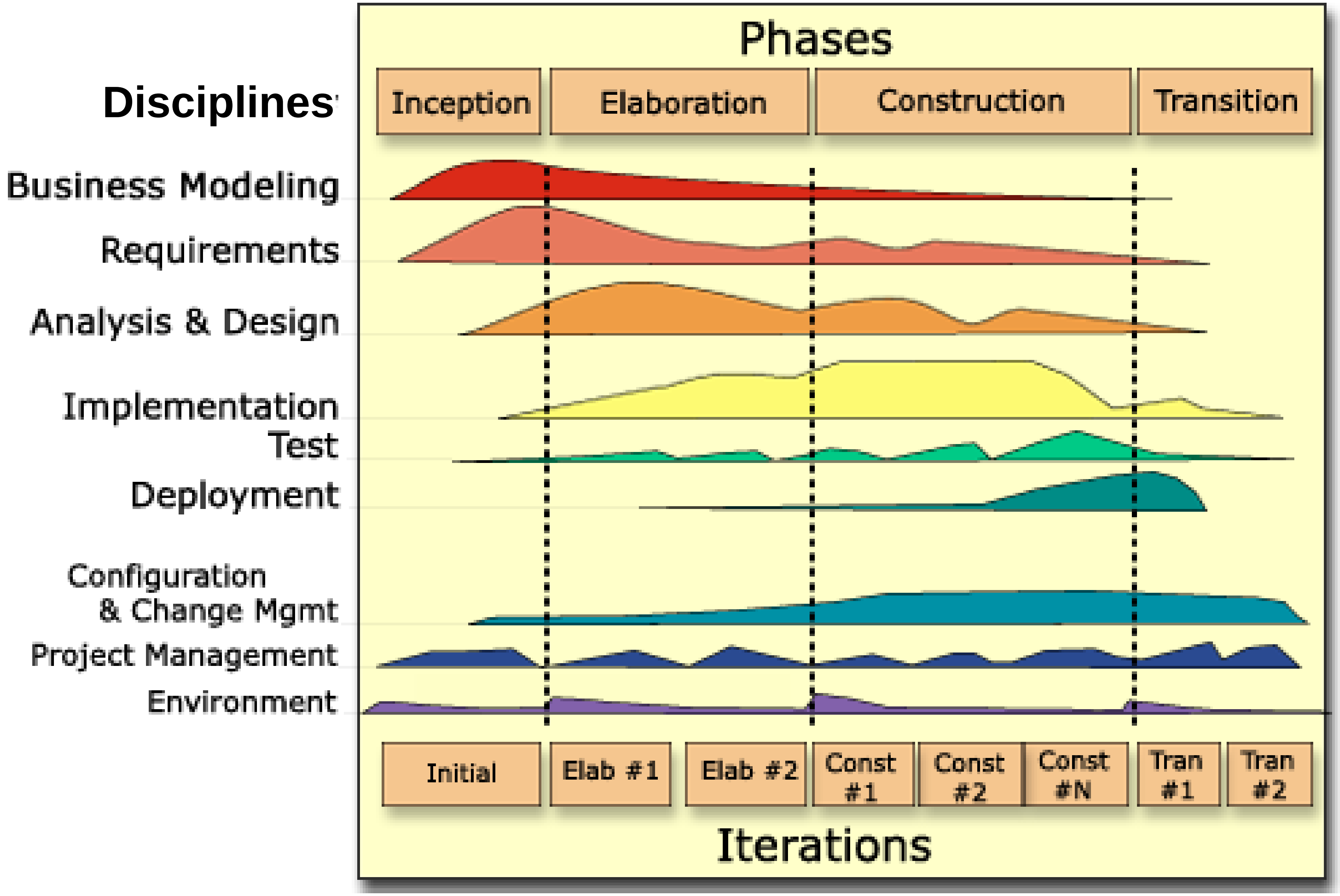
3. PROCESO ITERATIVO E INCREMENTAL



Incluye además:

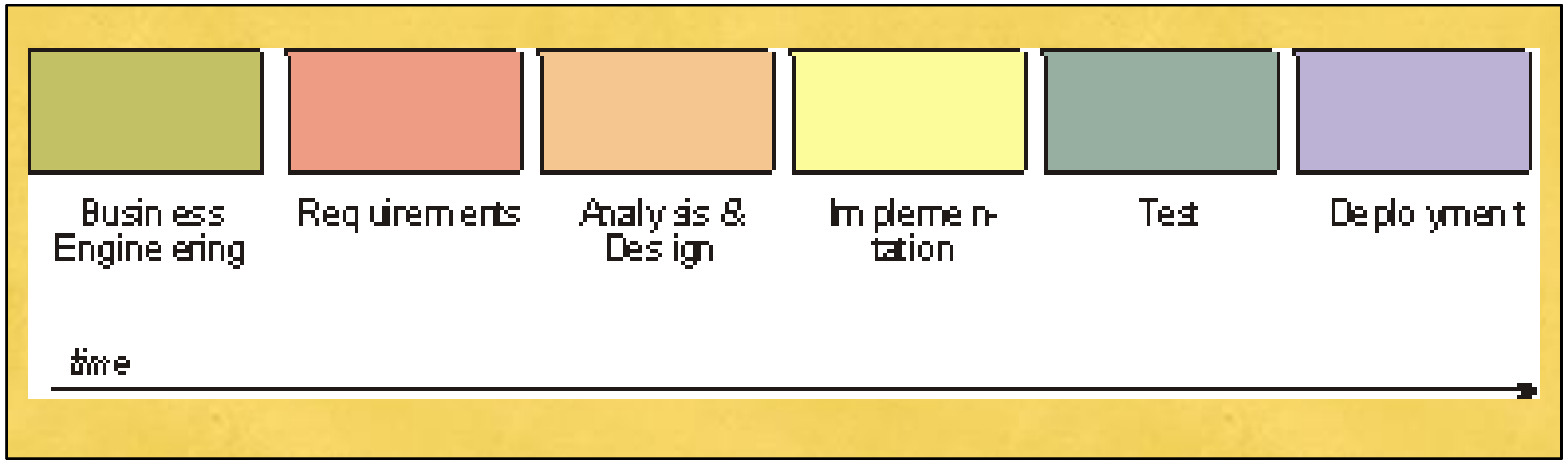
- Planificación de la Iteración
- Análisis de la Iteración
- Actividades Específicas

ESFUERZO DE ACTIVIDADES SEGÚN FASE DEL PROYECTO

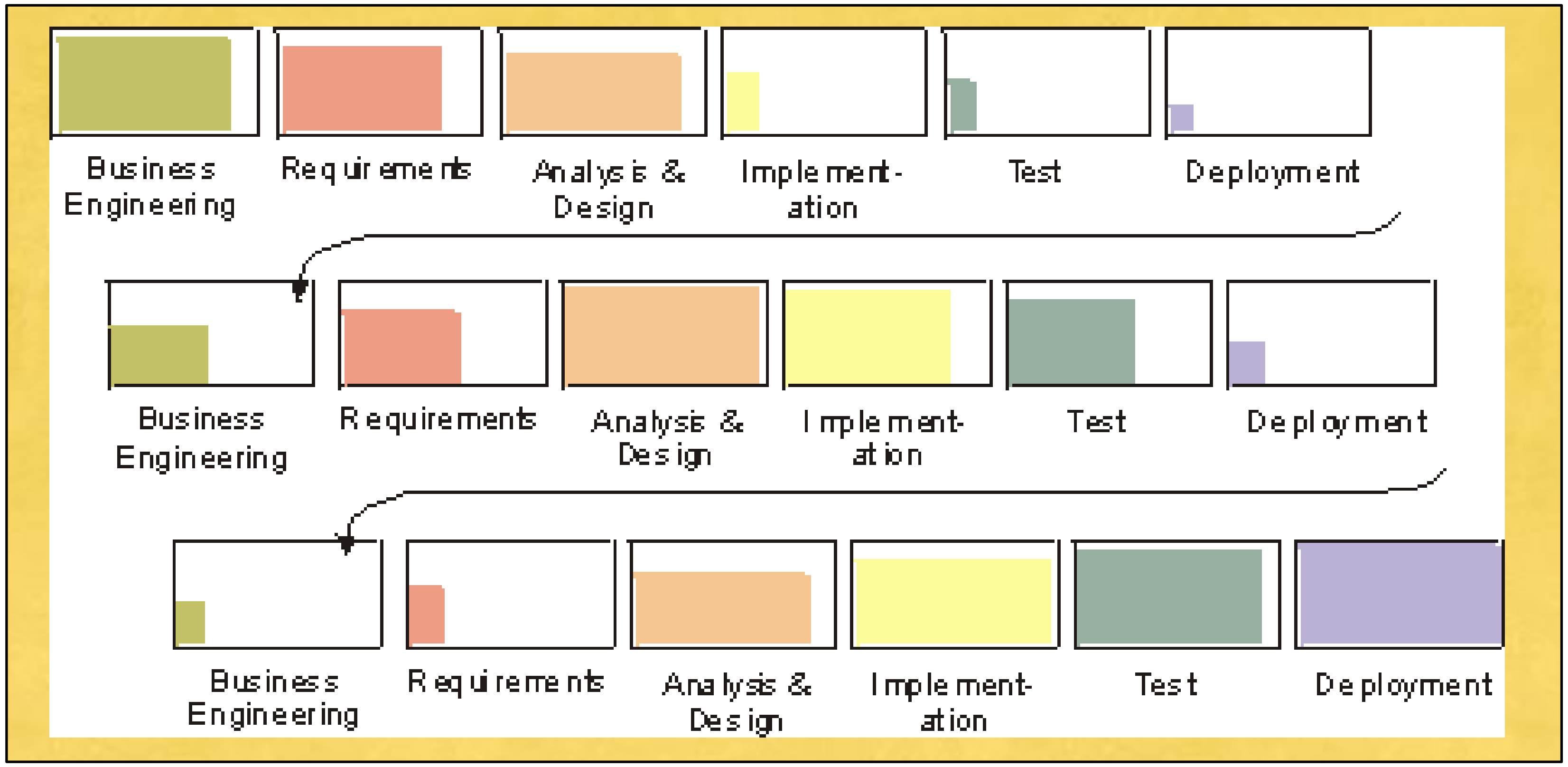


PROCESO ITERATIVO E INCREMENTAL

Enfoque Secuencial



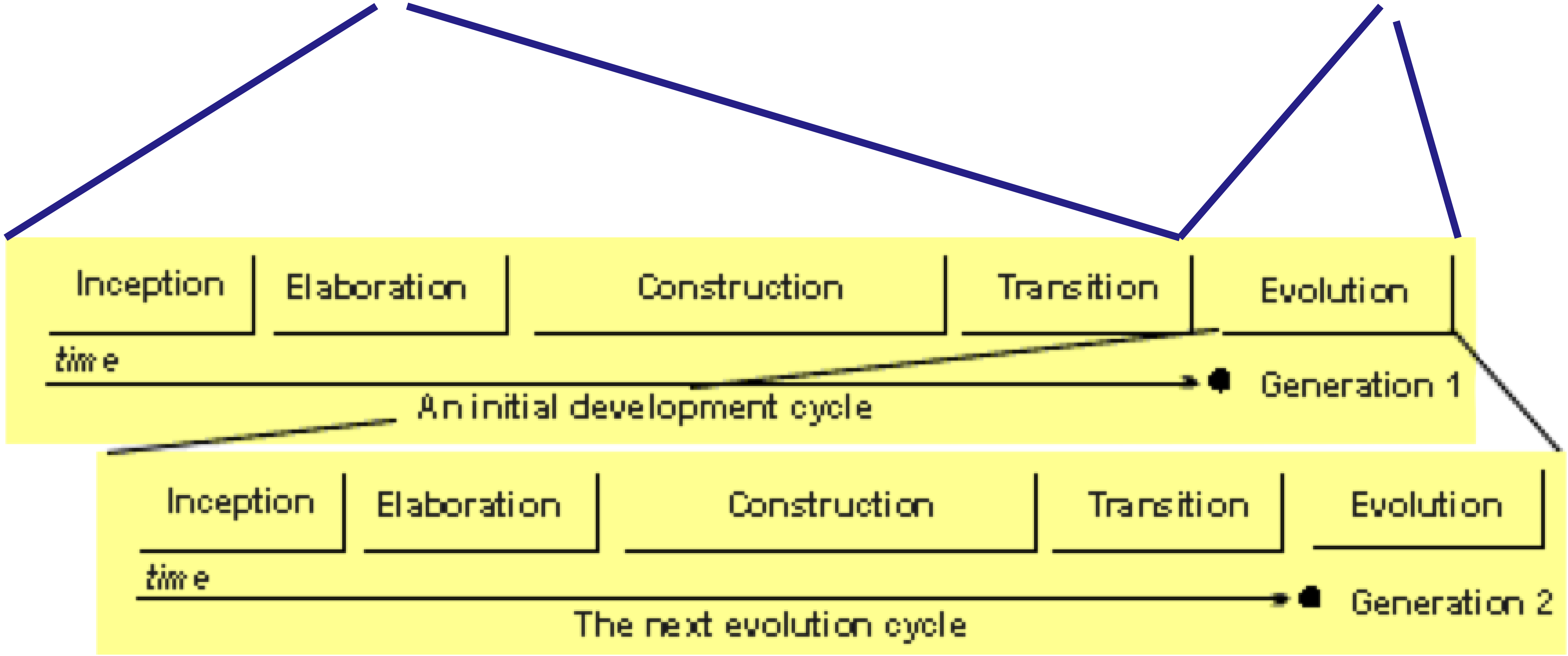
Enfoque Iterativo e Incremental



FASES, RELEASE, BASE LINE, GENERACIÓN

ciclo de desarrollo

ciclo de evolución



release
(producto al final de
una iteración)

baseline
(release asociada
a un hito)

generación
(release final de
un ciclo de desarrollo)

RUP GOOD PRACTICE

Develop software iteratively

- Plan increments based on customer priorities and deliver highest priority increments first.

Manage requirements

- Explicitly document customer requirements and keep track of changes to these requirements.

Use component-based architectures

- Organize the system architecture as a set of reusable components.

RUP GOOD PRACTICE

Visually model software

- Use graphical UML models to present static and dynamic views of the software.

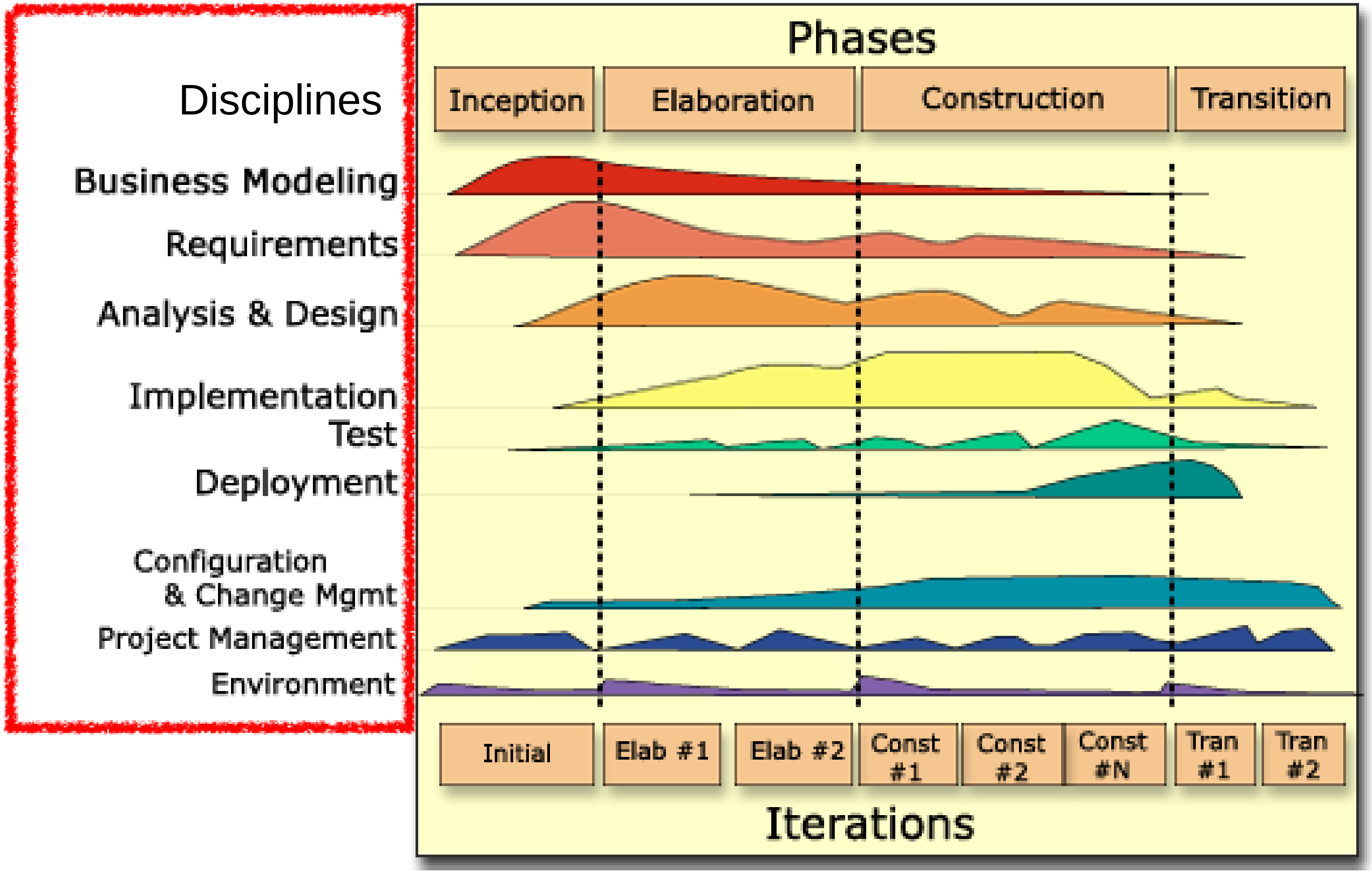
Verify software quality

- Ensure that the software meet's organizational quality standards.

Control changes to software

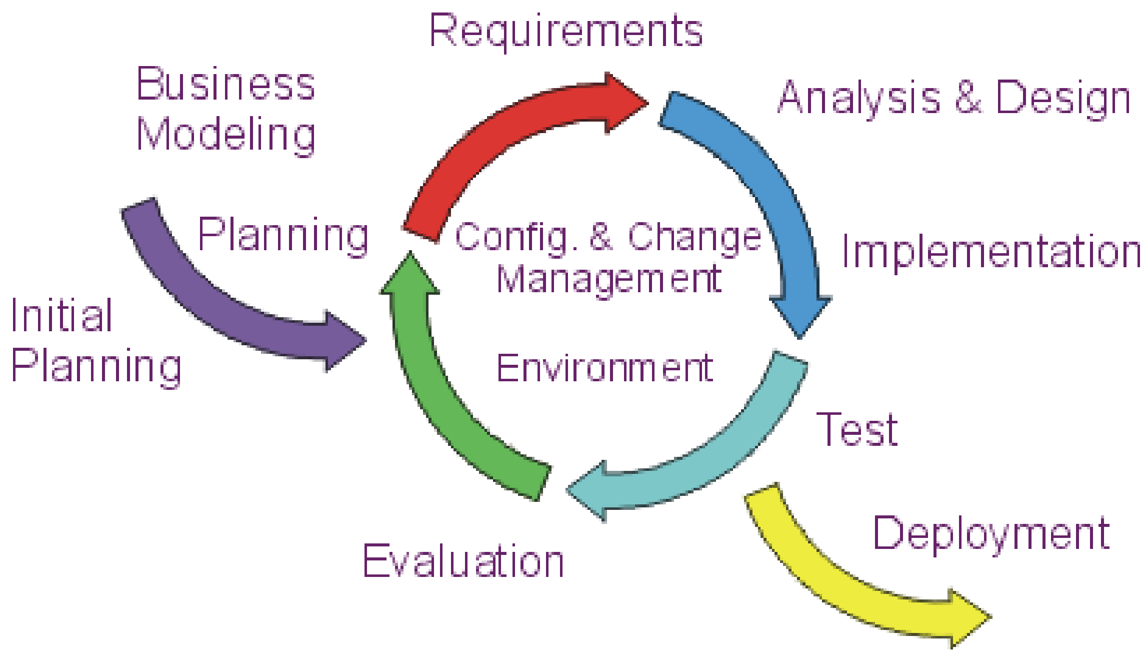
- Manage software changes using a change management system and configuration management tools.

RUP DISCIPLINES



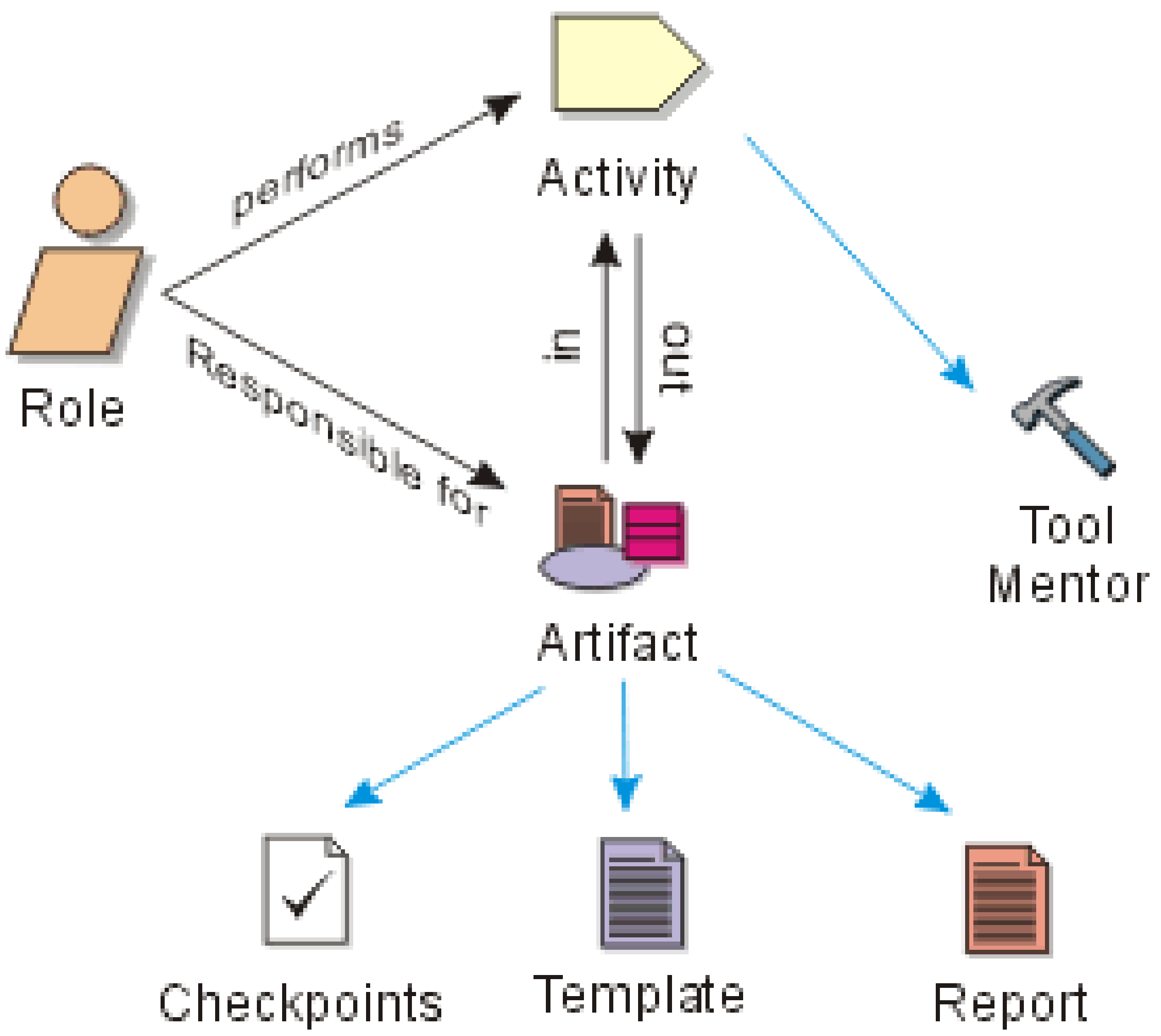
WHAT IS A DISCIPLINE?

- A set of activities that is performed by the various roles in a project
- Describes a meaningful sequence of activities that produce a useful result (an artifact)
- Shows interaction between roles



DISCIPLINES - 3 KEY ELEMENTS

- Artifacts
- Roles
- Activities



MAIN DISCIPLINES

Business Modelling

Requirements

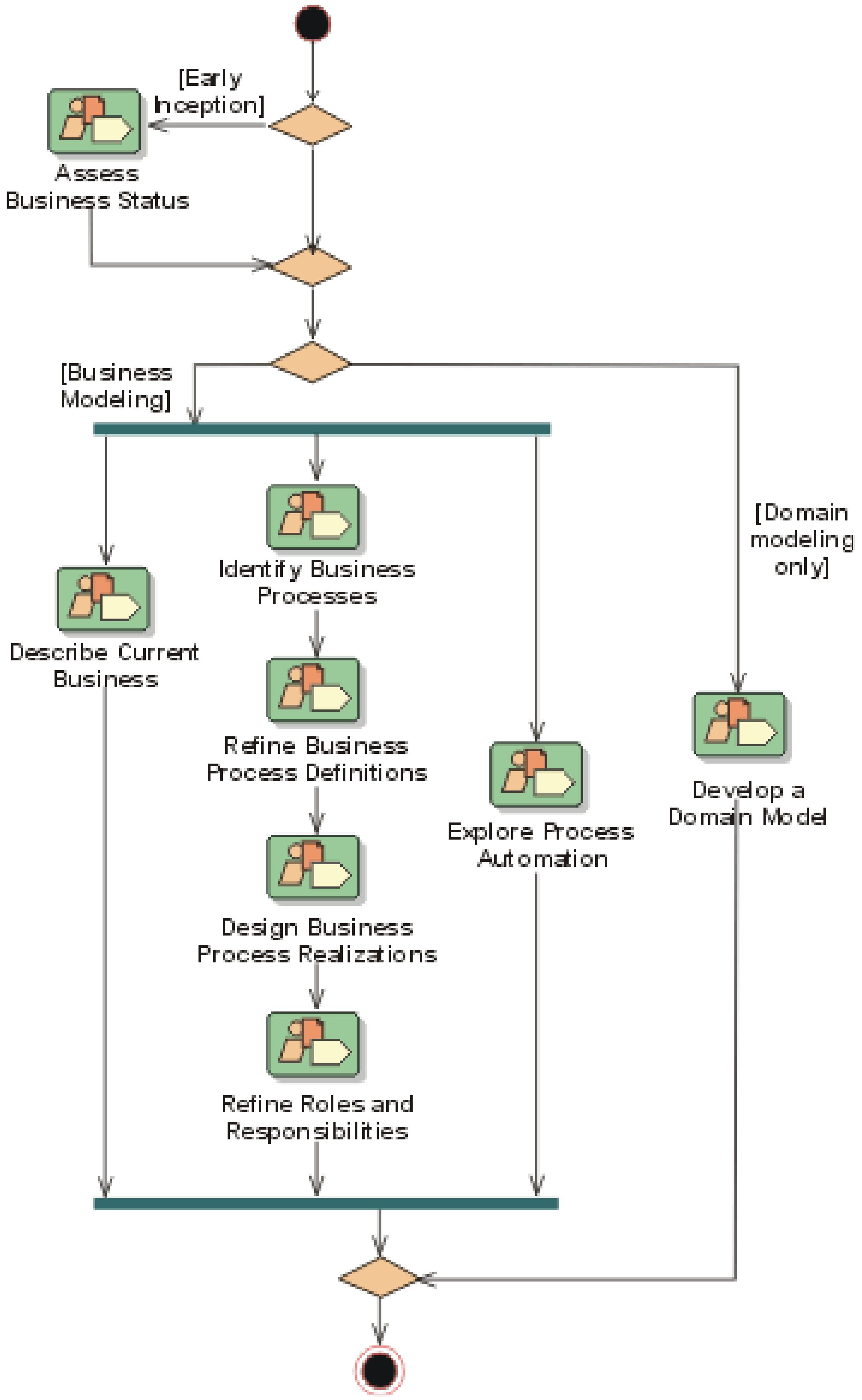
Analysis & Design

Implementation

Test

Deployment

BUSINESS MODELLING


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

BUSINESS MODELLING GOALS

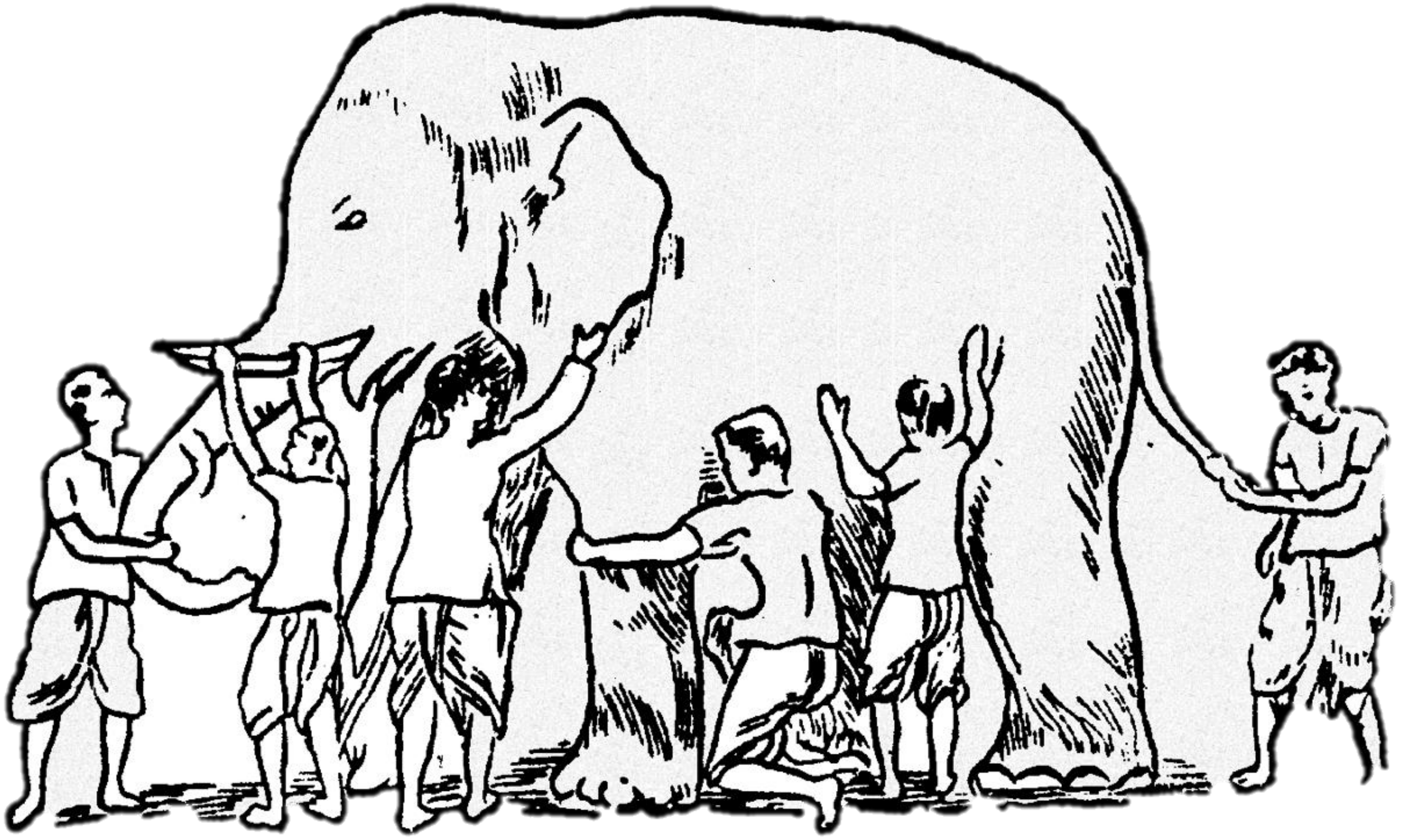
To understand current problems in the target organization and identify improvements potentials

BUSINESS MODELLING GOALS

To assess the impact of organizational change

BUSINESS MODELLING GOALS

To ensure that customers, end users, developers, and other parties have a common understanding of the organization

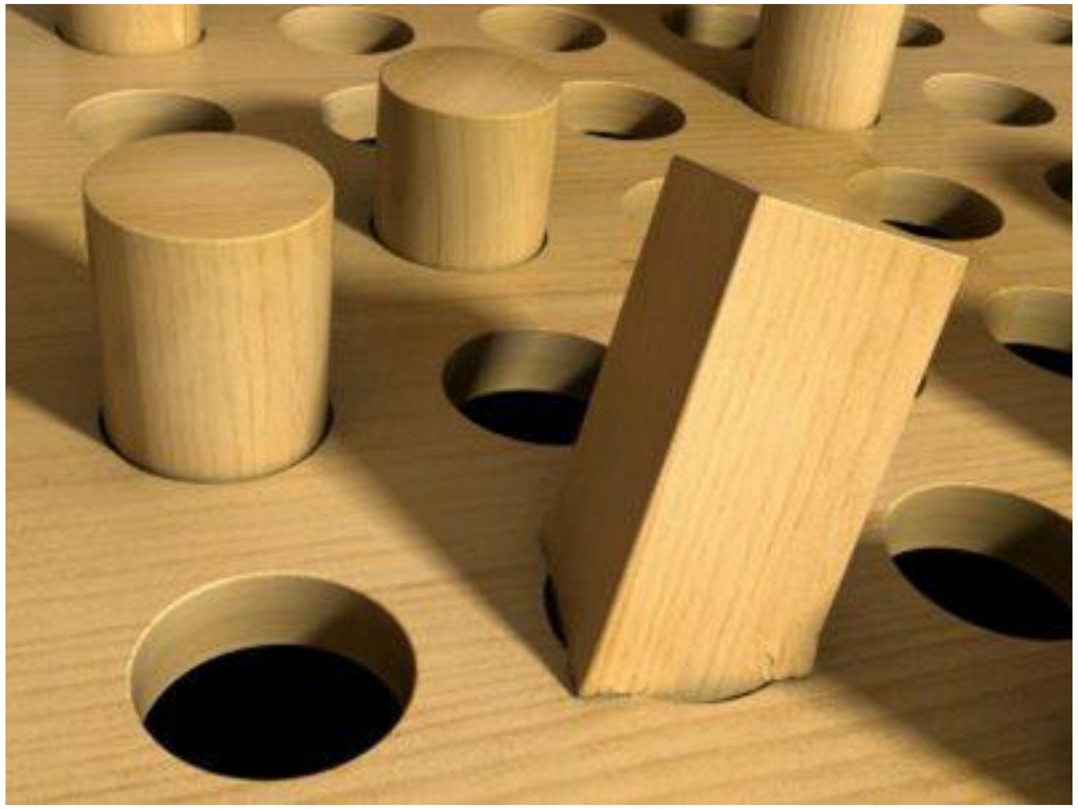


BUSINESS MODELLING GOALS

To derive the software system requirements needed to support the target organization

BUSINESS MODELLING GOALS

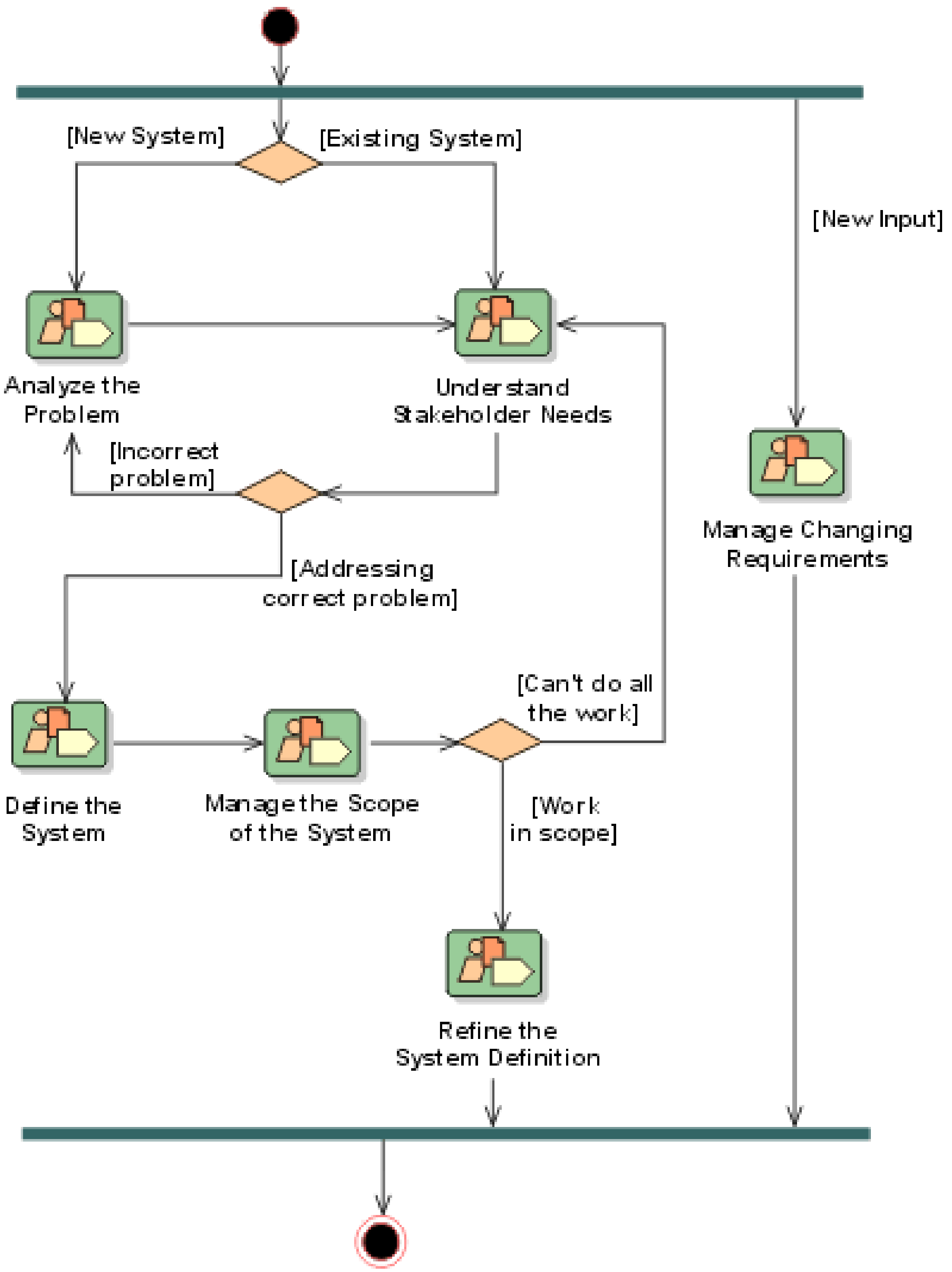
To understand how a to-be-deployed software system fits into the organization



BUSINESS MODELLING ARTIFACTS

- Business Vision
- Business Architecture Document
- Business Rules
- Business Glossary

REQUIREMENTS


[New System]
[Existing System]
[New Input]
Analyze the Problem
Understand Stakeholder Needs
Manage Changing Requirements
[Incorrect problem]
[Addressing correct problem]
[Can't do all the work]
Define the System
Manage the Scope of the System
[Work in scope]
Refine the System Definition

REQUIREMENTS

To establish and maintain agreement with the customers and other stakeholders on what the system should do.

To provide system developers with a better understanding of the system requirements.

To define the boundaries of (delimit) the system.

To provide a basis for planning the technical contents of iterations.

To provide a basis for estimating cost and time to develop the system.

To define a user-interface for the system, focusing on the needs and goals of the users.

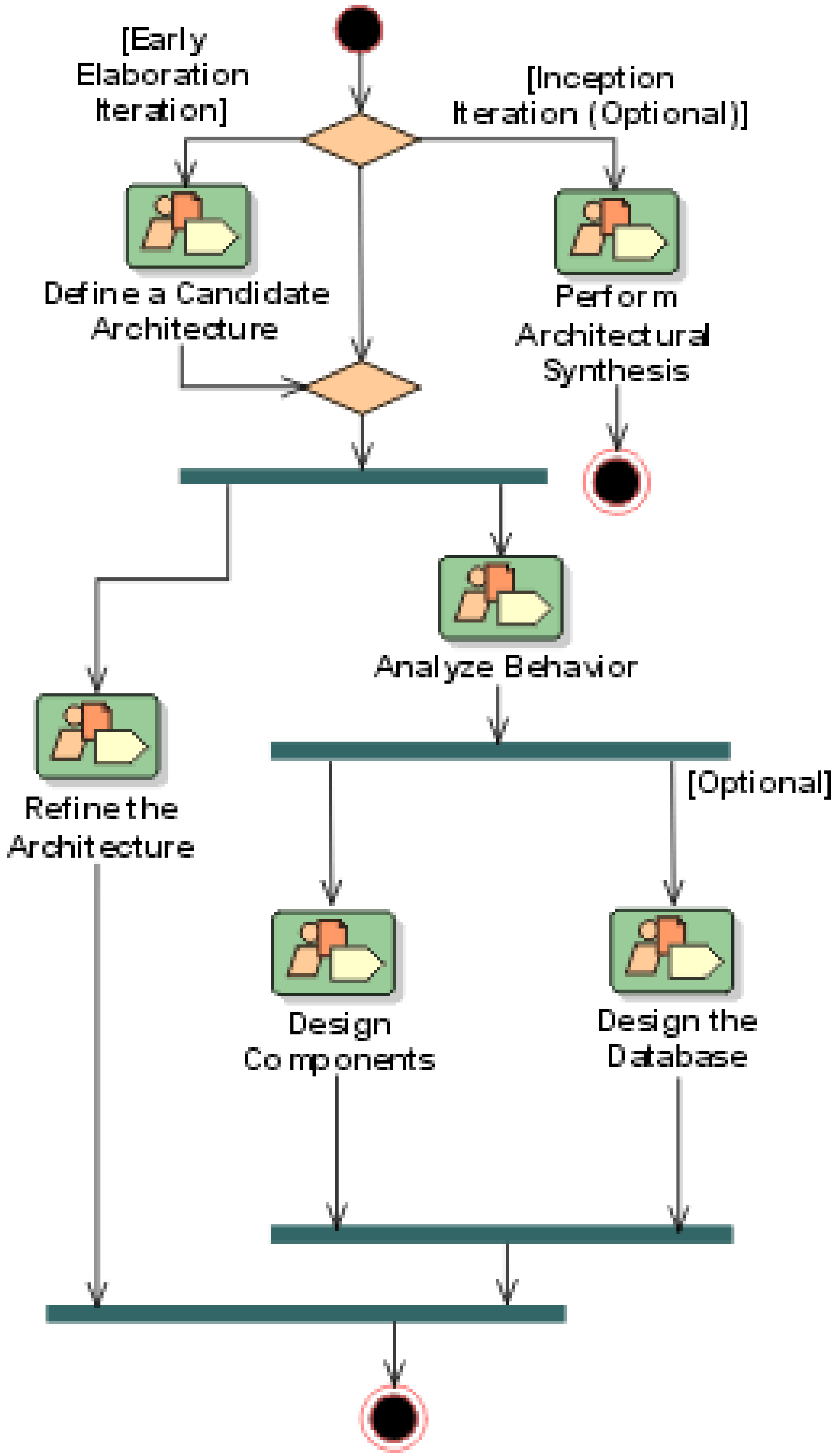
REQUIREMENTS: ARTIFACTS

- Glossary
- Use-case model
- Vision document
- User-interface prototype
- Business Glossary

DOCUMENTO DE VISIÓN

1. Introducción
2. Posicionamiento
    1. Descripción del Problema
    2. Descripción del Producto
3. Stakeholder y Usuarios
    1. Resumen de Stakeholders
    2. Resumen de Usuarios
    3. Necesidades clave de cada uno
4. Descripción de Producto
    1. Características del Producto
    2. Otros requerimientos

ANALYSIS & 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

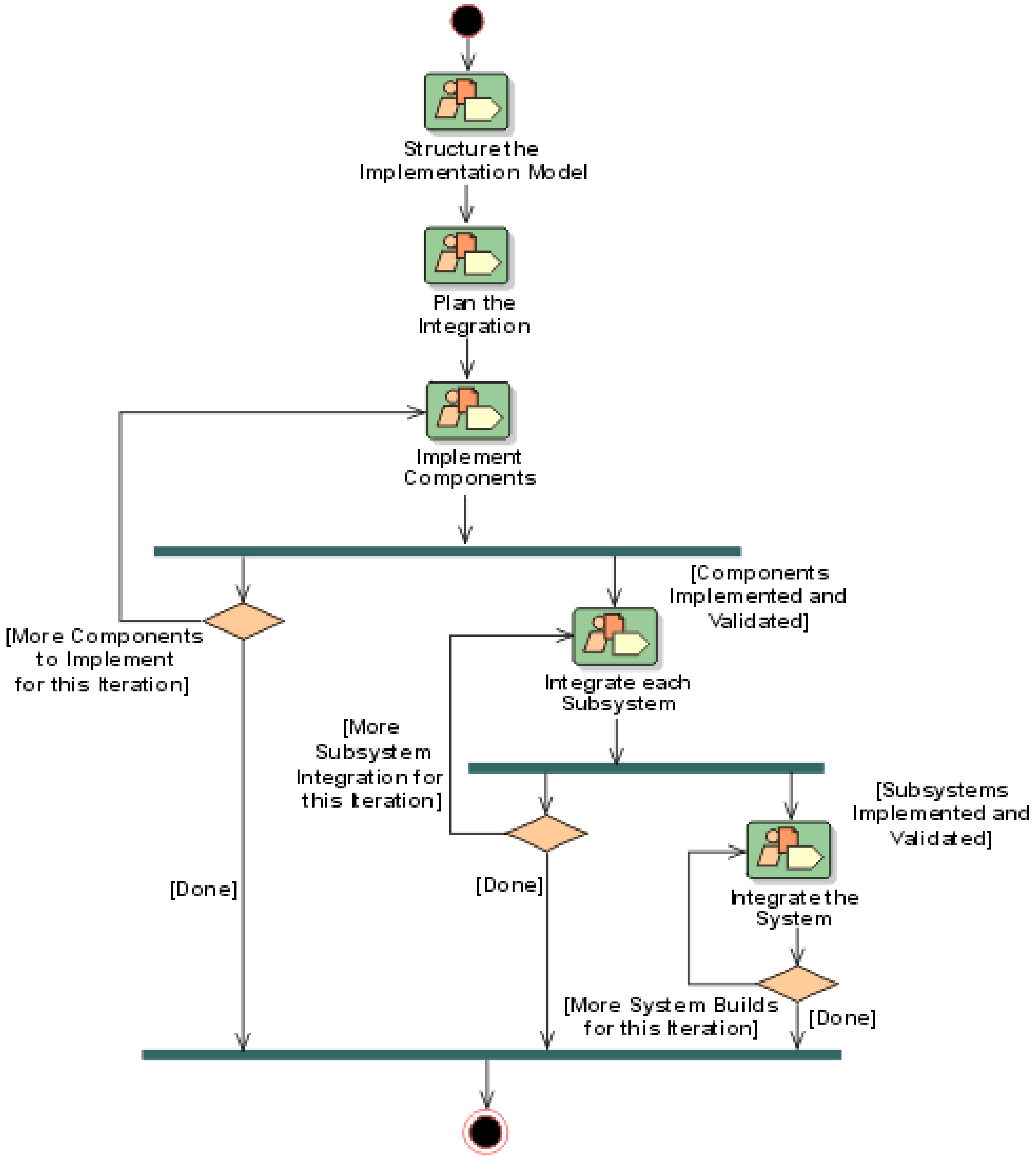
ANALYSIS AND DESIGN

- Transform requirements into a design of the system
- Evolve a robust architecture for the system
- Adapt design to match the implementation environment, designing it for performance

ANALYSIS AND DESIGN: ARTIFACTS

- Software architecture document
- Design model
- Data model

IMPLEMENTATION


Structure the Implementation Model
Plan the Integration
Implement Components
[More Components to Implement for this Iteration]
[Components Implemented and Validated]
Integrate each Subsystem
[More Subsystem Integration for this Iteration]
[Subsystems Implemented and Validated]
Integrate the System
[Done]
[Done]
[More System Builds for this Iteration]
[Done]

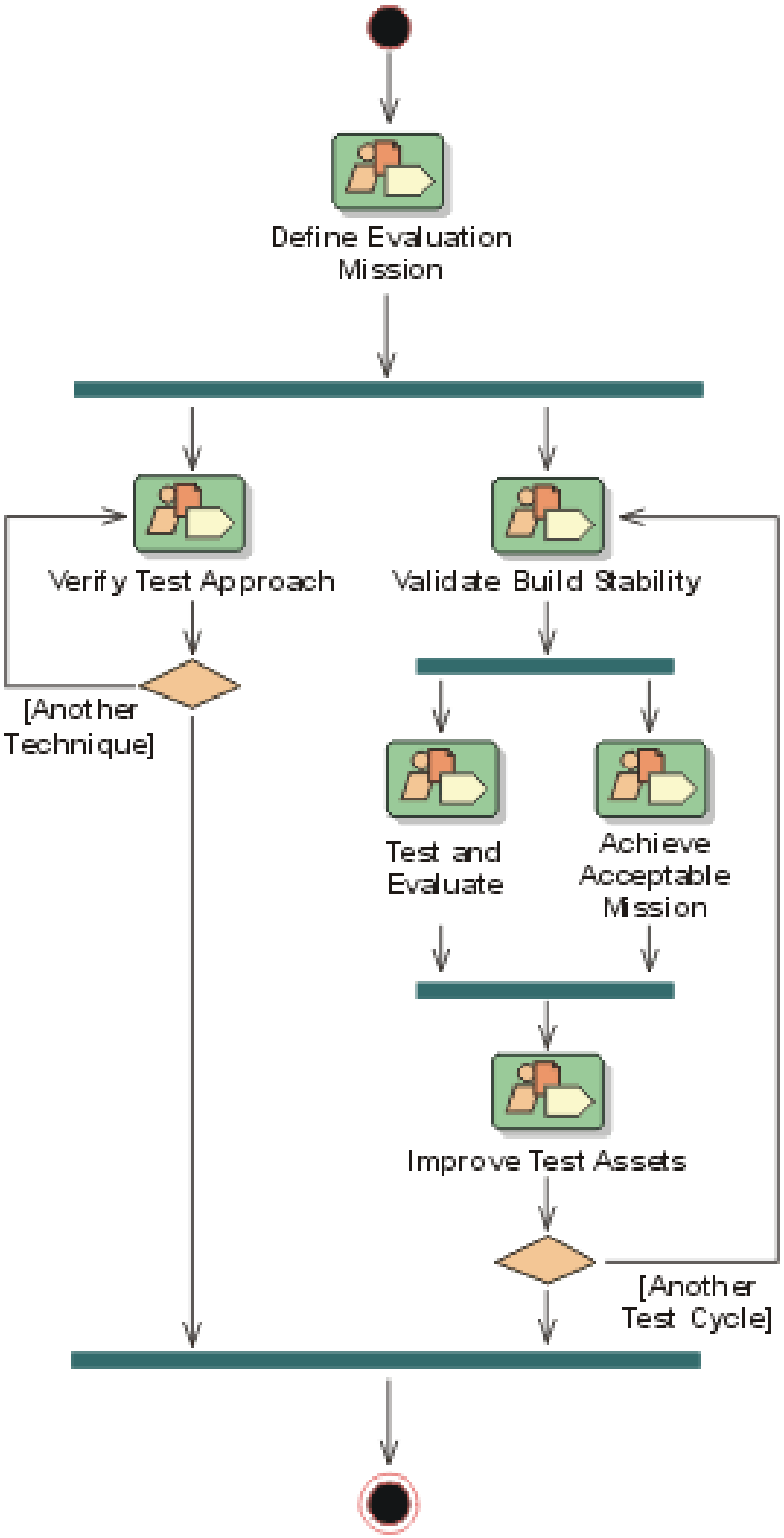
IMPLEMENTATION

- Define organization of the code, in terms of implementation subsystems organized in layers
- Implement classes & objects in terms of components (source files, binaries, executables, and others)
- Test developed components as units
- Integrate results into an executable system

IMPLEMENTATION: ARTIFACTS

- Build
- Implementation model

TEST


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

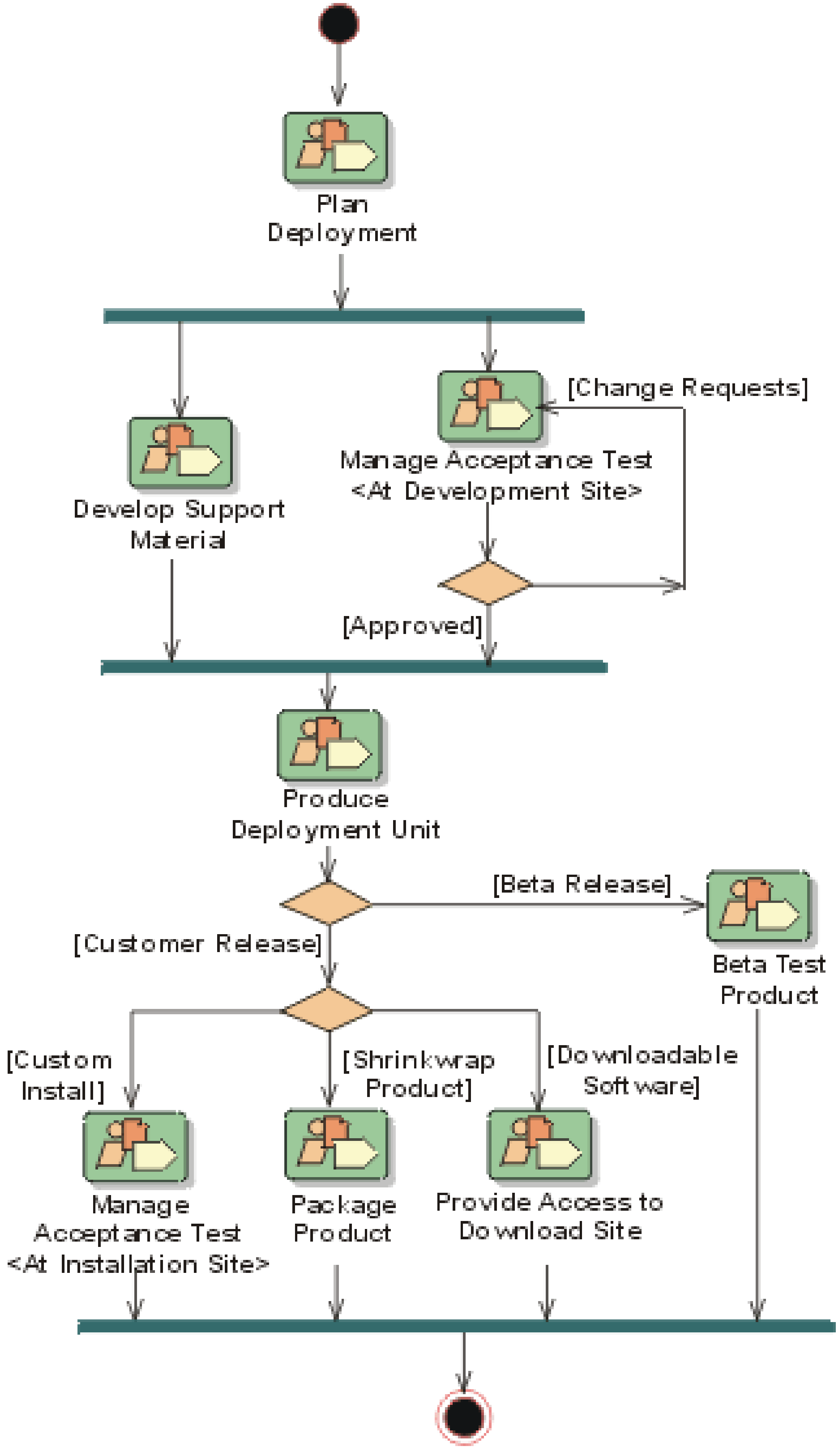
TEST

- Verify interaction between objects
- Verify proper integration of all components of the software
- Verify that all requirements have been correctly implemented
- Identify & ensure defects are addressed prior to deployment

TEST ARTIFACTS

- Test_Ideas List
- Test Plan
- Test Results
- Test Evaluation Summary

DEPLOYMENT


Plan Deployment
Develop Support Material
Manage Acceptance Test <At Development Site>
[Change Requests]
[Approved]
Produce Deployment Unit
[Beta Release]
Beta Test Product
[Customer Release]
[Custom Install]
[Shrinkwrap Product]
[Downloadable Software]
Manage Acceptance Test <At Installation Site>
Package Product
Provide Access to Download Site

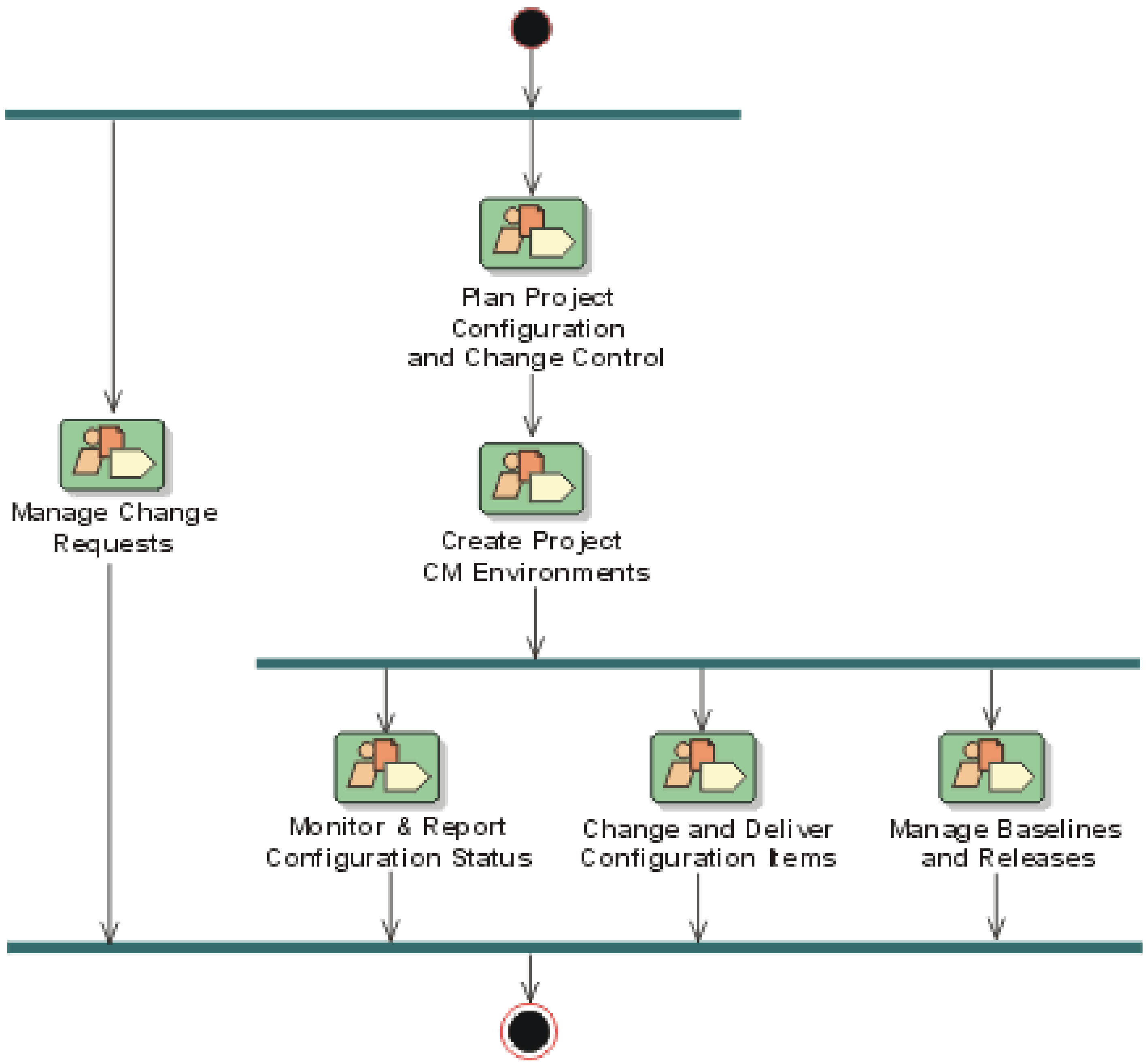
DEPLOYMENT

- Provide custom installation
- Provide shrink wrap product offering
- Provide software over internet

DEPLOYMENT: ARTIFACTS

- Deployment Plan
- End-user support material

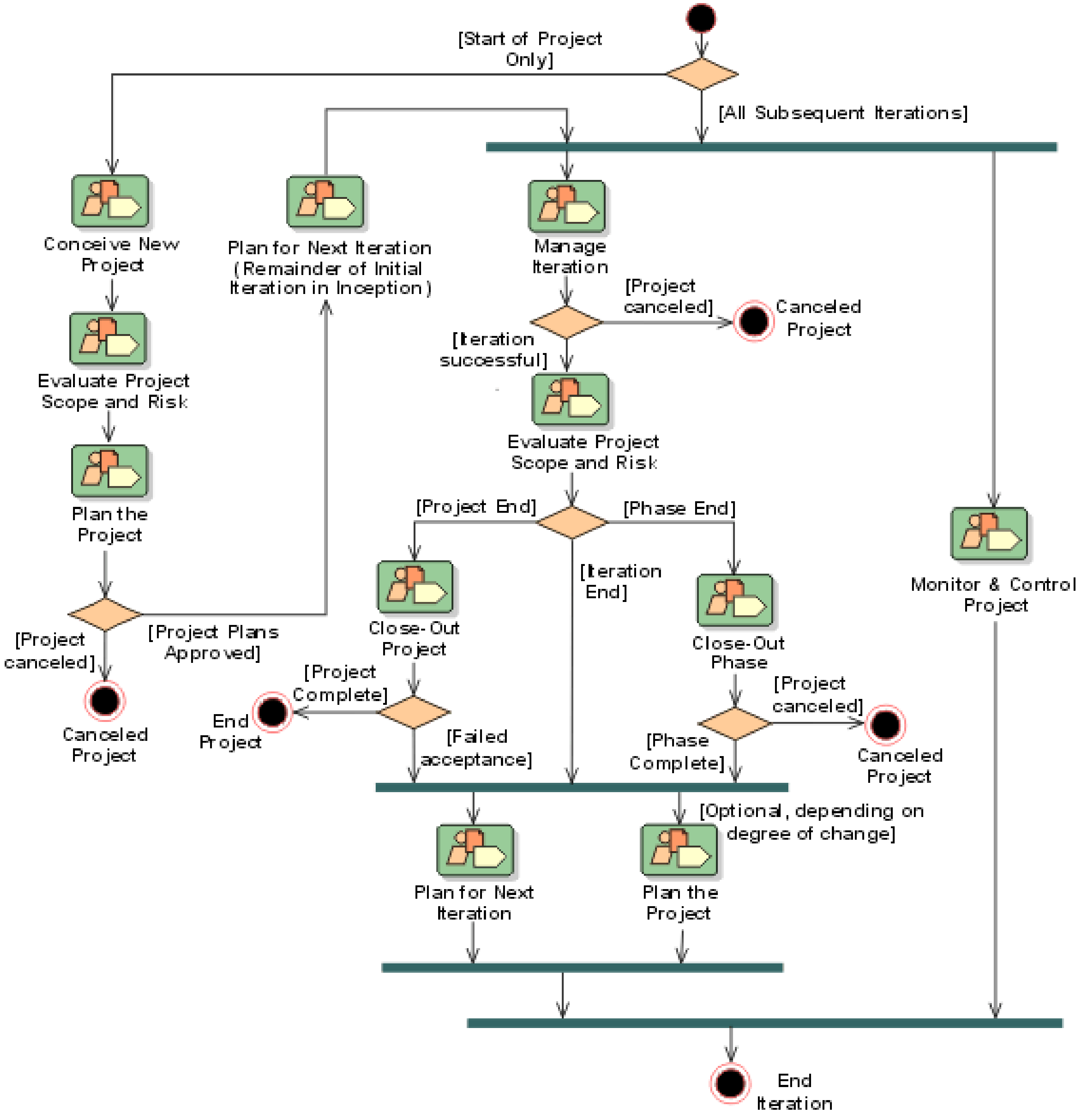
CONFIGURATION AND CHANGE MANAGEMENT


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

CONFIGURATION AND CHANGE MANAGEMENT

- identifying configuration items
- restricting changes to those items
- auditing changes made to those items
- defining and managing configurations of those items.

PROJECT MANAGEMENT



PROJECT MANAGEMENT: GOALS

- To provide a framework for managing software-intensive projects.
- To provide practical guidelines for planning, staffing, executing, and monitoring projects.
- To provide a framework for managing risk.

PROJECT MANAGEMENT: WHAT DOESN'T COVER

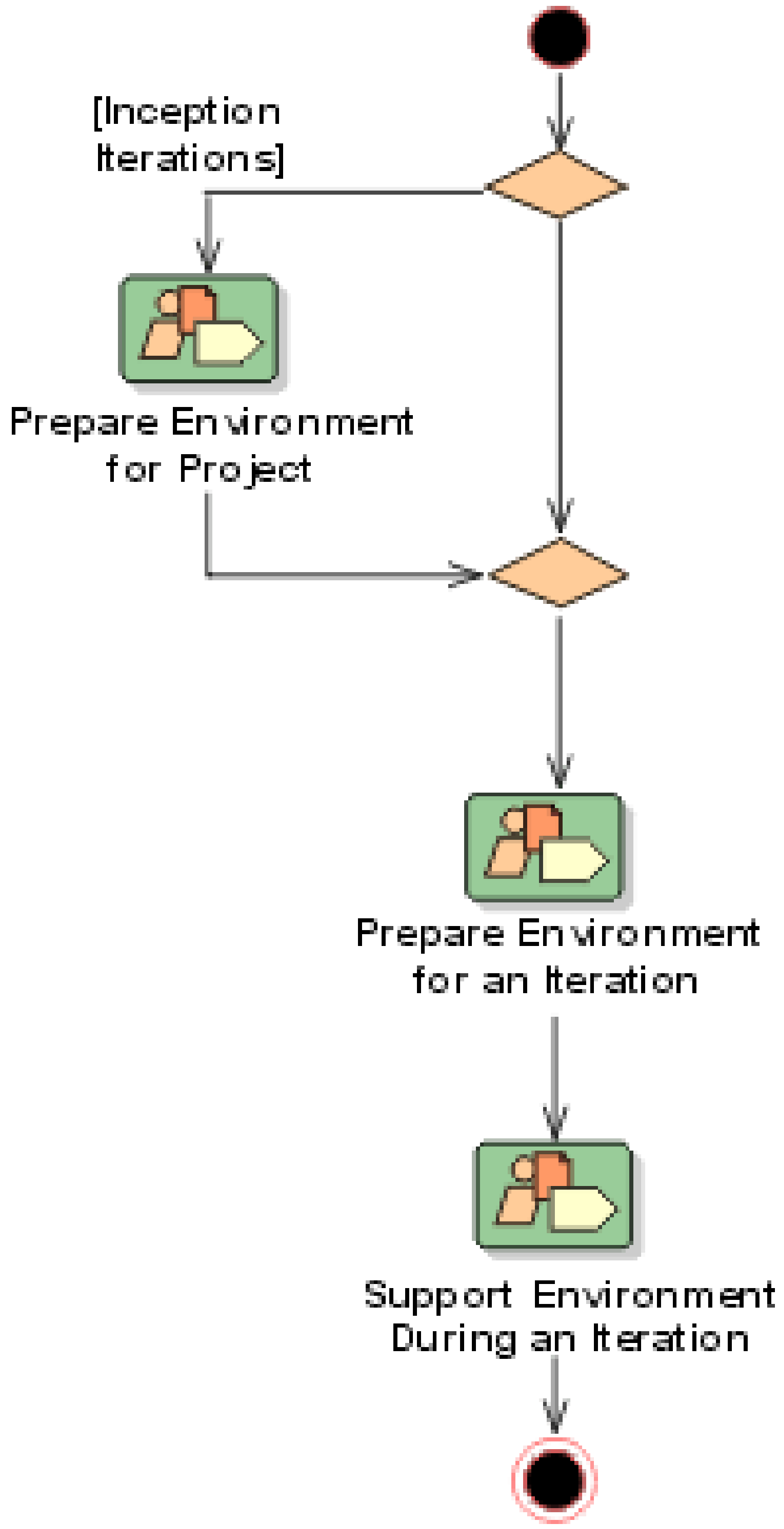
- Managing **people**: hiring, training, coaching
- Managing **budget**: defining, allocating, and so forth
- Managing **contracts**, with suppliers and customers

PROJECT MANAGEMENT: WHAT DOES COVER

This discipline focuses mainly on the important aspects of an iterative development process:

- Risk management
- Planning an iterative project, through the lifecycle and for a particular iteration
- Monitoring progress of an iterative project, metrics

ENVIRONMENT


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

ENVIRONMENT

- The environment discipline focuses on the activities necessary to configure the process for a project.
- It describes the activities required to develop the guidelines in support of a project.
- The purpose of the environment activities is to provide the software development organization with the software development environment—both processes and tools—that will support the development team.