AOT LAB

Agent and **O**bject **T**echnology **Lab**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Advanced Software Engineering

Unified Process

Prof. Agostino Poggi

Unified Process

- Unified Software Development Process (USDP), usually named Unified Process (UP) is a method of managing object-oriented software development
- UP is become the industrial standard for software development
- UP defines a very general development process that can be extended and modified
  - Rational Unified Process (RUP)
  - Open Unified Process (Open-UP)

UP Origins

1967 Ericsson builds the foundations of Component-Based Development

- A complex system is represented by a set of interconnected blocks

1976 CCITT released the Specification and Description Language

- Telecommunication system language where systems are modeled as a set of components exchanging signals

1987 Jacobson founded Objectory AB that put on the market a software engineering process: Objectory (Object Factory)

- Development process is based on a set of standard documents and a CASE tool

1995 Rational Rose bought Objectory AB

1997 UML becomes an OMG standard

1999 born UP

2003 IBM bought Rational

UP Good Practices

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

UP and UML

- UML should provide both a visual language and a software development process, but nowadays it only provides the visual language
- UP provides the software development process and uses UML as visual language
- Unified Process is a modeling technique where each model is a set of UML diagrams that represent various aspects of the software product we want to develop

UP as Generic Process

- UP is a generic process that must be instantiated for each organization and project

- Instantiation process should define and integrate:
  - Working team
  - Document models
  - Development tools (compilers, configuration managers, etc.)
  - Backup and storing (bug management, project management, etc.)
  - Development life-cycle

UP Axioms

- Use-case driven
  - Design and implementation realize the use cases
  - Testing is based on use cases
  - User documentation can be based on use cases
- Early risk identification and management
  - Early testing and deployment as opposed to late testing in traditional methods

UP Axioms

- Iterative and incremental
  - Projects are decomposed in sub-projects (iterations)
  - Each iteration releases some new functionalities
- Architecture-centric
  - An architecture is a view of the whole design, with emphasis on important characteristics (system components, components interactions, etc.)
  - Each iteration is based on the refinement of an executable architecture

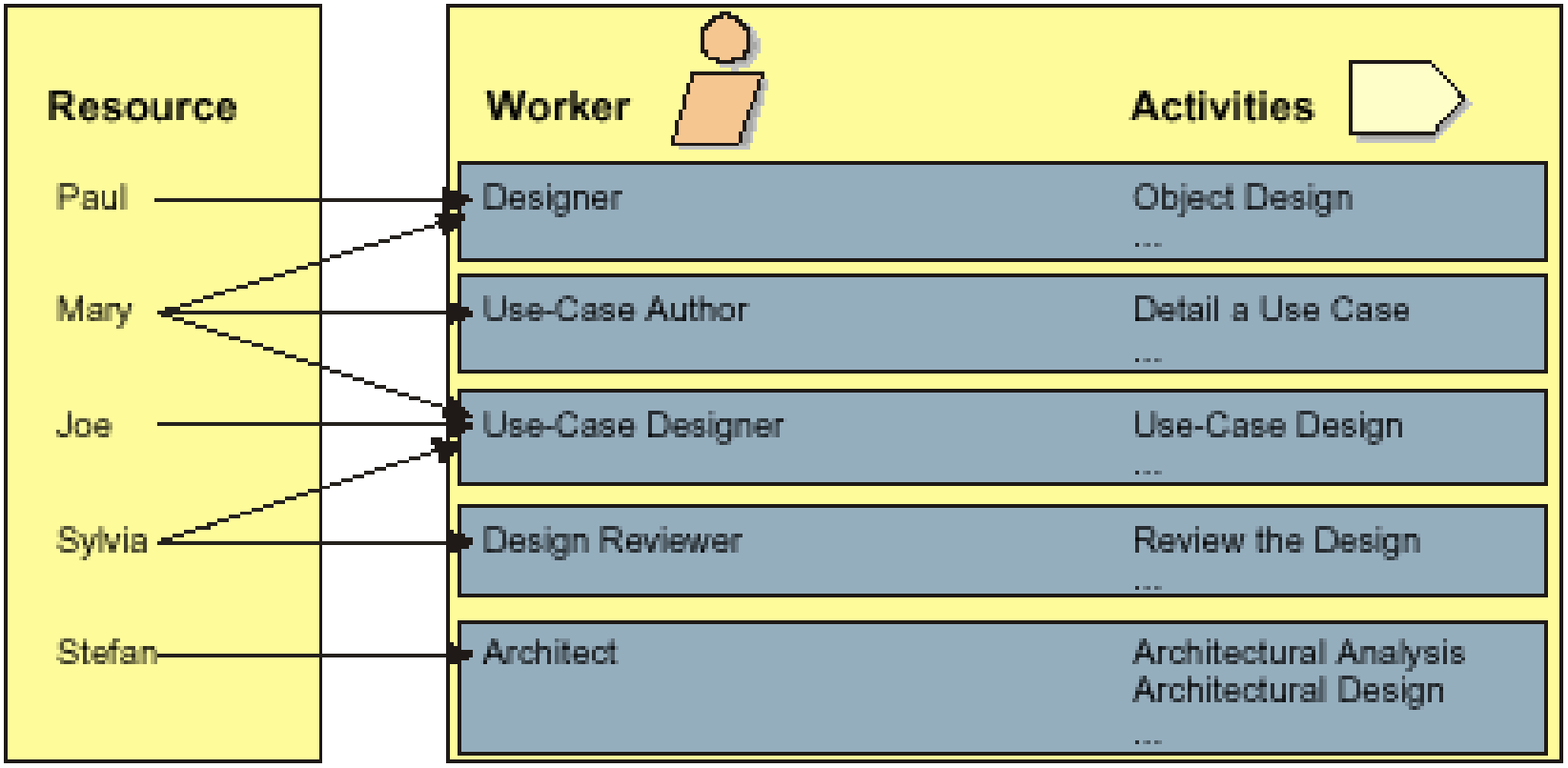
The Four 'P's

- People
  - Differing skills, experience - must make the project happen
  - Perform different roles during project's lifetime
- Project
  - A project is the collection of activities which finally result in a product
  - A project is an "instance" of a process
- Product
  - Complete system, including software binaries and all other collateral
- Process
  - A process is a description of how to conduct activities to create products

Workers

- Process Engineer
- System Analyst
- Use Case Specifier
- Use Case Engineer
- User Interface Designer
- Architect
- Component Engineer
- Test Engineer
- System Integrator
- Integration Tester
- System Tester
- Project Manager
- Stakeholder

Workers


Resource
Worker
Activities
Paul
Mary
Joe
Sylvia
Stefan
Designer
Object Design
...
Use-Case Author
Detail a Use Case
...
Use-Case Designer
Use-Case Design
...
Design Reviewer
Review the Design
...
Architect
Architectural Analysis
Architectural Design
...

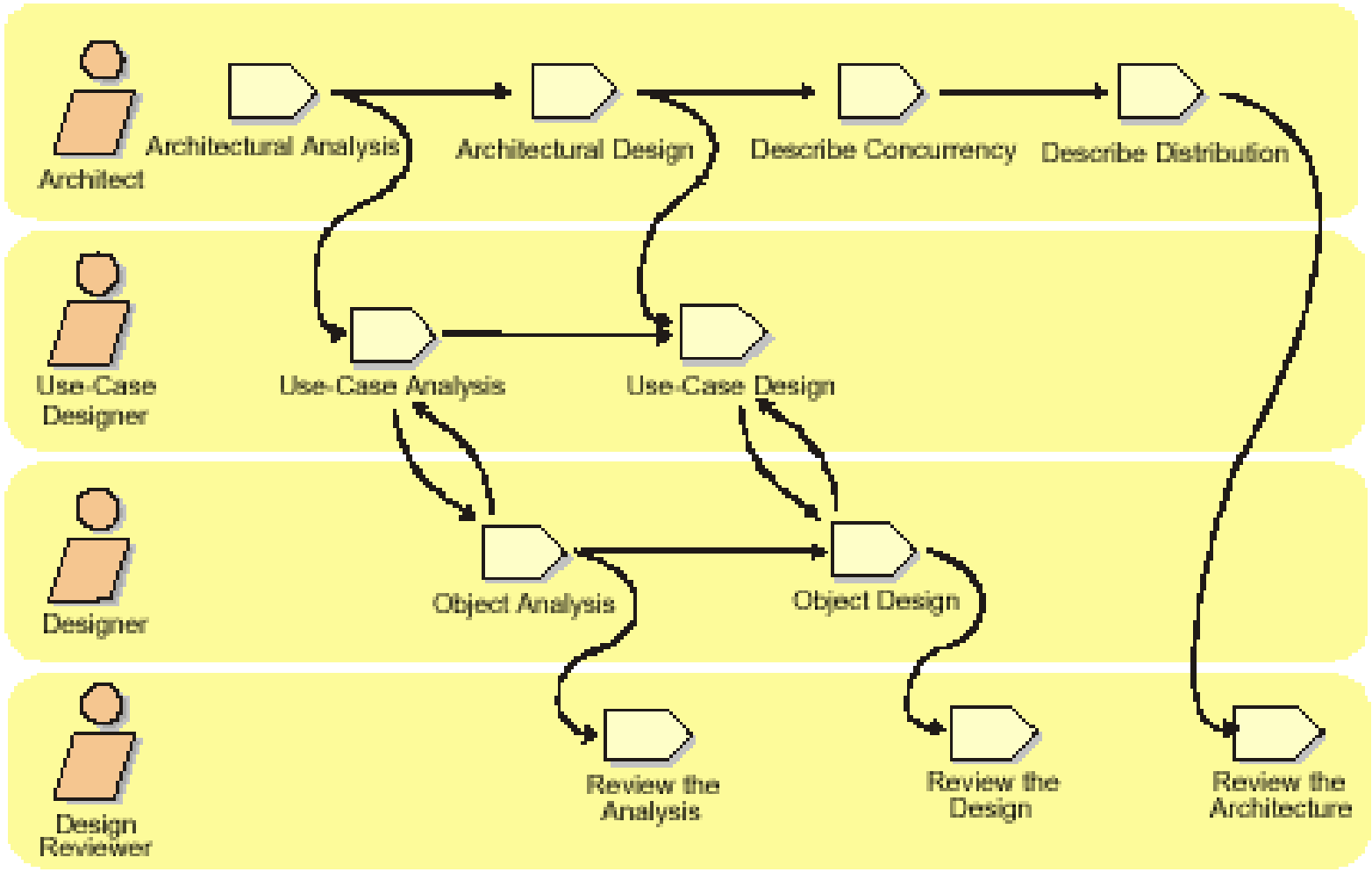
Iterations

- Iterations are sequences of activities with an established plan and evaluation criteria whose goals:
  - Refinement of an executable architecture
  - Definition of the related documentation
- Each iteration is based on the execution of five main software engineering activities (workflows):
  - Requirements
  - Analysis
  - Design
  - Implementation
  - Test

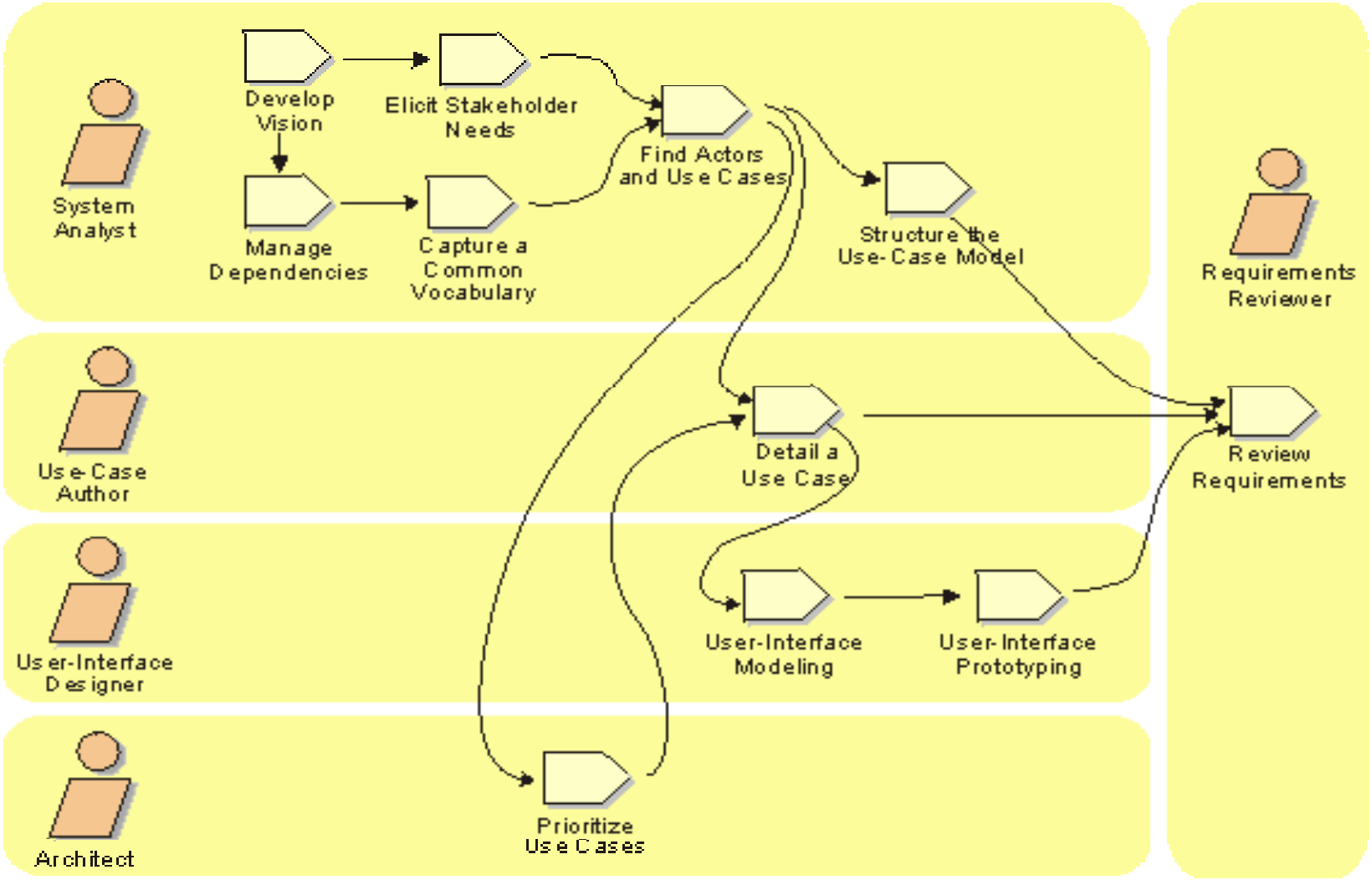
Workflows

- Requirements
  - Describes what the system is to do and allow developers and customers to agree on that description
- Analysis
  - Analyzes and refine requirements and defines a system model centered on system functionalities (analysis model)
- Design
  - Refines the analysis model until obtaining a model that can be implemented by programmers (design model)
  - It corresponds to the design of the system architecture and to the design of the components realizing the different system functionalities
- Implementation
  - Realizes the software units defining the different system components and integrates them
- Test
  - Verifies the correct behavior of the system

Workflows


Architect
Architectural Analysis
Architectural Design
Describe Concurrency
Describe Distribution
Use-Case Designer
Use-Case Analysis
Use-Case Design
Designer
Object Analysis
Object Design
Design Reviewer
Review the Analysis
Review the Design
Review the Architecture

Workflows


System Analyst
Develop Vision
Elicit Stakeholder Needs
Manage Dependencies
Capture a Common Vocabulary
Find Actors and Use Cases
Structure the Use-Case Model
Requirements Reviewer
Use-Case Author
Detail a Use Case
Review Requirements
User-Interface Designer
User-Interface Modeling
User-Interface Prototyping
Architect
Prioritize Use Cases

Workflows, Models and UML

- Requirements
  - Use Case Model (Use Case, Sequence, State, Activity)
- Analysis
  - Analysis Model (Class, Sequence, Communication, State, Activity)
- Design
  - Design Model (Class, Sequence, Communication, State, Activity)
  - Deployment Model (Deployment, Sequence, Communication)
- Implementation
  - Implementation Model (Component, Sequence, Communication)
- Test
  - Test Model

Artifacts

- Requirements Set
  - Use case models, UI prototypes, requirements, description of business issues and context of system
- Design Set
  - Description of how the system will be constructed - static and dynamic design models
- Implementation Set
  - Source code, data files, database schema
- Deployment Set
  - How software is put on customer’s machine (installation and configuration management)

Milestones

- Milestones are synchronization points between development team, project manager and customers
- Milestones are decision points:
  - Continuation or stopping of the project
  - Re-working of the last iteration if problems have been discovered
  - Schedule updating
  - Decide has the expected work items really been completed
- Milestones usually correspond to the delivery of some project artifacts

Development Phases

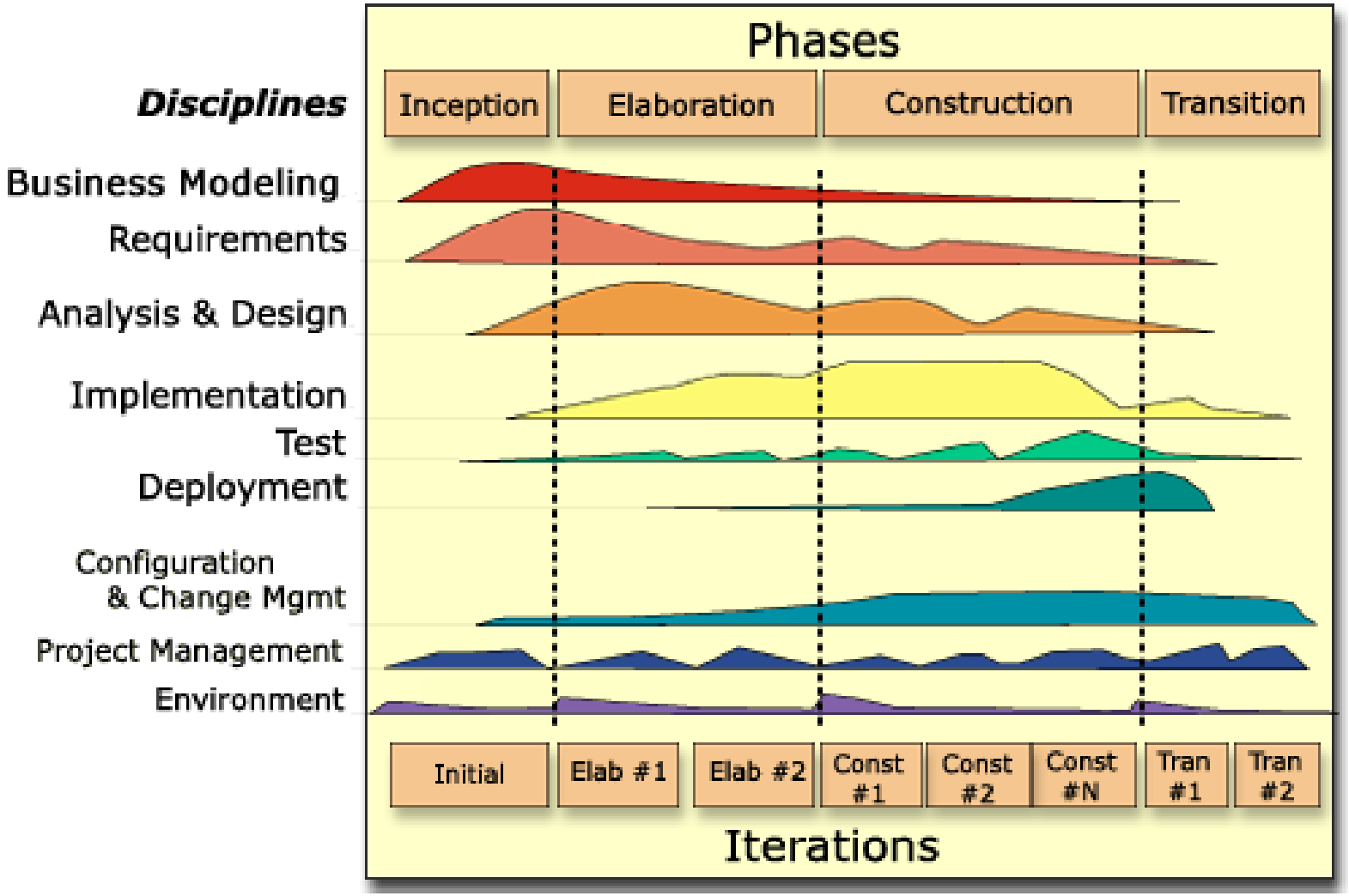
- System development process is divide in four phases:
  - **Inception**: establishes the business case for the system
  - **Elaboration**: develops an understanding of the problem domain and the system architecture
  - **Construction**
    - Completes system design
    - Performs programming and testing
  - **Transition**: deploys the system in its operating environment
- Some processes add two additional phases:
  - **Production**: provides assistance to the user and upgrades the system
  - **Retirement**: guides the phase out of the system

Development Phases

- Development phases results are important milestone:
  - Vision (goals)
  - Baseline architecture
  - Initial capability
  - Product release

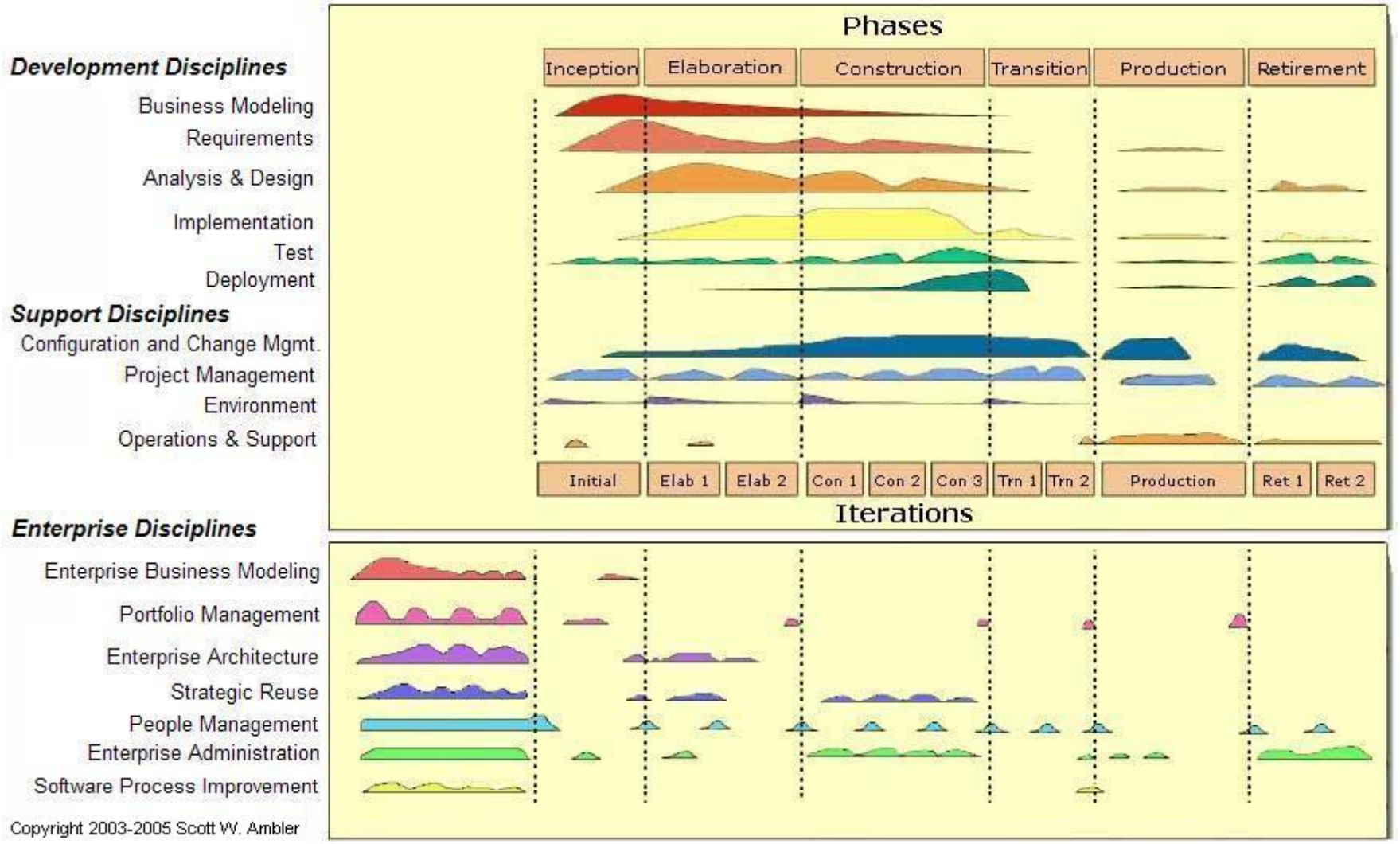
AOT LAB

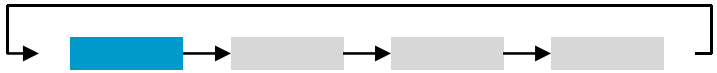
Development Phases and Workflows



AOT LAB

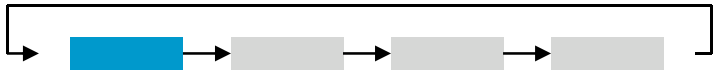
Development Phases and Workflows





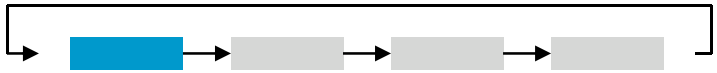
Inception

- Main goal
  - Determine whether the proposed software product is economically viable
- Other goals
  - Gain an understanding of the domain
  - Build the business model
  - Delimit the scope of the proposed project
  - Begin to make the initial business case
- Milestone
  - **Vision** (project goals)



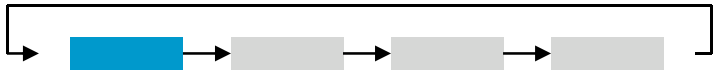
Workflows

- Work is mainly done in the requirements workflow and then in the test workflow. At the end of the phase, work starts in the analysis and design workflows and if necessary in the implementation workflow
- Requirements
  - An initial set of requirements is defined
- Analysis
  - Information for the design of the system architecture may be extracted
- Design and Implementation
  - Some design about the system architecture can be done
  - A proof-of-concept prototype is sometimes build to test the feasibility of constructing part of the software product
- Test
  - Tests can be defined to check requirements satisfiability



Deliverables

- A vision document illustrating the core project's requirements, key features, and main constraints
- An initial use-case model (10%-20% complete)
- An initial project glossary (may optionally be partially expressed as a domain model)
- An initial business case, which includes business context, success criteria (revenue projection, market recognition, and so on), and financial forecast
- An initial risk assessment
- A project plan, showing phases and iterations
- A business model, if necessary



Evaluation

- Stakeholder concurrence on scope definition and cost/schedule estimates
- Requirements understanding as evidenced by the fidelity of the primary use cases
- 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

- Main goals
  - Analyze the problem domain
  - Establish a sound architectural foundation
  - Develop the project plan
  - Eliminate the highest risk elements of the project
- Milestone
  - **Baseline architecture**

Workflows

- Work is mainly done in the analysis workflow and then in the design workflow. At the end of the phase, the work in the implementation workflow becomes the most important
- Requirements
  - Define the large part of requirements
- Analysis
  - Define what must be realized
- Design and Implementation
  - Design and realize the baseline architecture
- Test
  - Test the baseline architecture

Deliverables

- A use-case model (at least 80% complete) - all use cases and actors have been identified, and most use-case descriptions have been developed
- Supplementary requirements capturing the non functional requirements and any requirements that are not associated with a specific use case
- A software architecture description
- An executable architectural prototype

Deliverables

- A revised risk list and a revised business case
- A development plan for the overall project, including the coarse-grained project plan, showing iterations and evaluation criteria for each iteration
- An updated development case specifying the process to be used
- A preliminary user manual (optional)

Evaluation

- Is the vision of the product stable?
- Is the architecture stable?
- Does the executable demonstration show that the major risks have been addressed and credibly resolved?
- Is the plan for the construction phase sufficiently detailed and accurate?
- Do all stakeholders agree that the current vision can be achieved if the current plan is executed to develop the system, in the context of the current architecture?
- Is the actual resource expenditure versus planned expenditure acceptable?

Construction

- Main goal
  - Produce the first operational-quality version of the software product (beta release)
- Milestone
  - **Initial operational capability**

Workflows

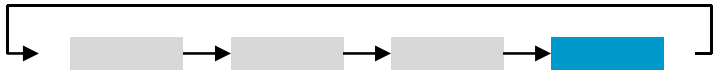
- Work is mainly done in the implementation workflow and then in the test workflow because for each iteration both unit tests and integration tests must be executed
- Requirements
  - Complete the requirements
- Analysis
  - Complete the analysis model
- Design
  - Complete the design model
- Implementation
  - Build the “initial operational capability”
- Test
  - Test the “initial operational capability”

Deliverables

- The initial user manual and other manuals, as appropriate
- All the artifacts (beta release versions)
- The completed architecture
- The updated risk list
- The project management plan (for the remainder of the project)
- If necessary, the updated business case

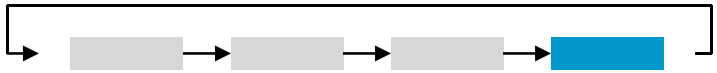
Evaluation

- Is this product release stable and mature enough to be deployed in the user community?
- Are all stakeholders ready for the transition into the user community?
- Are the actual resource expenditures versus planned expenditures still acceptable?



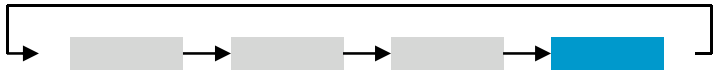
Transition

- Main goals
  - Achieving user self-supportability
  - Achieving stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision
  - Achieving final product baseline as rapidly and cost effectively as practical
- Milestone
  - **Product release**



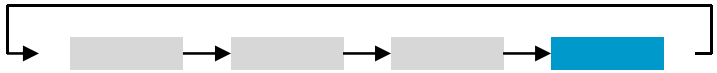
Workflows

- This phase is based on the implementation and test workflows. However, some design may be useful for the correction of some design errors recognized during the beta test phase
- Design
  - Models are updated to cope with errors recognized during the beta test phase
- Implementation
  - Software is personalized for client sites and errors recognized during the beta test are corrected
- Test
  - Beta test and acceptance test is performed at the client sites



Deliverables

- All the artifacts (final versions)
- The completed manuals
- Installation and configuration scripts
- Sales and marketing materials
- Maintenance issues

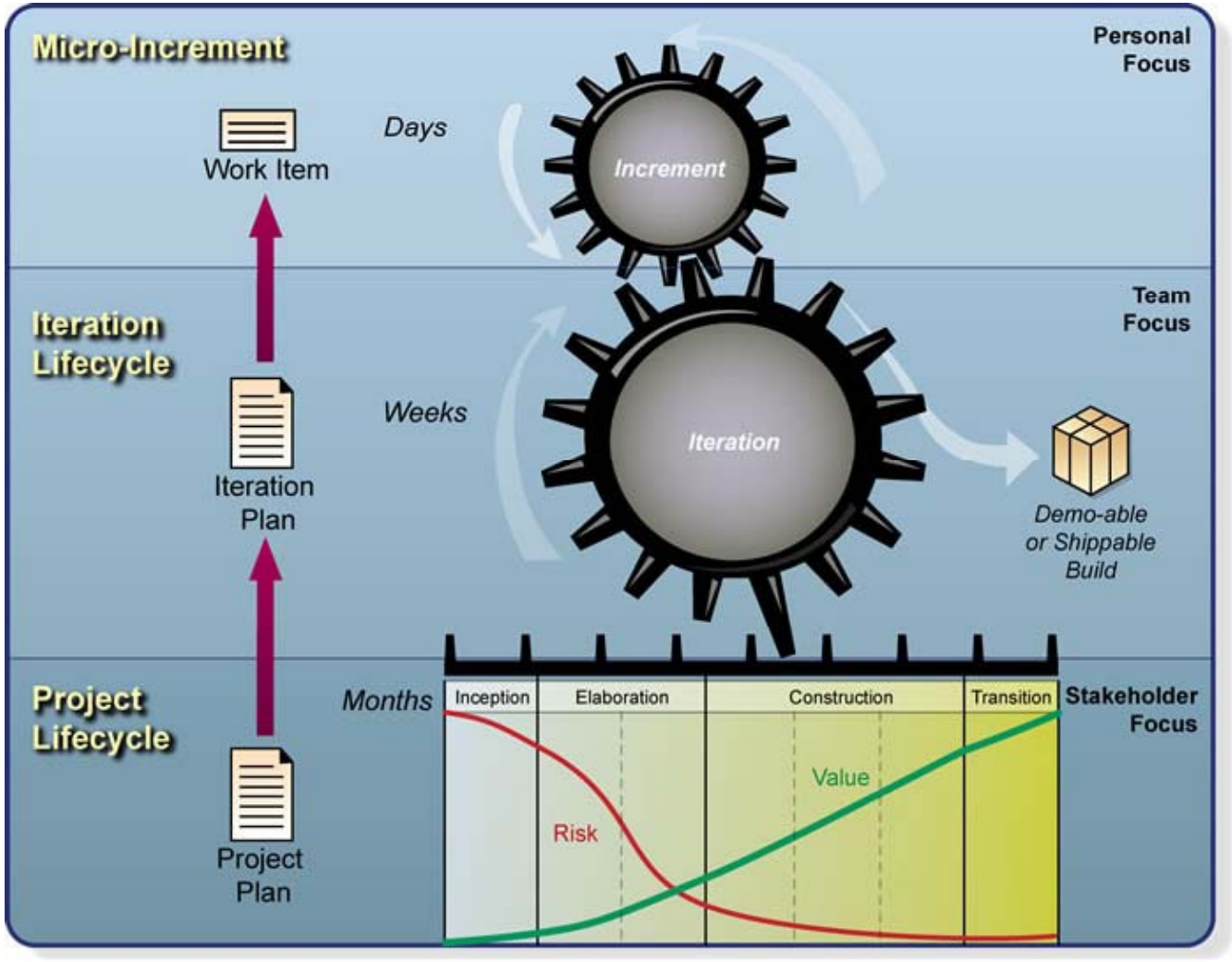


Evaluation

- Is the user satisfied?

- Are the actual resources expenditures versus planned expenditures still acceptable?

OpenUP



http://epf.eclipse.org/wikis/openup/

- Minimal
  - Only fundamental content is included
- Complete
  - It can be manifested as an entire process to build a system
- Extensible
  - It can be used as a foundation on which process content can be added or tailored as needed

OpenUP Principles

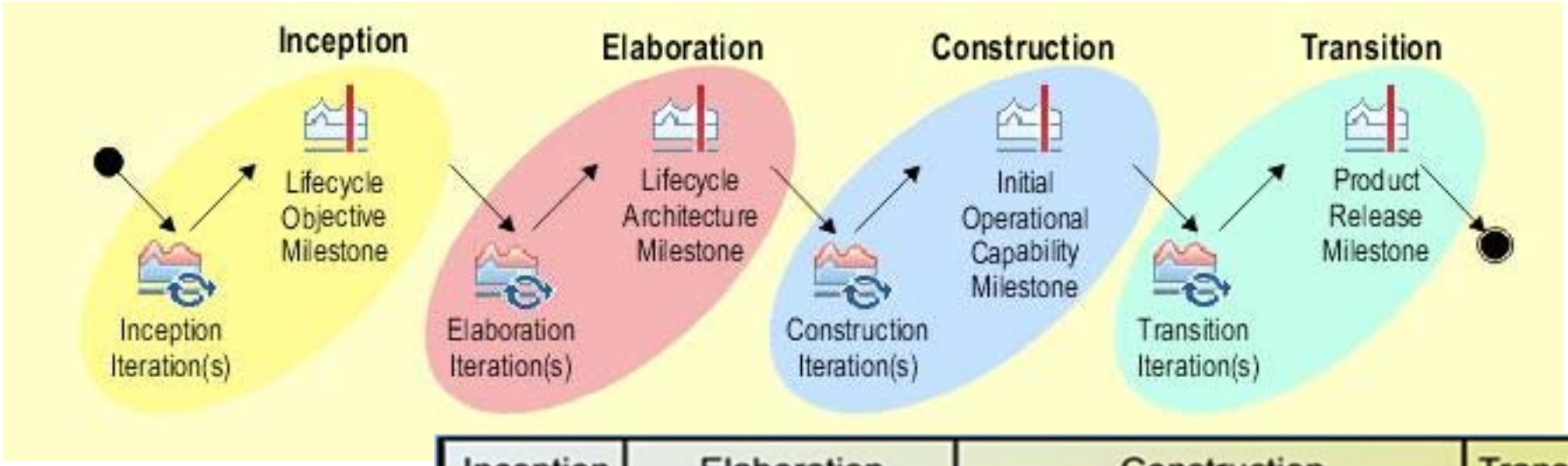
- Balance competing priorities to maximize stakeholder value
  - Promote practices that allow project participants and stakeholders to develop a solution that maximizes stakeholder benefits, and is compliant with constraints placed on the project
- Collaborate to align interests and share understanding
  - Promote practices that foster a healthy team environment, enable collaboration and develop a shared understanding of the project

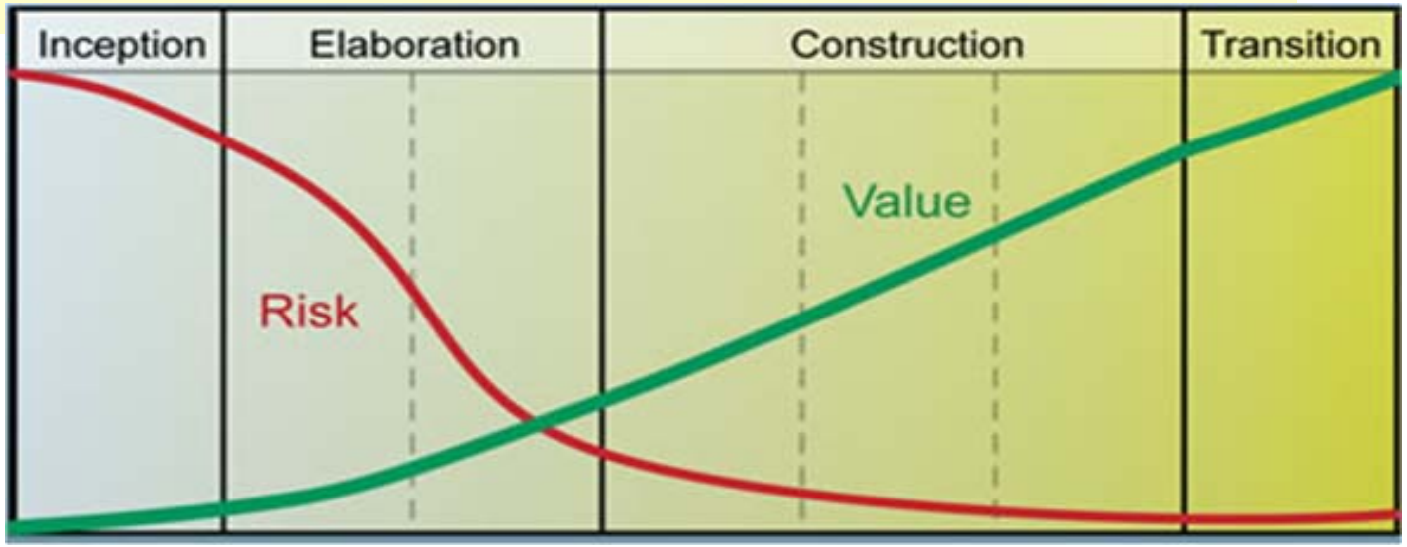
OpenUP Principles

- Focus on the architecture early to minimize risks and organize development
  - Promote practices that allow the team to focus on architecture to minimize risks and organize development
- Evolve to continuously obtain feedback and improve
  - Promote practices that allow the team to get early and continuous feedback from stakeholders, and demonstrate incremental value to them

AOT LAB

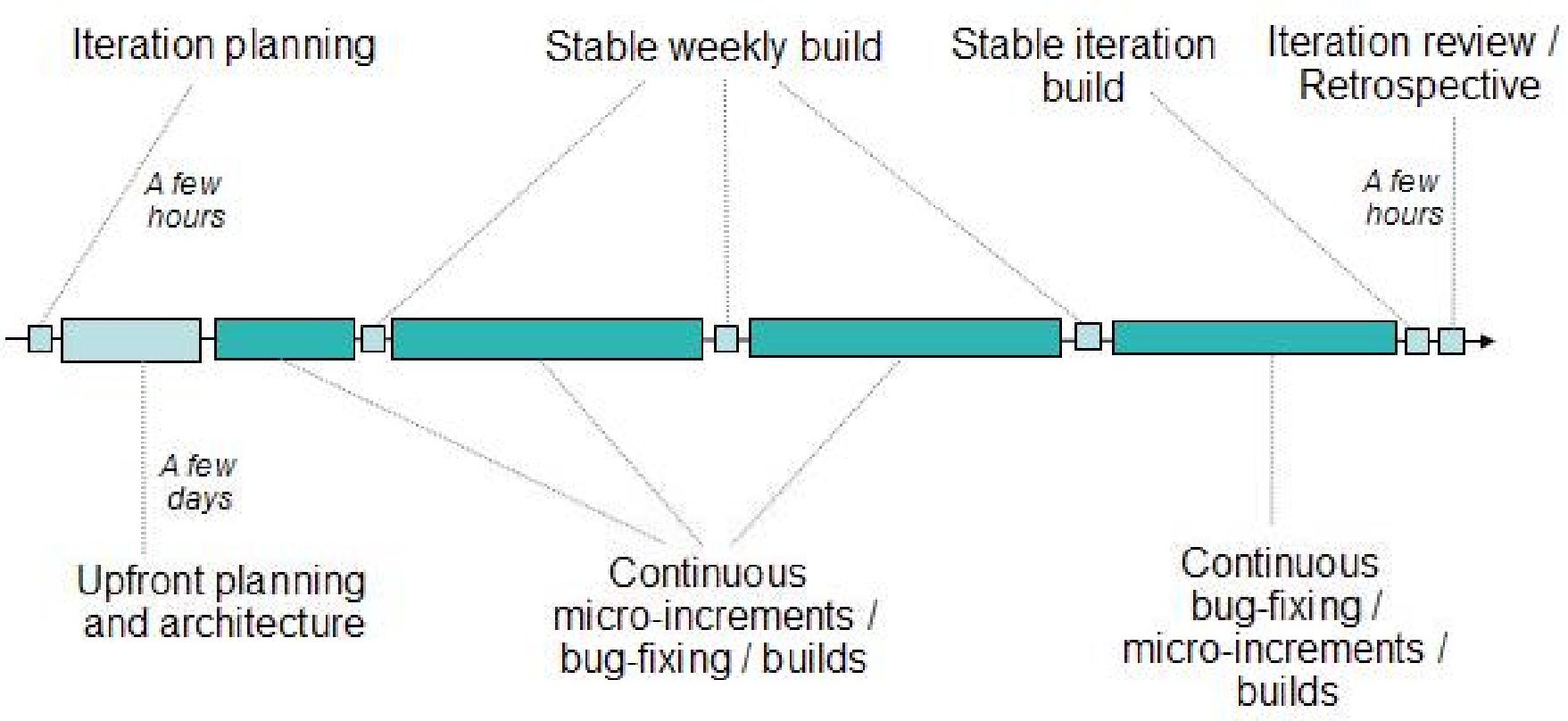
Project Lifecycle





AOT LAB

Increment Lifecycle



Micro-Increment

- A micro-increment is a small, measurable step towards reaching the goals of an iteration
- A micro-increment represents the outcome of a few hours to a few days of work performed by one or typically a few people collaborating to reach a goal of the iteration
- Micro-increments provide an extremely short feedback loop that drives adaptive decisions within each iteration

AOT LAB

Work Products

- Requirements
- Architecture
- Development
- Test
- Project Management

Glossary
System-Wide Requirements
Use Case
Use-Case Model
Vision

Architecture Notebook

Build
Design
Developer Test
Implementation

Test Case
Test Log
Test Script

Iteration Plan
Project Plan
Risk List
Work Items List

Roles

- Analyst
- Architect
- Developer
- Tester
- Project Manager
- Stakeholder

Analyst

Modifies

Glossary	System-Wide Requirements	Use Case
Use-Case Model	Vision	

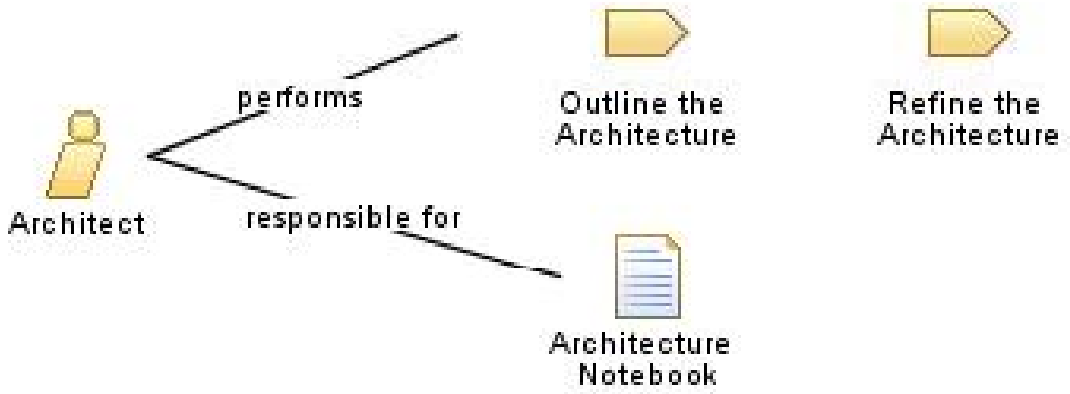
Additional Performs

Assess Results	Create Test Cases	Design the Solution
Implement Tests	Manage Iteration	Outline the Architecture
Plan Iteration	Plan Project	

Architect

Modifies

Architecture Notebook		



Additional Performs

Assess Results	Design the Solution	Detail System-Wide Requirements
Detail Use-Case Scenarios	Develop Technical Vision	Identify and Outline Requirements
Manage Iteration	Plan Iteration	Plan Project

AOT LAB

Developer

Modifies

Build	Design	Develop Test
[Software Implementation] Build Implementation	[Technical Specification] Glossary System-Wide Requirements Use Case, & Use Case Model Vision	[Technical Test Results] Test log

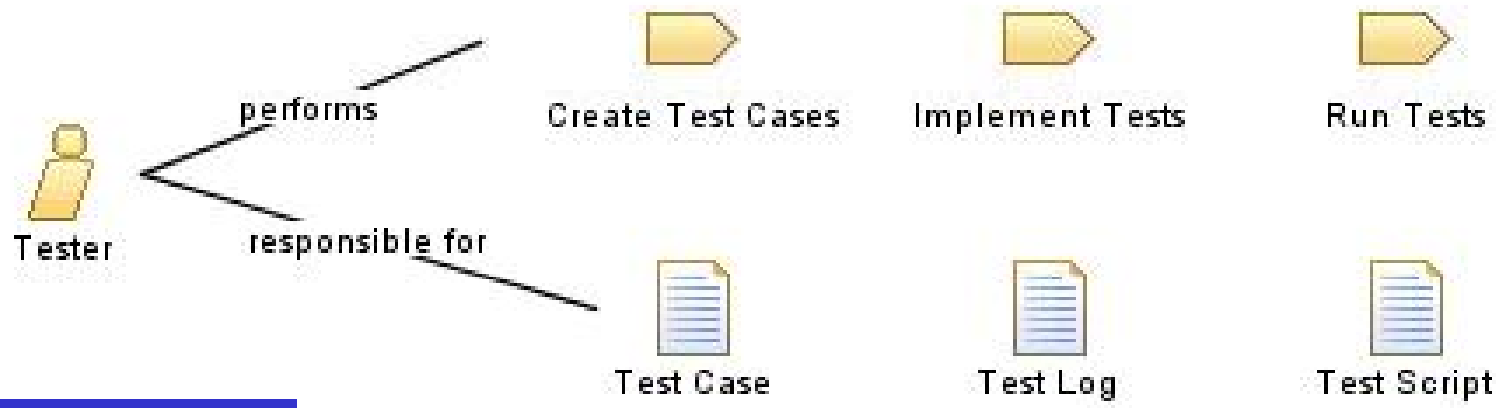
Additional Performs

Assess Results	Identify and Outline Requirements	Detail System-Wide Requirements
Create Test Cases		
Implement Tests	Manage Iteration	Outline the Architecture
Plan Iteration	Plan Project	Refine the Architecture

Tester

Modifies

Test Case	Test Log	Test Script



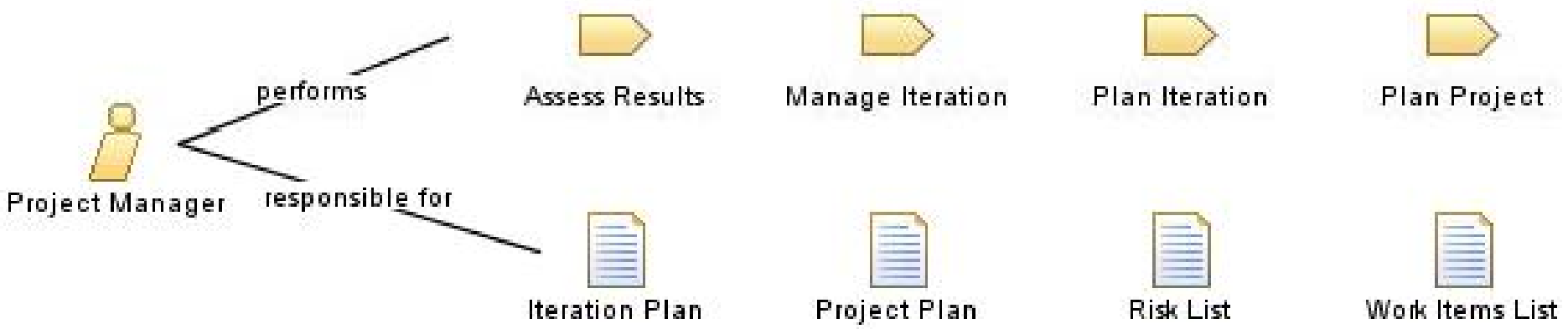
Additional Performs

Assess Results	Design the Solution	Detail System-Wide Requirements
Detail Use-Case Scenarios	Identify and Outline Requirements	Implement Developer Tests
Implement Solution	Manage Iteration	Plan Iteration
Plan Project		

Project Manager

Modifies

Iteration Plan	Project Plan	Risk List
Work Items List		



Additional Performs

Develop Technical Vision	Outline the Architecture	Refine the Architecture

Stakeholder

Additional Performs

Assess Results	Create Test Cases	Design the Solution
Detail System-Wide Requirements	Detail Use-Case Scenarios	Develop Technical Vision
Identify and Outline Requirements	Implement Solution	Implement Tests
Manage Iteration	Outline the Architecture	Plan Iteration
Plan Project		