Xelaration

■■■ Software Corporation ■■■

Business Analysis Best Practices of the Unified Process

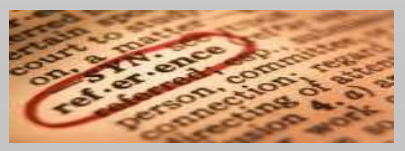
Version 2007.05.28

Objectives of this Presentation

- To understand what the Unified Process is.
- To understand what portions of the Unified Process can be applied to implement a sound business analysis process.
- To review case studies where the Unified Process was applied to implement business analysis processes.

References

- IBM Rational Unified Process (RUP) Version 7.0
- Eclipse Open Unified Process (OpenUP) Version 0.9
- A Guide to the Business Analysis Body of Knowledge (BABOK) Version 1.6.



Agenda

1. What is the Unified Process?
2. How can the Unified Process Help in Business Analysis?
3. Case Studies

1. What is the Unified Process?

- Topics
  - Definition of a Software Development Process
  - Definition of the Unified Process
  - Best Practices and the Unified Process
  - Characteristics of the Unified Process
  - Scope of the Unified Process
  - Benefits and Challenges of Implementing the Unified Process
  - Recommendations for Implementing the Unified Process
  - Instances of the Unified Process
  - Structure of the Unified Process

Definition of a Software Development Process

- A software development process defines who does what, when and how to reach the goals of a software development project

Definition of the Unified Process

- Software development process
  - Set of activities that transform stakeholder needs into a software system that fulfill these needs
- Generic process framework
  - Can be tailored for different types and sizes of projects, and business domains
    - Project types: green-field development, evolution of legacy systems, packaged application implementation, maintenance, etc.
    - Project size: small, medium, large
    - Business domains: banking, insurance, bio-medical, military, aerospace, etc.

Best Practices and the Unified Process

- Unified Process implements 6 best practices of software engineering

Unified Process

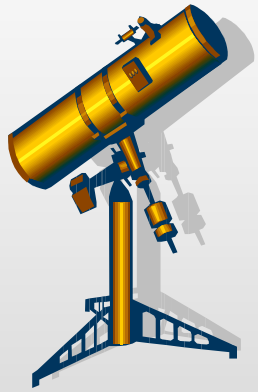
- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually
- Continuously Verify Quality
- Manage Change

Characteristics of the Unified Process

- Use-case driven
  - To drive process activities from functional requirements expressed using use case modeling techniques
- Architecture-centric
  - To build an executable architecture early in order to address important behavioral and structural system considerations
- Iterative and incremental
  - To partition the project lifecycle into multiple mini-waterfall projects each resulting in an executable release in order to mitigate risks

Scope of the Unified Process

- In-scope
  - Software development
  - Green-field development
  - Object-oriented paradigm
  - Project lifecycle
- Out-of-Scope
  - System development
  - Maintenance
  - Functional paradigm
  - Product lifecycle



Unified Modeling Language™ (UML™)

- Industry standard from the Object Modeling Group (OMG®) used to describe the behavior and structure of software-intensive systems
- Visual language that includes building blocks with clearly defined purpose and semantics
- Provides different types of diagrams to describe the different perspectives of a software system

Unified Process and the UML™

- Unified Process produces models
  - Business Use Case Model
  - Business Analysis Model
  - Business Design Model
  - Business Deployment
  - Use-Case Model
  - Analysis Model
  - Design Model
  - Data Model
  - Implementation Model
  - Deployment Model
- Models are built using UML™ diagrams

Benefits of Implementing the Unified Process

- Provides a common lexicon for all workers on a project
- Increases project outcome predictability
- Reduces project risk




Challenges of Implementing the Unified Process

- Many disciplines, roles, tasks and artifacts
- Implementing all of the Unified Process may take years
- "What do we implement first?"

Recommendations for Implementing the Unified Process

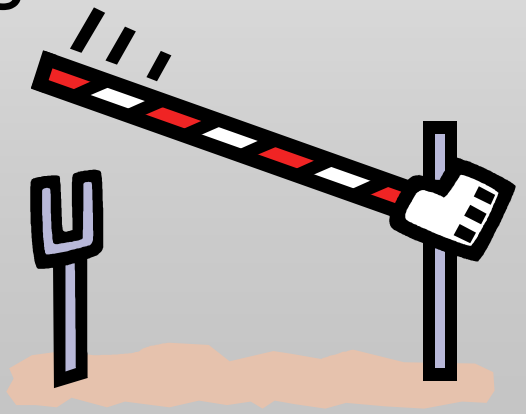
- Implement the process incrementally
- Start on a small pilot project (a real one)

Instances of the Unified Process

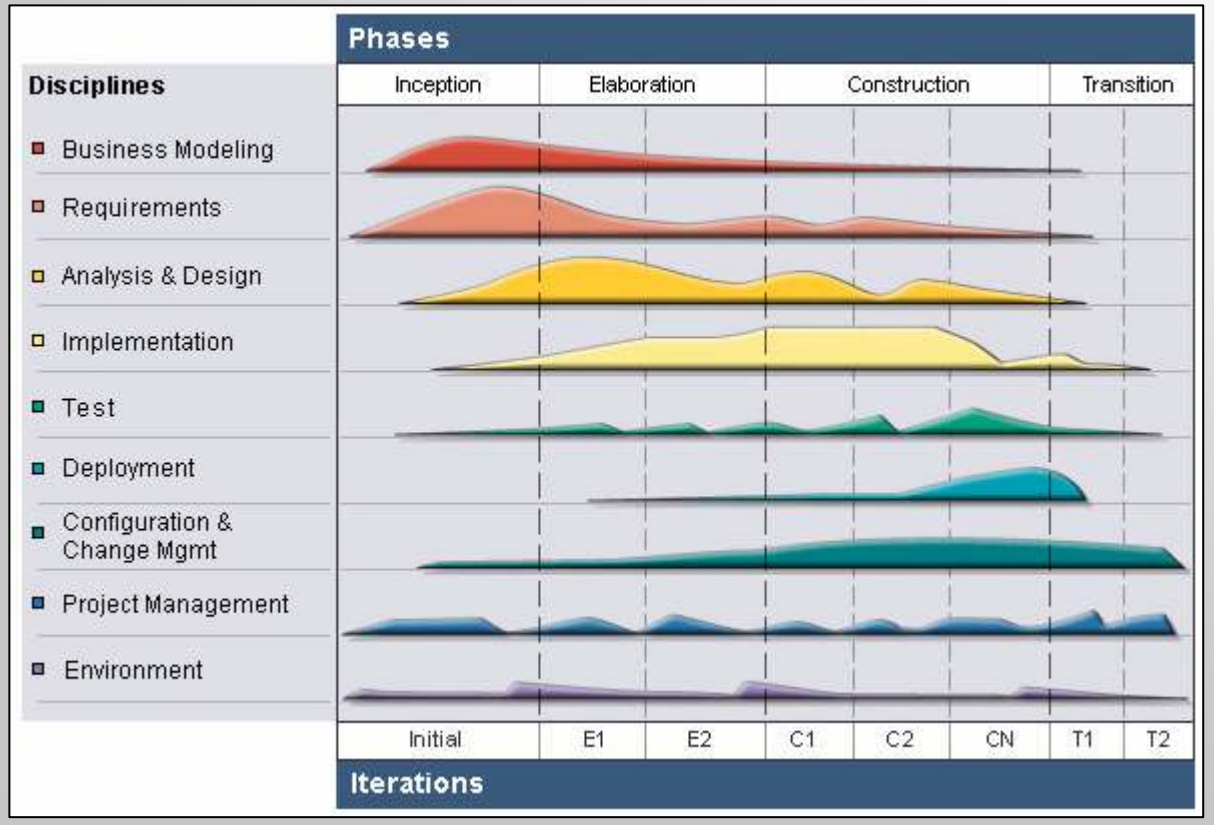
- The Unified Software Development Process
  - Book from Ivar Jacobson, Grady Booch and James Rumbaugh originally published in 1999
  - Origin of the Unified Process
- IBM® Rational® Unified Process® (RUP®)
  - For small to large scale projects
  - Product from IBM® Rational® Software
  - Current version: 7.0
- Eclipse Open Unified Process (OpenUP)
  - For small scale projects only
  - Product from the Eclipse Foundation
  - Current version: 0.9

IBM® Rational® Unified Process® (RUP®)

- From this point on, this presentation focuses on the IBM® Rational® Unified Process® (RUP®) since it is the more complete, up-to-date and mature of all Unified Process instances

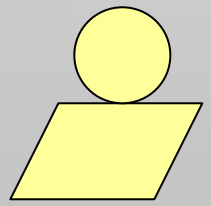
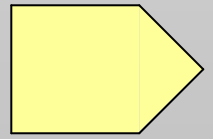
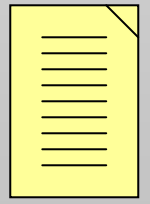


Structure of the Unified Process



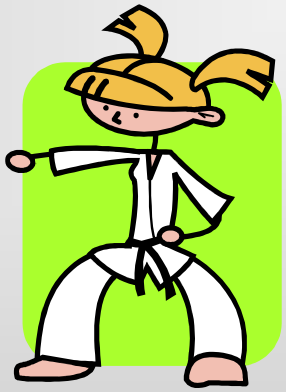
Building Blocks of the Unified Process

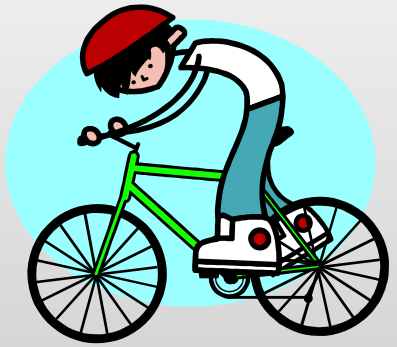
- Two Dimensions
  - Content
    - Disciplines, Workflows , Activities, Roles, Tasks and Artifacts
  - Time
    - Lifecycle, Phases, Workflows, Iterations, Milestones

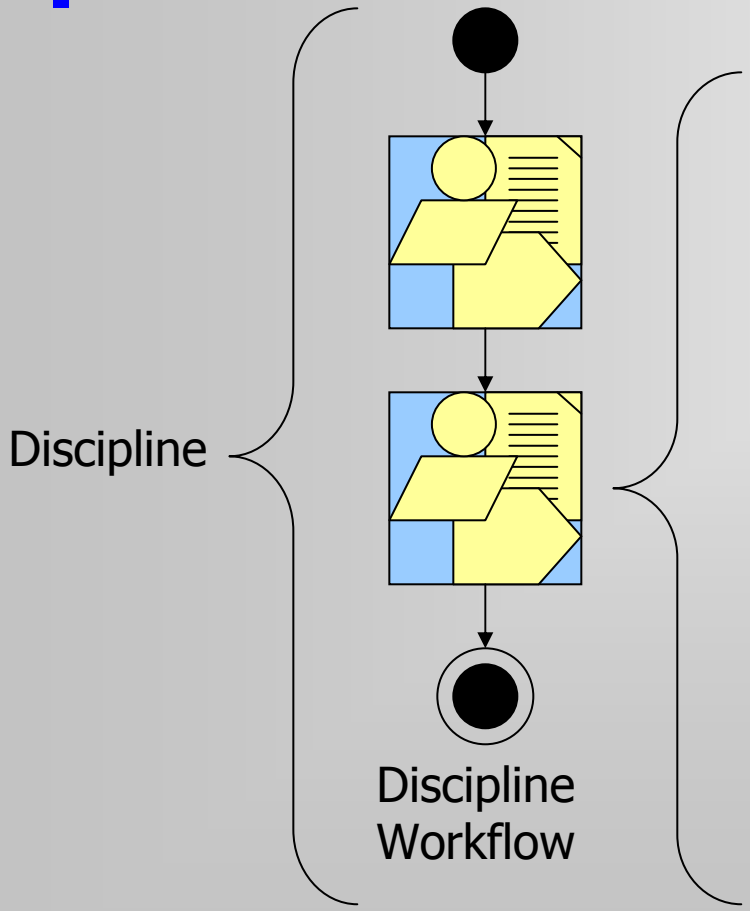
Disciplines of the Unified Process

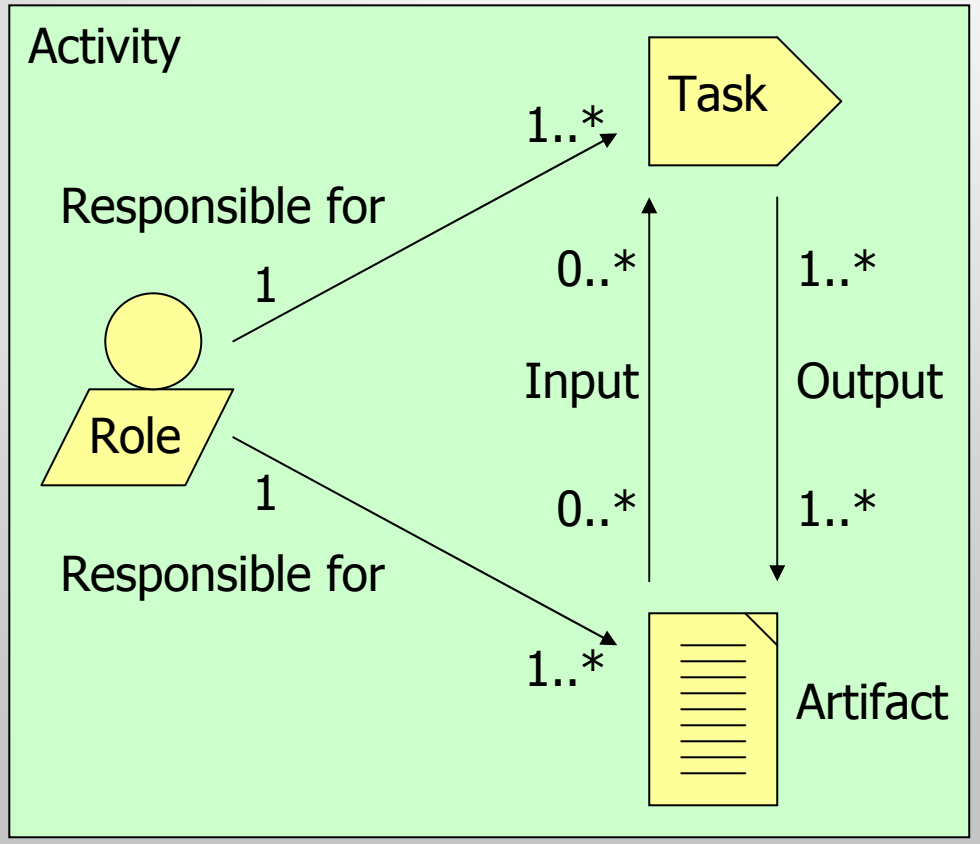
- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Management
- Project Management
- Environment



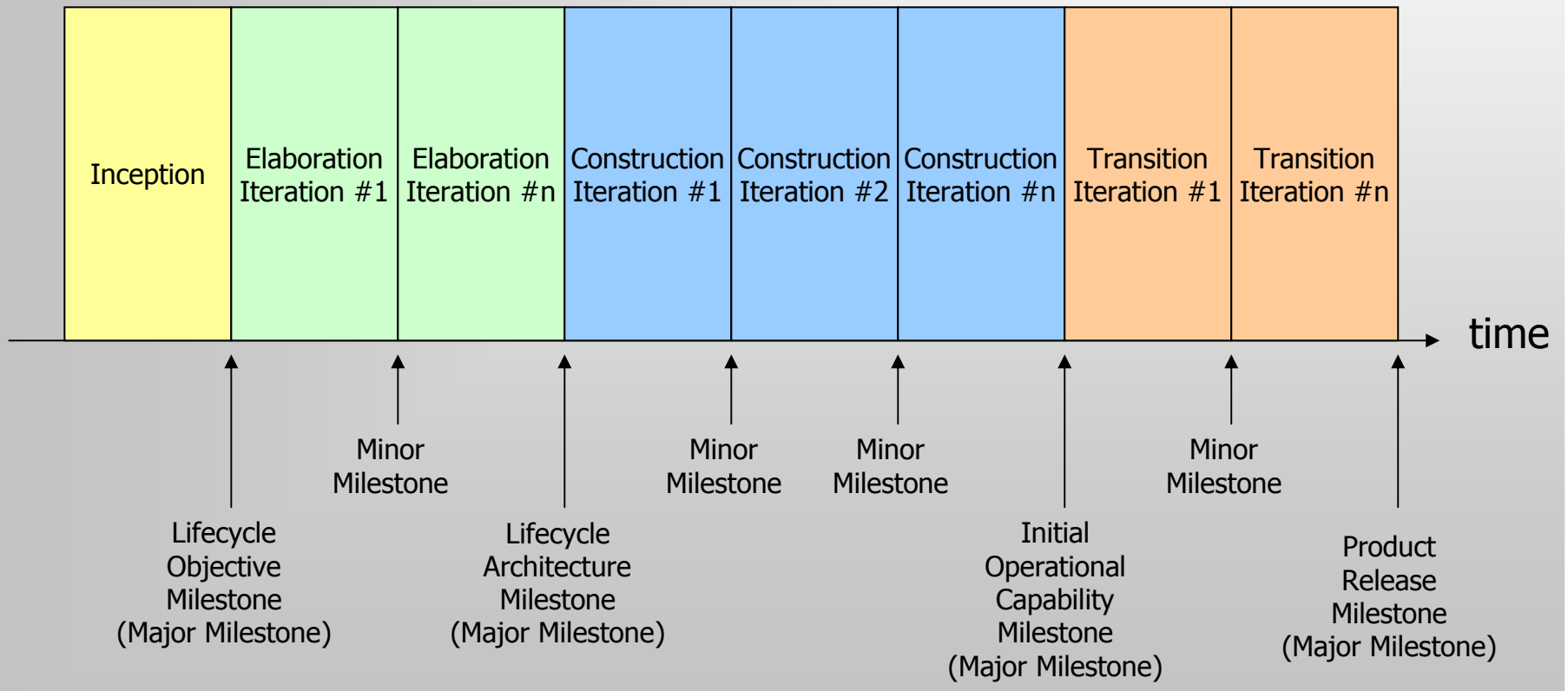


Disciplines, Workflows, Activities, Roles, Tasks and Artifacts





Phases, Iterations and Milestones

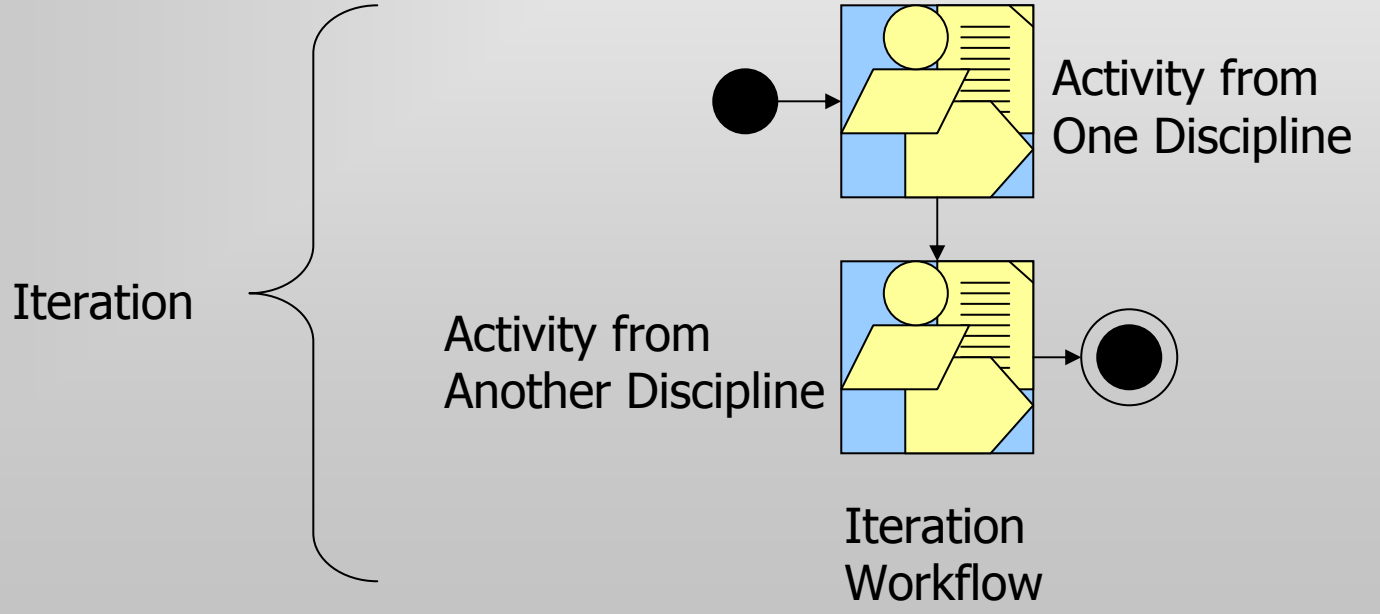


Lifecycle of the Unified Process

- 4 Phases and Major Milestones
  - Inception → Lifecycle Objective Milestone
    - Define project scope
  - Elaboration → Lifecycle Architecture Milestone
    - Build executable system architecture
  - Construction → Initial Operational Capability Milestone
    - Complete Beta version of the software system
  - Transition → Product Release Milestone
    - Complete GA version of the software system

Iterations, Workflows and Activities

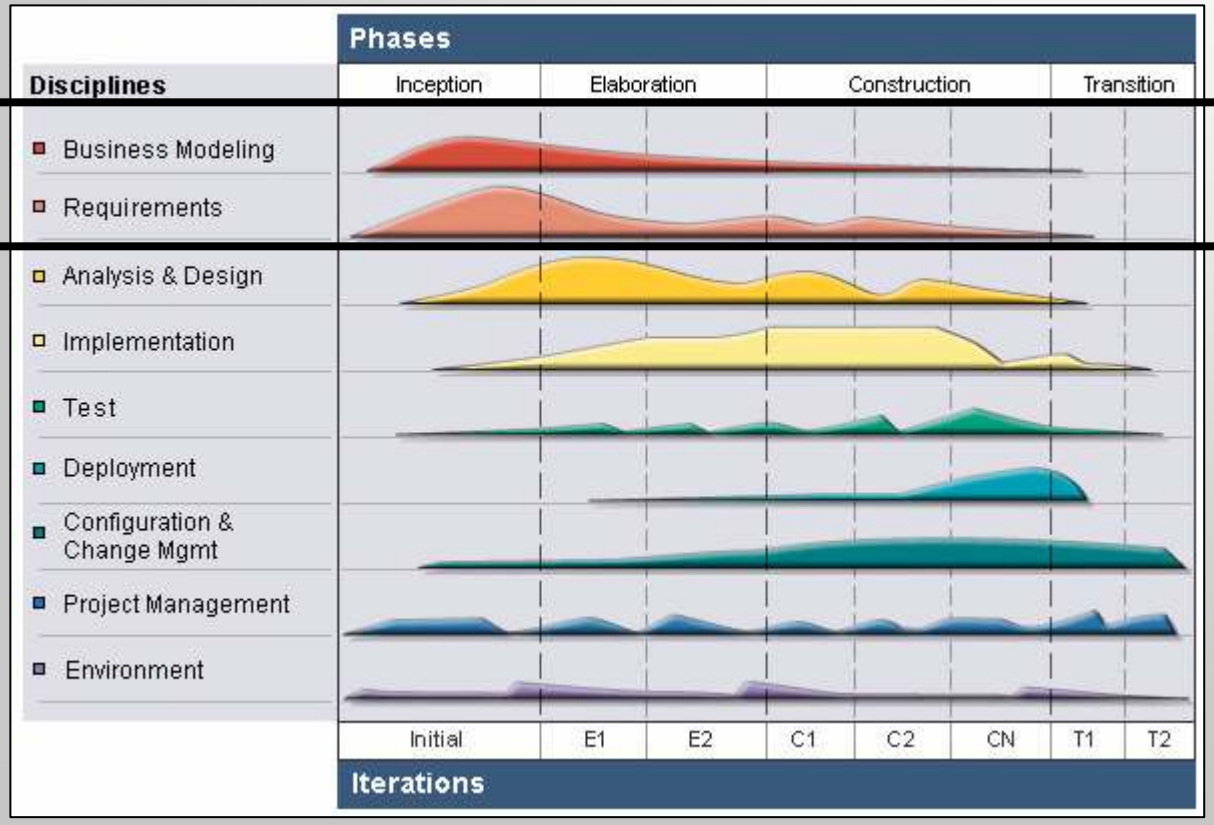
- Mini-waterfall project
- All disciplines are visited with a relative level of effort



2. How can the Unified Process Help in Business Analysis?

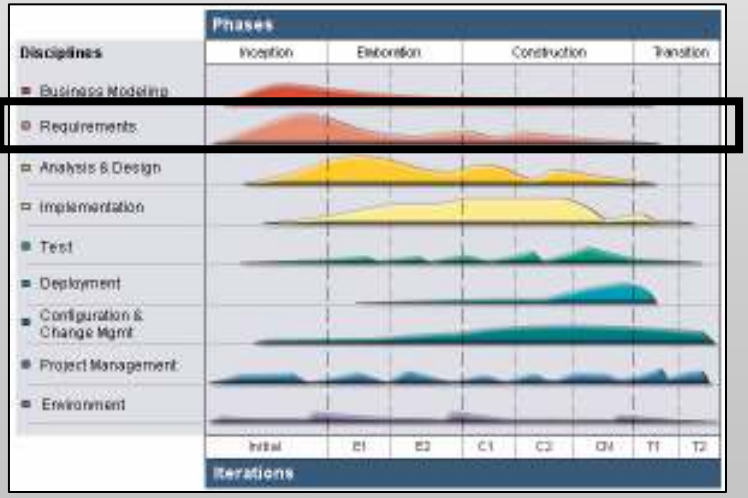
- Topics
  - Requirements Discipline
  - Business Modeling Discipline

Disciplines Relevant to Business Analysis



Requirements Discipline

- Topics
  - Best Practice: Manage Requirements
  - Requirement Definition
  - Requirements Management Definition
  - Functional vs. Non-Functional Requirements (FURPS+)
  - Hierarchy of Requirements
  - Requirements Attributes
  - Requirements Traceability
  - Roles
    - System Analyst
    - Requirements Specifier
  - Artifacts
    - Requirements Management Plan
    - Vision Document
    - Use Case Model
    - Supplementary Specifications



Best Practice: Manage Requirements

- Understand the problem (the what) before defining the solution (the how)
- Formally capture and document requirements
- Manage requirements with attributes and traceability to ensure coverage and assess impact of changes

Requirement Definition

- Condition or a capability a system must conform to
- Property that must be exhibited in order to solve some problem of the real world

"The system can…"

"The system shall…"

"The system must…"

Requirements Management Definition

- A systematic approach to eliciting, organizing, and documenting the requirements of the system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system
- Process of creating and maintaining requirements specifications based on stakeholder needs
- Includes requirements elicitation, analysis, specification and validation
- Sometimes refer to as requirements engineering

Requirements Activities in the Software Development Process

- Requirements activities start early in the software project lifecycle
- Continue throughout the project lifecycle
- Individual requirements are configuration items
- Needs to be tailored to the specifics of a project or organization

Requirements as Configuration Items

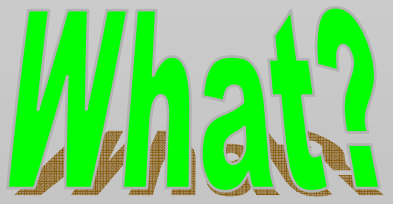
- Requirements are versioned
- Each requirement version is kept under version control
- Each time a requirement, its attributes or traceability changes, a new version is created
- Versions of requirements are baselined at project milestones

Requirements and Technology

- Except when the problem is motivated by technology, the problem is an artifact of the problem domain and is generally technology-neutral

Requirements Versus Design

- There is a fine line between requirements and design
- Requirements describe the **WHAT** while design describes the **HOW**
- Design-related requirements are captured as design constraints

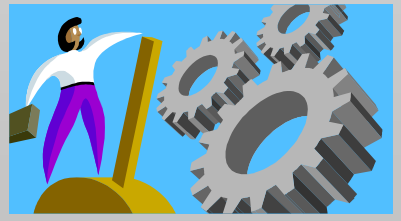


Requirements Categories

- Functional requirements
- Non-functional requirements
- Design constraints

Functional Requirements

- Describe the behavior of a system or the functions that the software is to execute
- Sometimes known as capabilities
- Can be expressed using declarative statements or use case modeling techniques
- Example
  - The ATM system shall first prompt the user for a debit card and then for a PIN before attempting to authenticate him or her.



Non-Functional Requirements

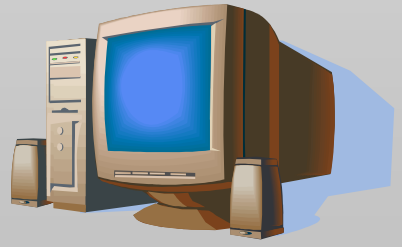
- Act to constrain the solution
- Sometimes knows as constraints or quality requirements
- Four categories (URPS+)
  - Usability
    - System ease of use
  - Reliability
    - Availability, MTBF, MTTR, Accuracy, etc.
  - Performance
    - Response time, throughput, capacity, degradation, etc.
  - Supportability
    - Ability of the software to be easily modified
  - Others

Non-Functional Requirements Examples

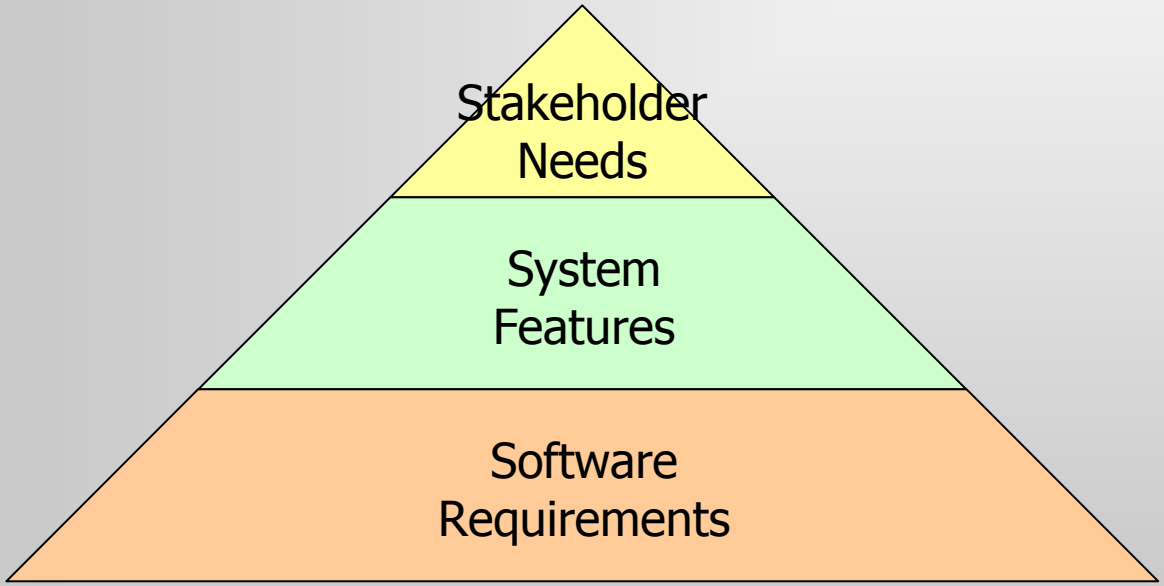
- Usability Requirement
  - The ATM keypad shall be composed of number and letter keys in brail.
- Reliability Requirement
  - The ATM system shall be operational 24 hours a day, 7 days a week except during regularly scheduled maintenance periods daily between 3am and 4am.
- Performance Requirement
  - The ATM system shall provide feedback to the user within 5 seconds after receiving a request.
- Supportability Requirement
  - The ATM system shall provide built-in self test to allow maintenance representatives to diagnose hardware problems.

Design Constraints

- Impose limitations on the design or the process use to build the system
- Allow for one design option unlike requirements that allow for more than one
- Example
  - The ATM system shall include a hardware backbone that is compatible with VME technology.
  - The ATM system shall be built with the VxWorks real-time operating system.



Requirements Hierarchy



Requirement Types

- Different requirement types for different stakeholders and to represent different levels of abstraction:
  - Stakeholder Needs
  - System Features
  - Software Requirements
    - Use Cases
    - Supplementary Requirements

Stakeholder Needs

- Captured to describe our understanding of the needs of the end-users and stakeholders
- For an ATM system, a stakeholder need may be:
  - Bank customers need to access their bank account outside normal banking hours

System Features

- Services that the system provides to fulfill one or more stakeholder needs
- For an ATM system, features may be:
  - Cash Withdrawal
  - Fund Deposit

Software Requirements

- Specifications of externally visible behavior of a system
- Two types
  - Functional expressed using use case modeling techniques or declarative statements
  - Non-functional expressed using declarative statements only

Use Cases

- Sequence of actions that yields a result of observable value to an actor (user of the system)
- Functional requirements only
- Expressed using use case modeling techniques
- For an ATM system, a use case may be:
  - Withdraw Cash

Supplementary Requirements

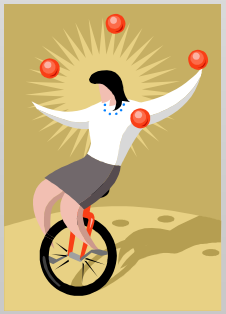
- Functional requirements not captured as use cases and non-functional requirements
- Expressed using declarative statements
- For an ATM system, supplementary requirements may be:
  - The ATM system shall provide cash to bank customers in US currency only.
  - The ATM system shall be available 24 hours a day and 7 days a week except during normal maintenance time between 1 am and 3 am on Monday mornings.

Requirements Attributes

- Ancillary information that helps manage and interpret requirements
- May include various classification dimensions of the requirement, verification method, acceptance test plan, summary rationale, source and change history
- Unique identifier is the most important attribute

Examples of Requirements Attributes

- Unique Identifier
  - Enables configuration control and management over the entire software life cycle
- Priority
  - Enables trade-offs in the face of finite resources
  - High, Medium, Low
- Status
  - Enables project progress to be monitored
  - Proposed, Approved, Incorporated, Validated



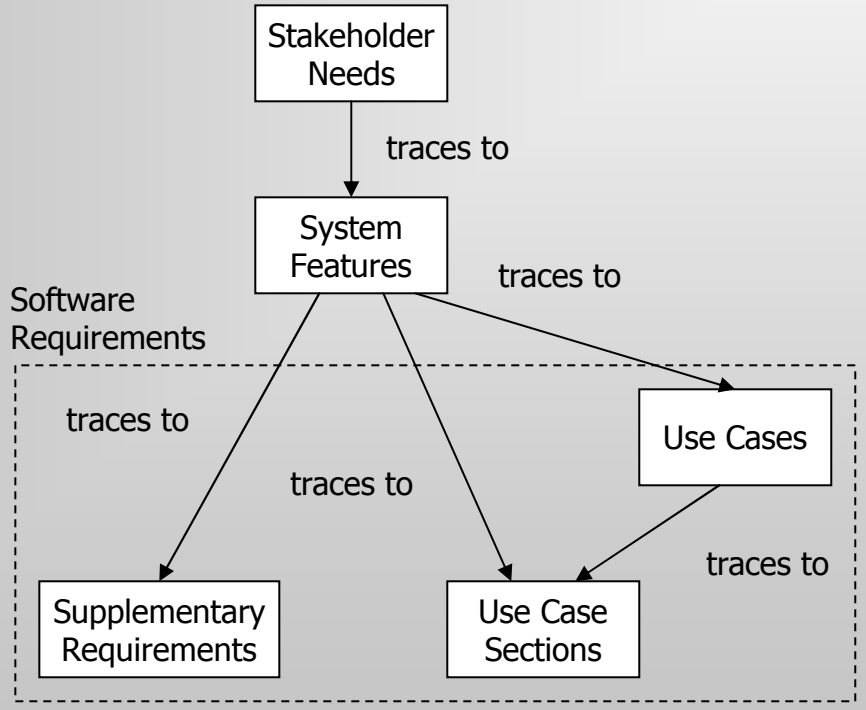
Traceability Definition

- The degree to which a relationship can be establish between two or more products of the development process

Requirements Traceability

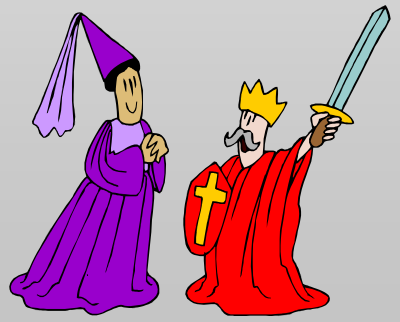
- Requirement should be traceable backwards to the requirements and stakeholders which motivated it
- Requirement should be traceable forwards into the requirements and design entities that satisfy it
- Fundamental to ensure coverage from one level of abstraction to the next and vice-versa
- Fundamental to perform impact analysis when requirements change

Requirements Traceability Example



Requirements Roles

- System Analyst
  - Responsible for all requirements
  - Owns the horizontal view (breath)
- Requirements Specifier
  - Responsible for the details of some requirements
  - Owns the vertical view (depth)



Requirements Artifacts

- System Analyst
  - Requirements Management Plan
  - Vision Document

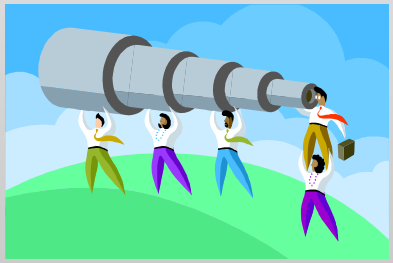
- System Analyst (Overall)/Requirements Specifier (Details)
  - Use-Case Model
  - Supplementary Specifications
  - Glossary

Requirements Management Plan

- Plan that defines how requirements are being managed on a project
- Includes
  - Requirement types
  - Requirements attributes
  - Traceability strategy
  - Baselining strategy

Vision Document

- Define the vision of the product to be developed in terms of needs and features
- Includes
  - Problem Statement
  - Product Positioning Statement
  - List of Stakeholders
  - List of Users
  - Stakeholder Needs
  - Product Features



System Feature

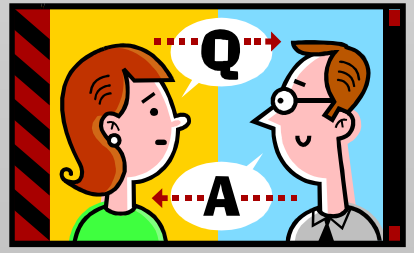
- A service the system provides to fulfill one or more stakeholder needs
- Easily expressed in natural language and consist of a short phrase
- Helpful construct for early product scope management
- A system of arbitrary complexity can be defined in a list of 25 to 99 features, with fewer than 50 preferred

ATM Example: System Features

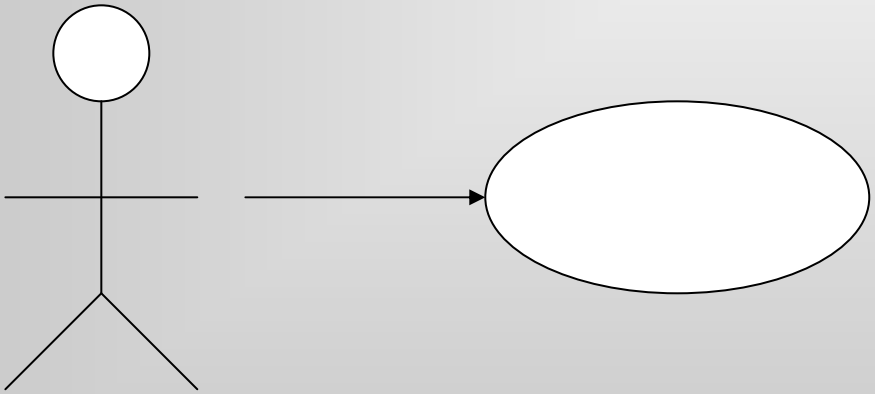
- Cash Withdrawal
  - The ATM system shall allow a bank customer to withdraw cash from one of his or her accounts in multiples of $20 up to a maximum of $1000 a day
- Funds Deposit
  - The ATM system shall allow a bank customer to deposit funds in paper format (cash, check or other except coins)
- Funds Transfer
  - The ATM system shall allow the bank customer to transfer funds from one of his or her account to another
- Bill Payment
  - The ATM system shall allow the bank customer to pay bills

Use-Case Model

- A model that describes a system's functional requirements in terms of use cases
- Each use case describes a dialog between actors and the system the use case models

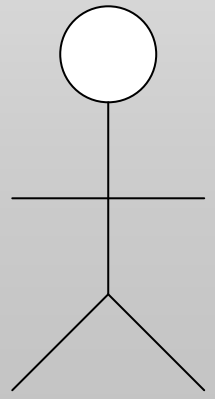


Use Case Model



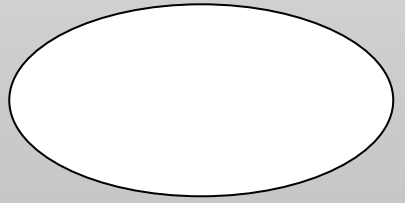
Actor

- Someone or something, outside the system, that interacts with the system
- Can represent a human being or a system
- UML™ representation of an actor is a stick figure
- Example
  - Customer

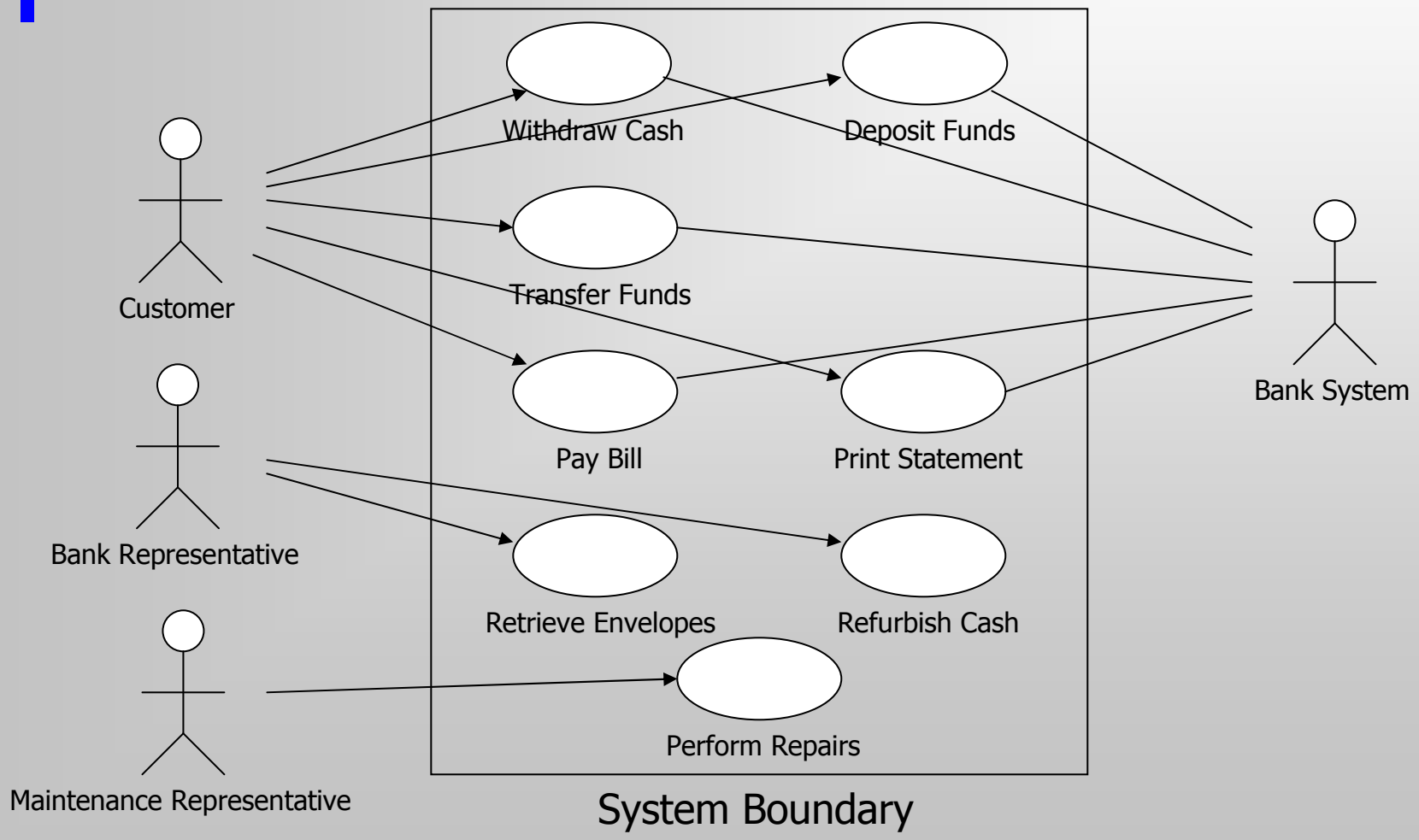


Use Case

- A sequence of actions that yields a result of observable value to an actor
- Represents one or more functional requirements
- UML™ representation of a use case is an oval
- Example
  - Withdraw Cash

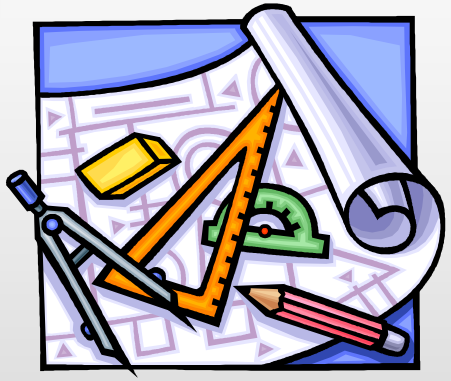


ATM Example: Use Case Model



Use Case Specification

- Name
  - Use case name
  - Verb followed by one or multiple nouns
- Brief description
  - Purpose of the use case
  - Summary description of the use case
- Flows of events/Scenarios
  - Basic and alternative series of steps
  - One basic or main flow/primary scenario
  - Zero, one or many alternative flows/secondary scenarios
- Preconditions
  - State of the system prior to the use case execution
- Postconditions
  - State of the system after the use case execution
- Special Requirements
  - Other requirements specific to the use case
- Extension Points
  - Points in the flows where the use case can be extended



ATM Example: Withdraw Cash Use Case Specification 1/2

- Name
  - Withdraw Cash
- Brief Description
  - This use case describes how the customer withdraws cash from the ATM system.
- Main Flow
  - See next slides.
- Alternative Flow 1
  - Account Balance Too Low - There is not enough money in the customer account to provide the customer with the requested amount.
- Alternative Flow 2
  - ATM System Balance Too Low - There is not enough money in the ATM system to provide the customer with the requested amount.

ATM Example: Withdraw Cash Use Case Specification 2/2

- Special Requirement 1
  - Currency - The system shall provide cash only in US Currency.
- Special Requirement 2
  - Currency Unit – The system shall provide cash amount in multiples of 20 dollar bills.
- Pre-Condition
  - The customer has at least one active account with the bank.
- Post-Condition
  - The amount withdrawn by the customer is subtracted from the customer account balance.
- Extension Point
  - None.

ATM Example: Withdraw Cash Main Flow 1/2

1. This use case starts when the customer is authenticated as described in use case "Authenticate User".
2. The system prompts the customer to select one of the following operations:
    1. Withdraw Cash;
    2. Deposit Funds;
    3. Transfer Funds;
    4. Pay Bills; and
    5. Print Statement.
3. The customer selects the withdraw cash option.
4. The system prompts the customer to select one of the following account:
    1. Checking Account;
    2. Savings Account; and
    3. Credit Margin Account.

ATM Example: Withdraw Cash Main Flow 2/2

5. The customer selects an account.
6. The system prompts the customer to enter an amount.
7. The customer enters an amount.
8. The system prompts the customer if he or she wants to perform another operation.
9. The customer selects not to perform another operation.
10. The system returns the card to the customer.
11. The customer takes the card.
12. The system dispenses cash to the customer.
13. The customer takes cash.
14. The system prints a receipt.
15. The customer takes the receipt.
16. This use case ends.

Supplementary Specifications

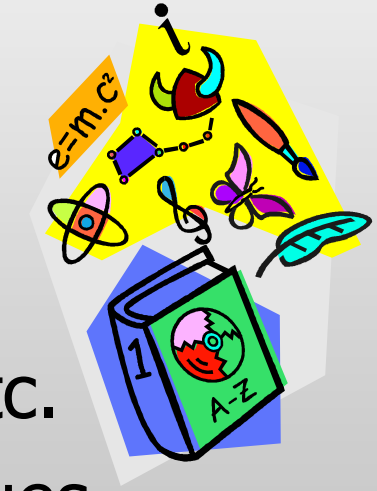
- Describes functional and non-functional requirements applicable to the entire system (not only to a single use case)
- Functional and non-functional requirements specific to a use case are described in the special requirements section of the use case specification

ATM Example: Supplementary Specification

- Functionality
  - The ATM system shall request that the customer provides an amount and an account before dispensing cash.
- Usability
  - The ATM keypad shall be composed of number and letter keys in brail.
- Reliability
  - The ATM system shall be operational 24 hours a day, 7 days a week except during regularly scheduled maintenance periods daily between 3am and 4am.
- Performance
  - The ATM system shall provide feedback to the user within 5 seconds after receiving a request.
- Supportability
  - The ATM system shall provide built-in self test to allow maintenance representatives to diagnose hardware problems.
- Others
  - The ATM system shall accept alphanumeric password of between 4 and 8 characters (security).
- Design Constraints
  - The ATM system shall include a hardware backbone that is compatible with VME technology.
  - The ATM system shall be built with the VxWorks real-time operating system.

Glossary

- Defines the domain-specific terms to support other requirements artifacts
- For each term
  - Name
  - Description
  - Type: Numeric, String, Boolean, etc.
  - List of valid values or range of values
  - Default value

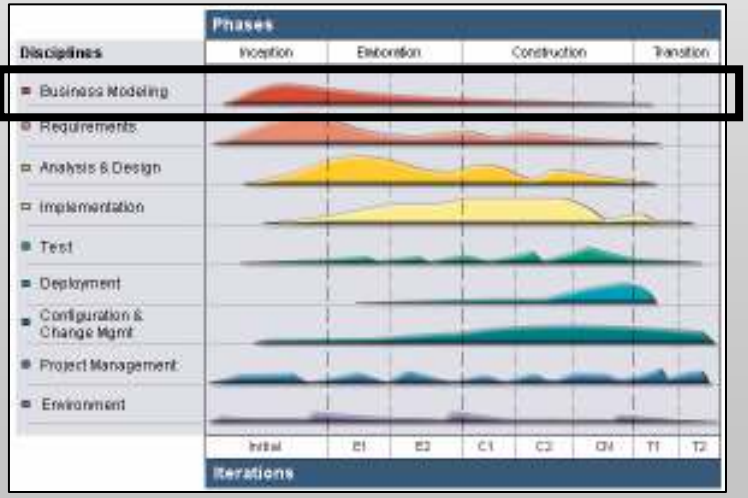


ATM Example: Glossary

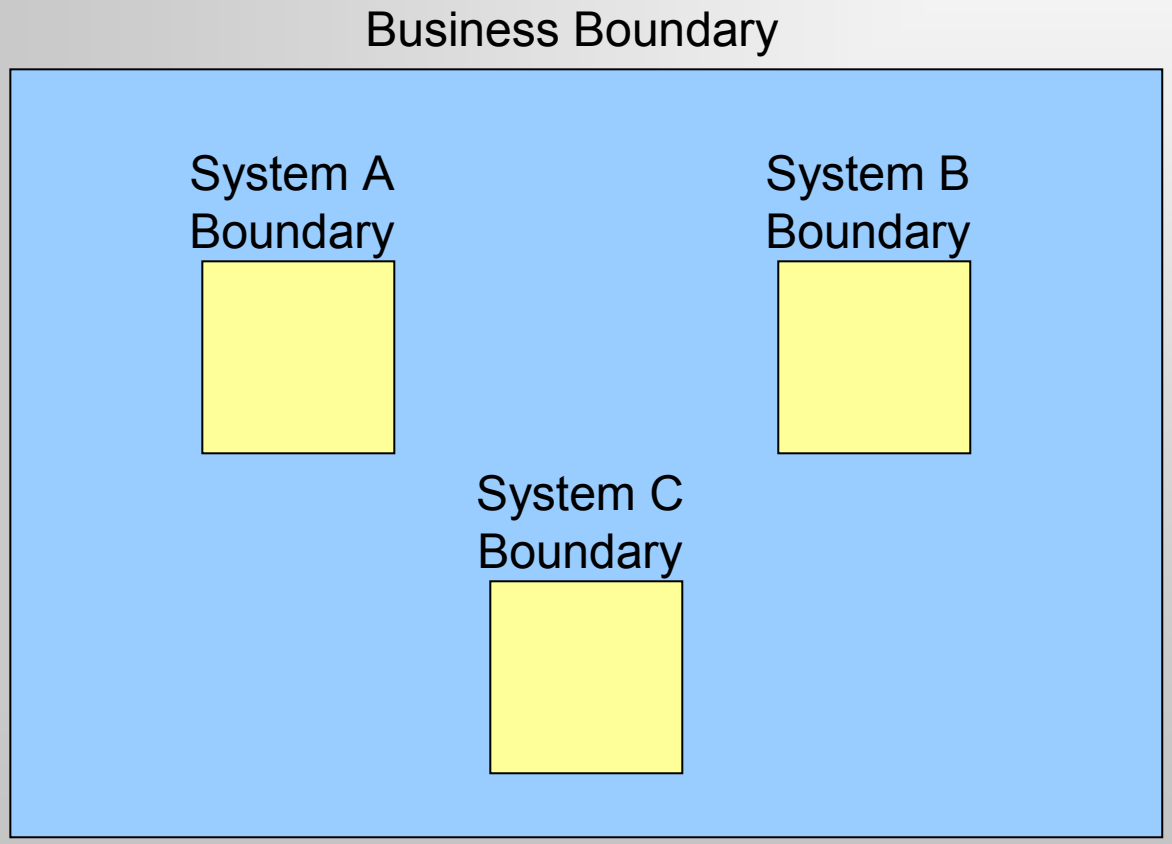
- ATM
  - Automated Teller Machine
- PIN
  - Name: PIN - Personal Identification Number
  - Description: Identifier used to authenticate user of the ATM with his/her debit card
  - Type: Numeric
  - Range of Values: 0000 – 9999
  - Default Value: 1234
- Account
  - Placeholder for funds with which transactions can be performed

Business Modeling Discipline

- Topics
  - Scope of Business Modeling
  - Software Development and Business Modeling Mapping
  - Roles
    - Business Process Analyst
    - Business Architect
    - Business Designer
  - Artifacts
    - Business Goal
    - Business Vision
    - Business Use Case Model
    - Supplementary Business Specification
    - Business Rule
    - Business Glossary
    - Business Architecture Document
    - Business Analysis Model
    - Business Design Model
    - Business Deployment Model



Scope of Business Modeling



Software Development and Business Modeling Mapping

Software Development	Business Modeling
Vision	Business Vision
Use-Case Model	Business Use Case Model
Supplementary Specification	Supplementary Business Specification
Glossary	Business Glossary
Software Requirement	Business Rule
Software Architecture Document	Business Architecture Document
Analysis Model	Business Analysis Model
Design Model	Business Design Model
Deployment Model	Business Deployment Model

Business Modeling Roles

- Business Process Analyst
  - Responsible for eliciting and managing the overall business requirements
- Business Architect
  - Responsible for defining the overall business architecture
- Business Process Designer
  - Responsible for defining specific elements of business requirements and business architecture

Business Modeling Artifacts

- Business Process Analyst
  - Business Goal
  - Business Vision
  - Supplementary Business Specification
  - Business Rule
  - Business Glossary
- Business Process Analyst (Overall)/Business Designer (Details)
  - Business Use Case Model
- Business Architect
  - Business Architecture Document
- Business Architect (Overall)/Business Designer (Details)
  - Business Analysis Model
  - Business Design Model
  - Business Deployment Model

Business Goal

- Requirement that must be satisfied by the business
- Purpose is to translate business strategy into measurable steps with which the operations can be steered in the right direction and improved if necessary
- Examples
  - Financial measures
    - Revenues, profits, cost
  - Non-financial measures
    - Employee satisfaction, customer success

Business Strategy Versus Business Goal

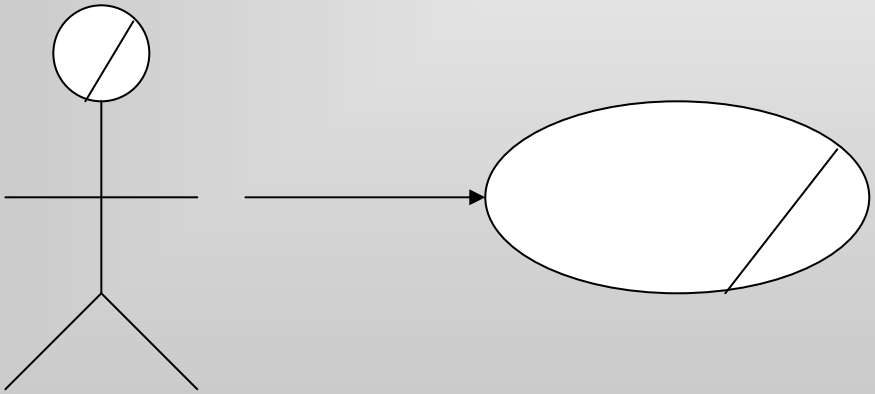
- Business Strategy
  - Describes the desired competitive position of the organization
  - Strategic
  - Focused on the what not the how
- Business Goal
  - Describe what must be achieved to reach that desired competitive position
  - Tactical
  - Focused on the what not the how

Business Vision

- Describes the business goals and objectives that justify the business modeling effort
- Includes
  - Problem Statement
  - Business Positioning Statement
  - List of Stakeholders
  - List of Customers
  - Key Stakeholders or Customer Needs

Business Use Case Model

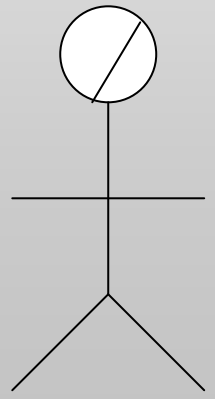
- Models the intended functions of the business



Business Actor Business Use Case

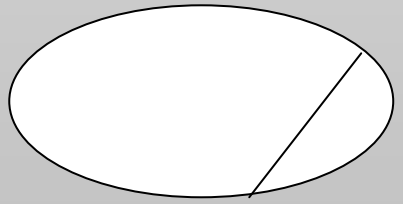
Business Actor

- Someone or something, outside the business, that interacts with the business
- Can represent a human being or another business
- UML™ representation of a business actor is an actor with the <<business actors>> stereotype
- Example
  - Supplier

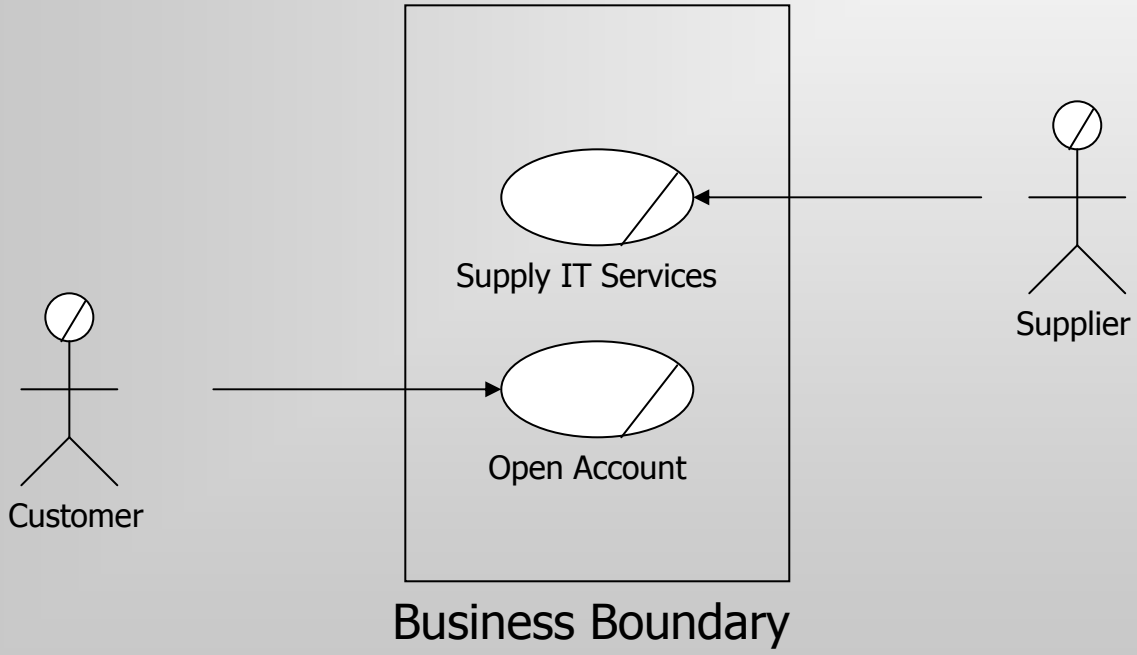


Business Use Case

- A sequence of actions that a business performs that yields a result of observable value to a business actor
- Represents one or more functions of the business
- UML™ representation of a business use case is a use case with the <<business use case>> stereotype
- Example
  - Supply IT Services



Bank Example: Business Use Case Model

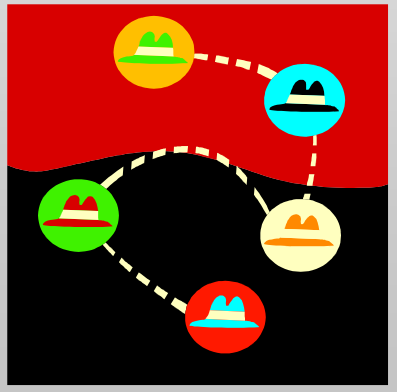


Supplementary Business Specification

- Describes quantifiers (or constraints or restrictions) applicable to the entire business (not only to a single business use case)
- Quantifiers specific to a business use case are described in the special requirements section of the business use case specification

Business Rule

- Describes a condition that must be satisfied
- Requirement on how the business must operate
- Examples
  - Laws and regulations
  - Business-specific policy

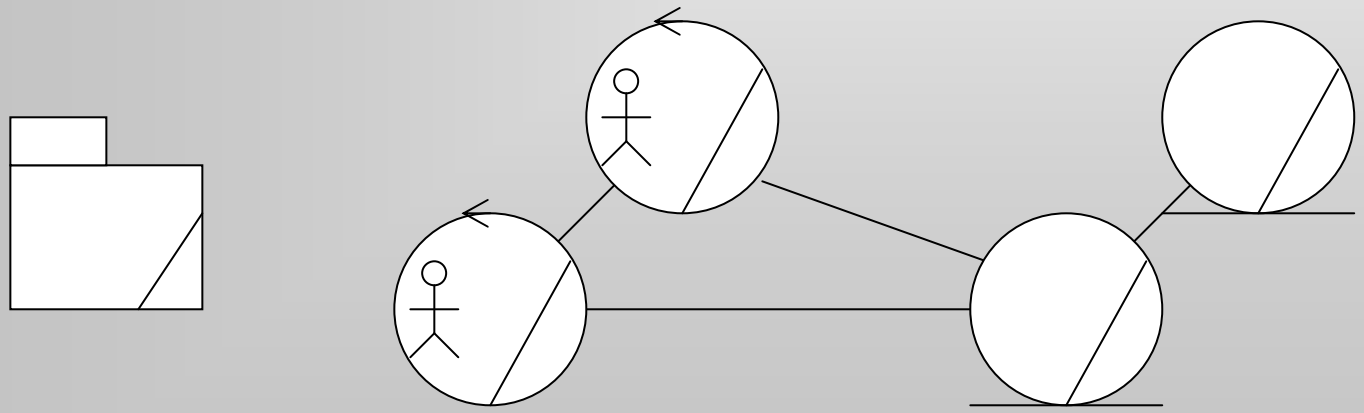


Business Glossary

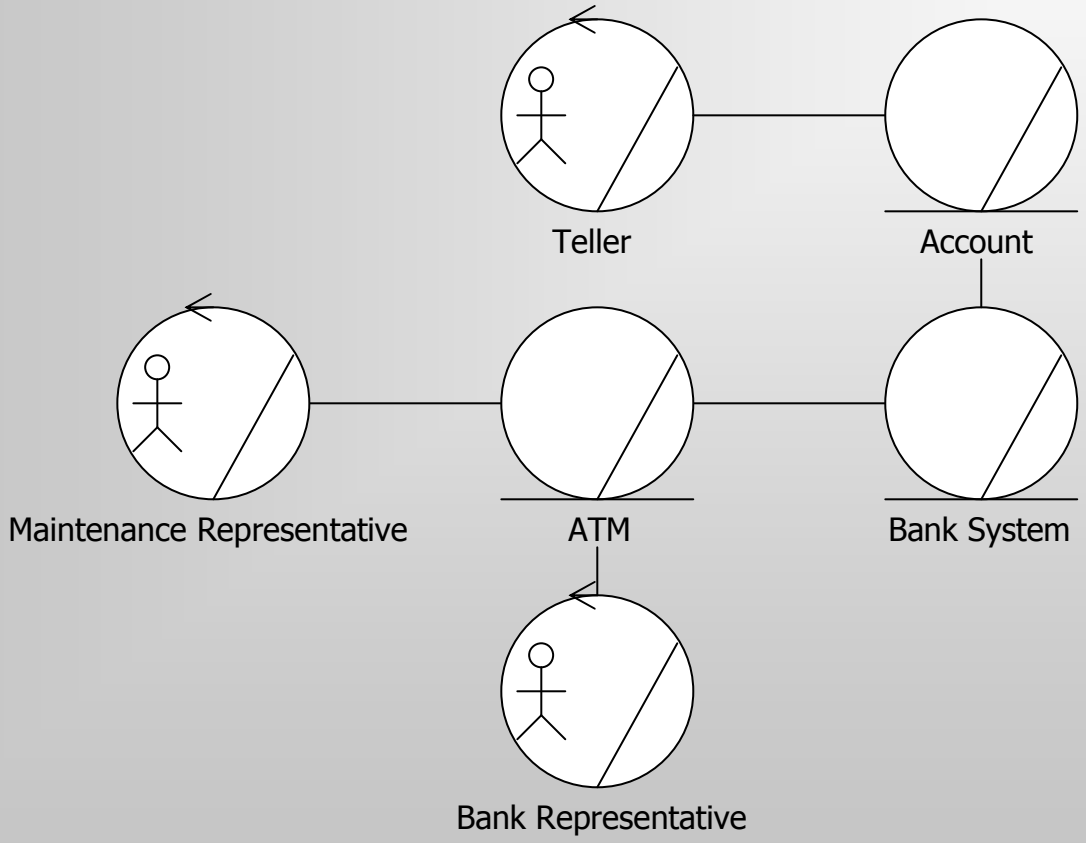
- Defines terms specific to the business being modeled

Business Analysis Model

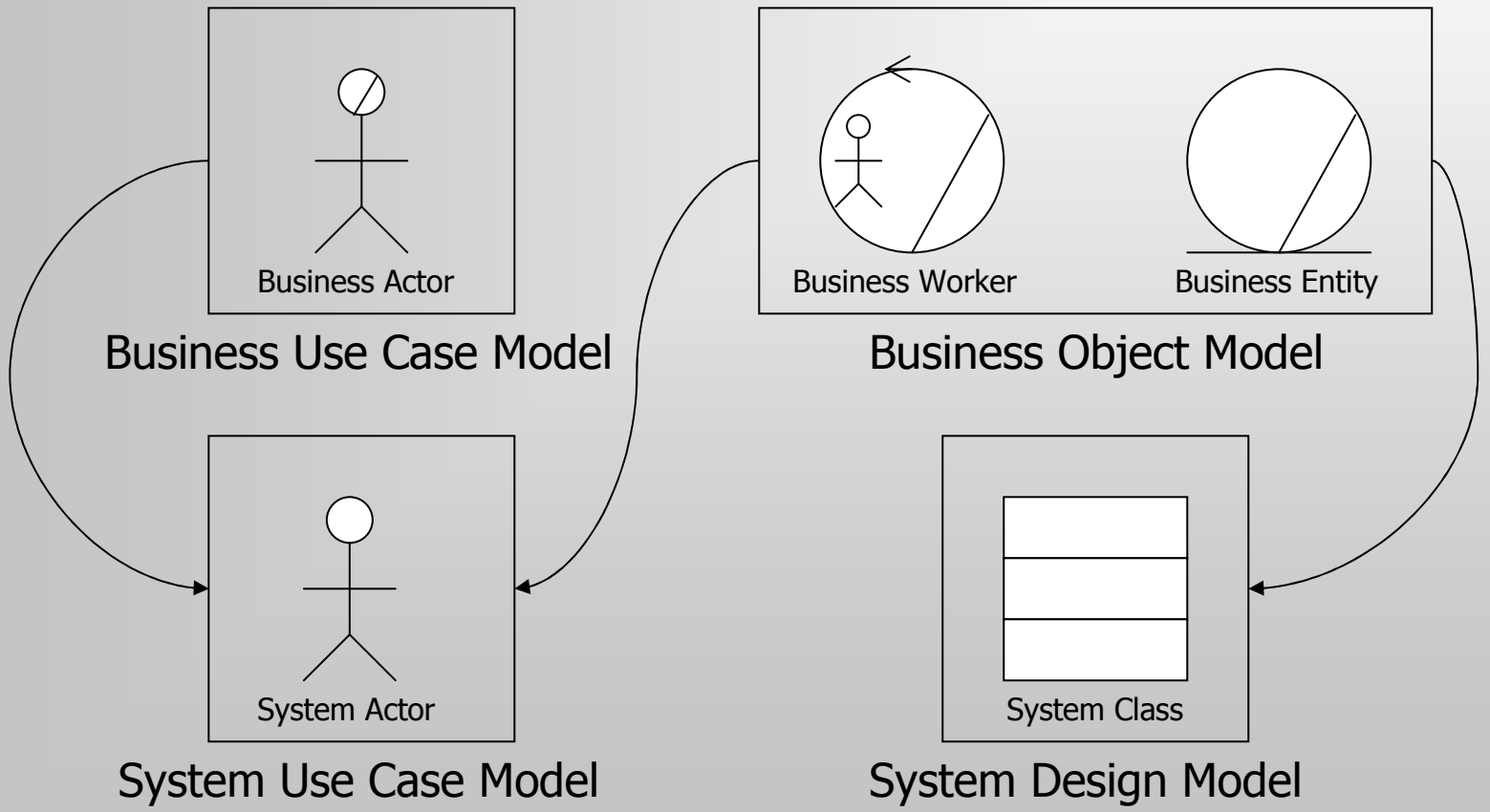
- Describes the realization of business use cases in terms of interacting business system, business workers and business entities



Bank Example: Business Analysis Model



From Business Models to System Models



3. Case Studies

- Topics
  - Innovapost
  - National Bank
  - Canoe

Innovapost (2004 to 2006)

- Goal
  - To improve software process maturity (CMMI) across all projects in the organization
- Requirements development process
  - Based on ADM, RUP (Requirements discipline) and CMMI
- Requirement types
  - Stakeholder needs → Business requirements
  - System features → High-Level requirements
  - Software requirements → Detailed requirements
- Tools
  - Requirements Management → Borland CaliberRM
  - Visual Modeling → Microsoft Visio and others
  - Report Generation → None

National Bank (2004 to 2005)

- Goal
  - To put together a set of best practices, templates and tools for a newly formed business analyst team
- Business analysis process
  - Based on Cadre d'Execution de projet and RUP (Requirements and Business Modeling disciplines)
- Artifacts
  - Vision → Manuel des besoins d'affaires (MBA)
  - Software requirements specification (SRS) → Spécifications d'affaires et de systèmes (SA)
- Tools
  - Requirements Management → IBM Rational RequisitePro
  - Visual Modeling → IBM Rational Software Modeler (RSM)
  - Report Generation → IBM Rational SoDA

Canoe (2007 to today)

- Goal
  - To standardize practices of the functional analyst team
- Functional analysis process
  - Based on RUP (Requirements discipline and some elements of the Business Modeling and Analysis & Design disciplines)
- Deliverables
  - Vision → Vision
  - Use Case Specification → Functional Analysis document
- Tools
  - Requirements Management → None
  - Visual Modeling → Microsoft Visio
  - Report Generation → None

Xelaration's Mission

Xelaration

■■■ Software Corporation ■■■

"To accelerate the success of our customers
in improving their software engineering capabilities
enabling them to achieve their business goals"