104KM Enterprise Information System

Topic: OO Analysis and Design – Part 1

Learning outcomes for today

- Object-oriented System Development Life Cycle
- Rational Unified Process (RUP)
- Understanding needs of stakeholders and working with project teams when developing IT solutions
- Unified Modelling Language (UML)

Software Development Process

What is a System Development Methodology?

Why are methodologies used?

What is a System Development Methodology?

Any logical process used by a ***System Analyst*** *to develop an* ***Information System*****.**

The methodology results in a high quality system that:

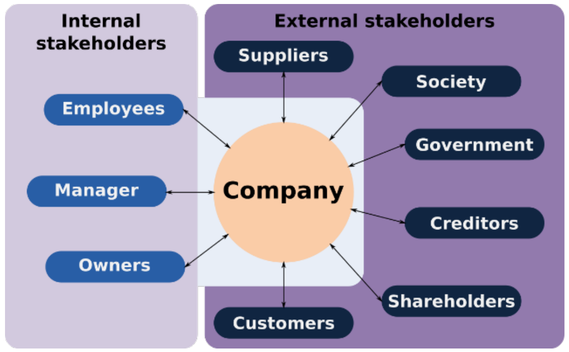
- *meets or exceeds customer expectations*
- *within time and cost estimates*
- *works effectively and efficiently in the current and planned Information Technology infrastructure*
- *cheap to maintain and cost-effective to enhance*

The need for different approaches…

Different methodologies place emphasis on different aspects of the system development process.

Issues to consider

- Who are the stakeholders?
- What type of system is being developed?
- What are the principal features of the environment in which the systems is to be developed? (*Most difficult to address*)

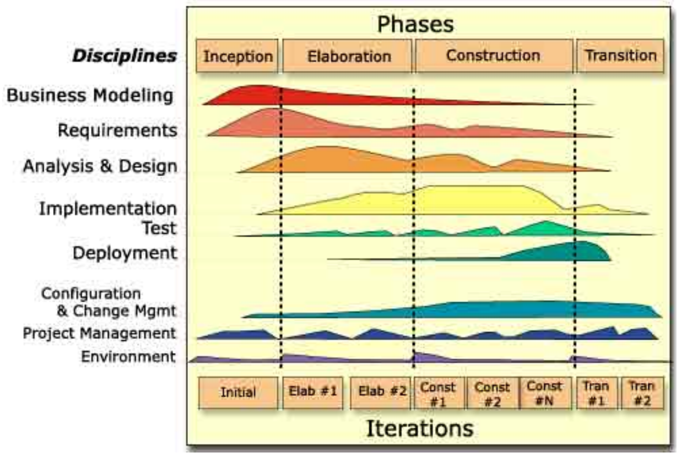


RUP

- Rational Unified Process is a

  → unified software development process

- Developed by IBM encompassing **Information Systems** development best practices.



RUP

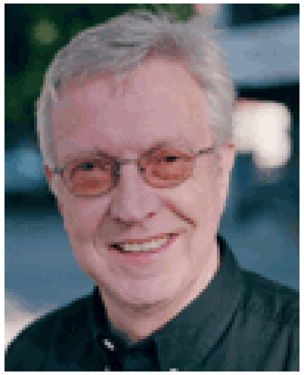
- An engineering process that provides a disciplined approach to assigning and managing tasks and responsibilities to developing software systems.

- An object-oriented systems development methodology.

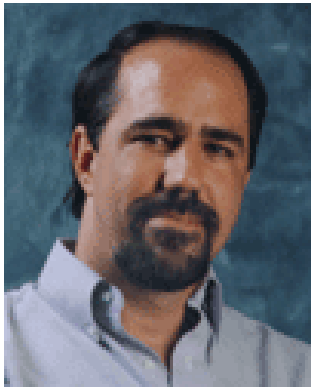
- Uses UML for visual notation and provides guidelines in how to use the UML effectively.

UML - Developer

Software Engineers best known for developing the UML.


Ivar Jacobson

James Rumbaugh


Grady Booch

RUP

- RUP is a method of managing OO Software Development.
- It can be viewed as a **Software Development Framework** which is extensible and features:

→ Iterative and incremental development

→ Component based development

→ Requirements driven development

→ Configurability

→ Architecture centrism

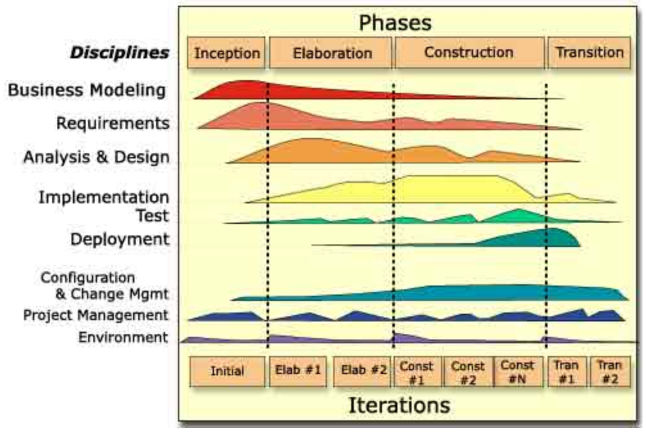
→ Visual modelling techniques

*<< **Progress is measured against clear milestones** >>*

RUP - Phases

RUP establishes 4 phases of development. Each phases is organized into a number of separate iterations.

- **Inception** – scope of the project
- **Elaboration** – requirements of product being built
- **Construction** – software is developed in this phase
- **Transition** – software is rolled out to customers



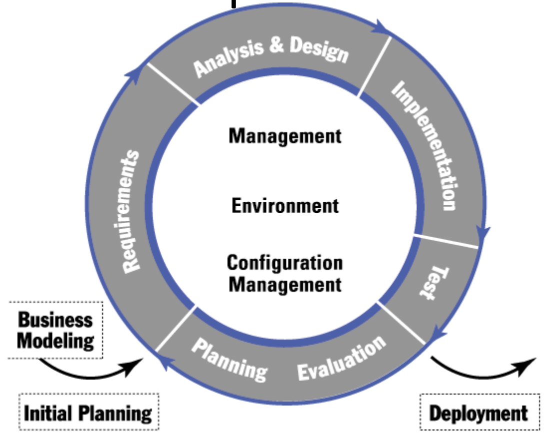
RUP - Project Structure

The RUP project structure is in two dimensions:

- The horizontal axis represents time and show the lifecycle aspects of the process as it unfolds.
- The vertical axis represents disciplines, which group activities logically by nature.

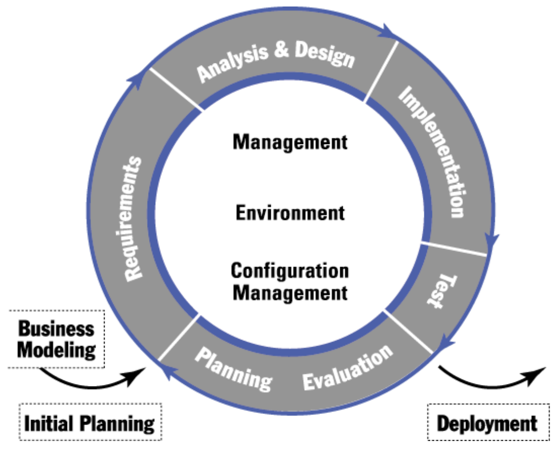
An Iteration Development Process

- Recognizes the reality of changing requirements
- Promotes early risk mitigation, by breaking down the system into mini projects and focusing on the riskier elements first
- Allows you to "*plan a little, design a little and code a little*"
- Encourages all participants, including testers, technical developers to be involved at the early stage
- Allows the process itself to modulate with each iteration, allowing you to correct errors sooner and put into practice lessons learned in the prior iteration
- Focuses on components architecture



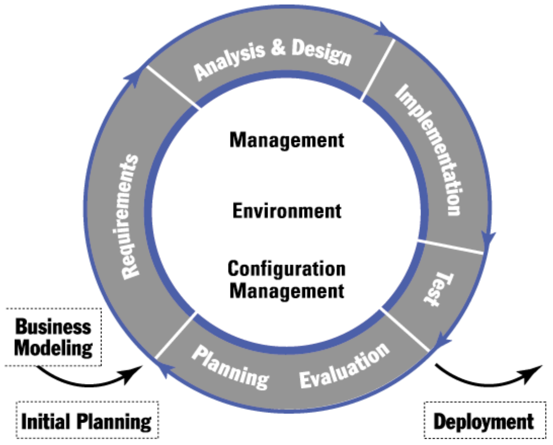
Business Modeling

- Assess business status
- Describe current business
- Identify business processes
- Refine business processes definitions
- Design business process realizations
- Refine roles and responsibilities
- Explore process automation
- Develop a domain model



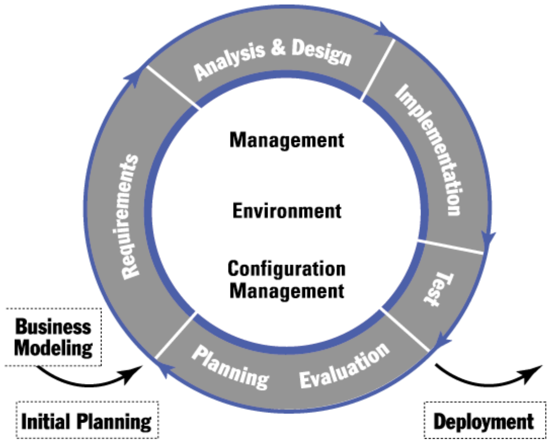
Requirements *(Inception Phase)*

- Analyze the problem
- Understand stakeholders needs
- Define the system
- Manage the scope of the system
- Refine the system definition
- Manage changing requirements



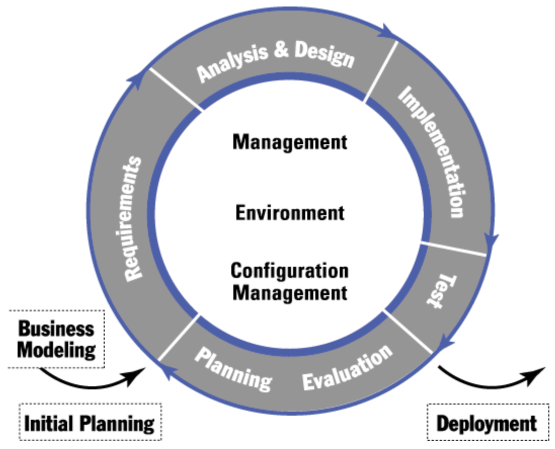
Analysis and Design *(Elaboration Phase)*

- Define candidate architecture
- Perform architecture synthesis
- Analyze behavior
- Design components
- Refine the architecture
- Design the database



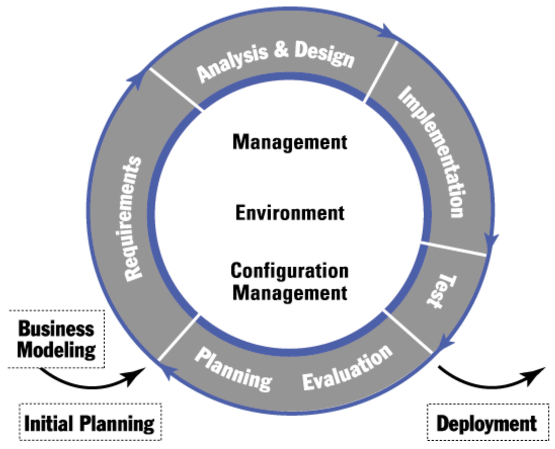
Implementation *(Construction Phase)*

- Structure the implementation model
- Plan the integration
- Implement components
- Integrate each sub system
- Integrate the system



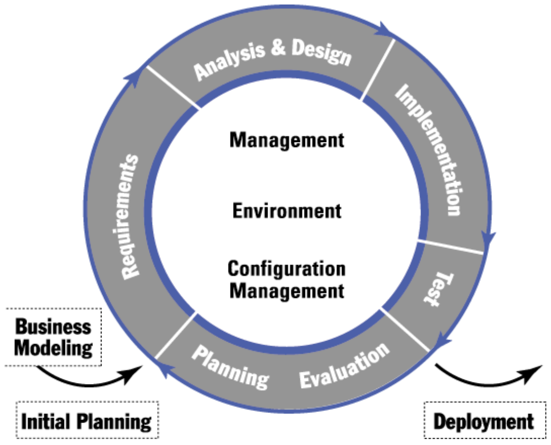
Test

- Define evaluation mission
- Verify test approach
- Validate build stability
- Test and evaluate
- Achieve acceptable mission
- Improve test assets



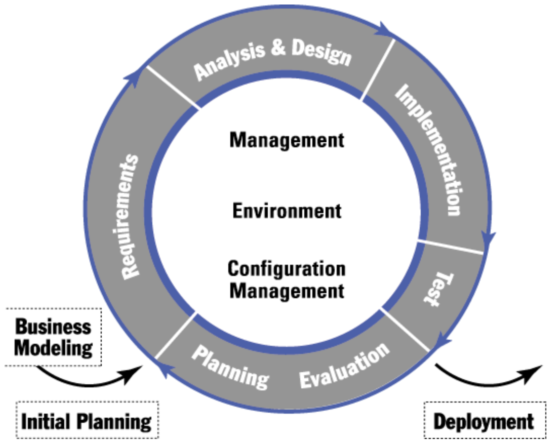
Configuration and Change Management (CM)

- Plan project configuration and change control
- Create project CM environments
- Change and deliver configuration items
- Manage baselines and releases
- Monitor and report configuration status
- Manage change requests



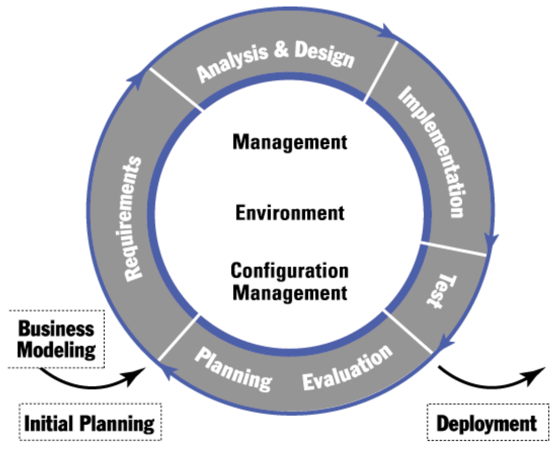
Deployment *(Transition Phase)*

- Plan deployment
- Develop support material
- Manage acceptance test
- Product unit
- Package product
- Provide access to download site
- Beta test product



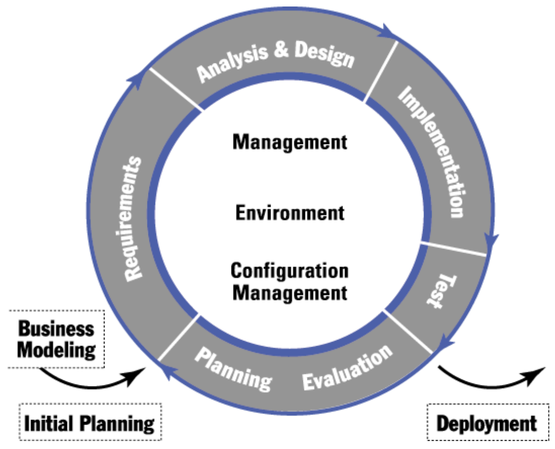
Project Management

- Conceive new project
- Evaluate project scope and risk
- Develop software development plan
- Monitor and control project
- Plan for next iteration
- Manage iteration
- Close-out phase
- Close-out project

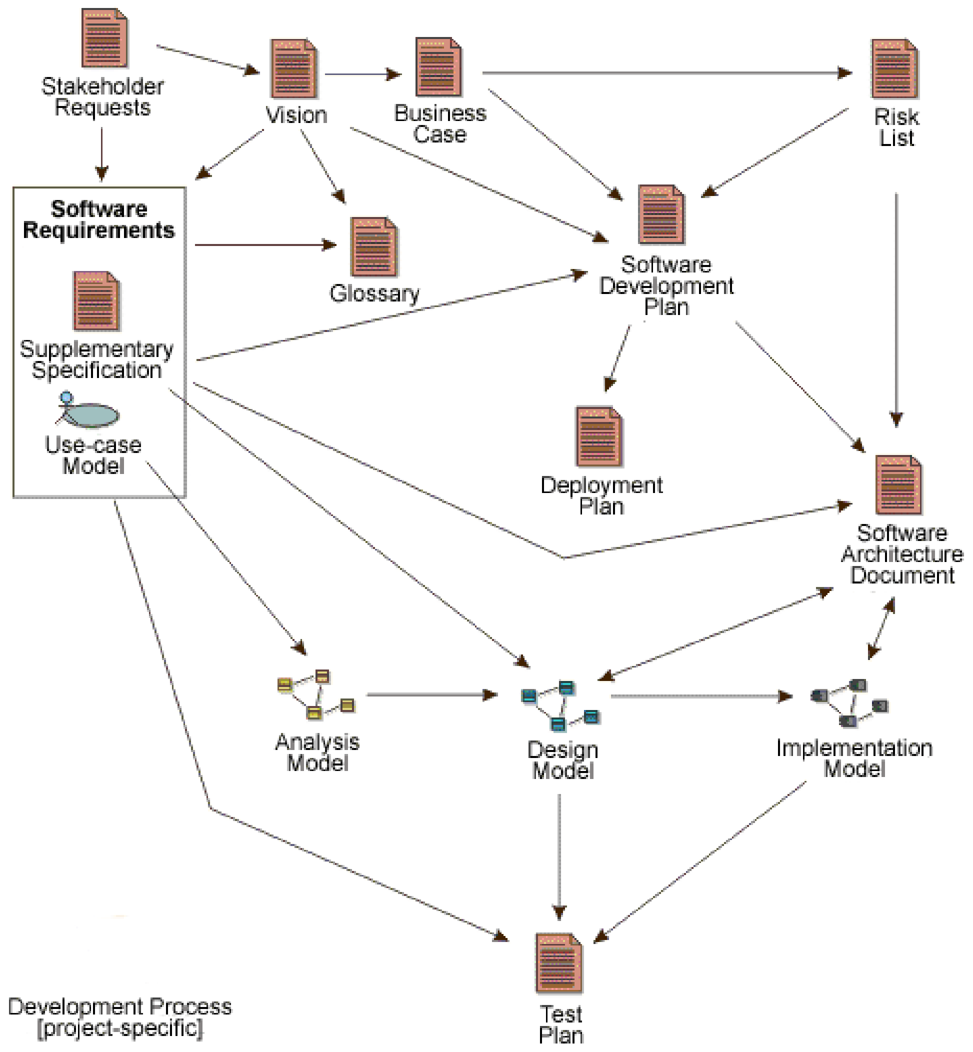


Environment

- Prepare environment for project
- Prepare environment for an iteration
- Prepare guidelines for an iteration
- Support environment during an iteration



Major Artifacts of RUP



Summary RUP

- *RUP is a method of managing OO Software Development Process*

- *Graphical UML*

- *RUP can be viewed as an iterative and incremental development process*

  (incl. iteration and milestones)

Object-oriented SDLC (OOSDLC) Methodologies

Approach

Focuses on improving, maintaining and reusability of software systems through a set of techniques, notations, tools and criteria.

Activities

Conceptualization
OO analysis and modeling
OO design
Implementation
Maintenance

Object-oriented SDLC (OOSDLC) Methodologies

Conceptualization

To establish the vision and core requirements of the software system to be developed.

Object-oriented analysis and modeling

To build models of the system's desired behavior, using notations such as the Unified Modeling Language (UML).

To capture the essential relevant aspects of the real world and to define the services to be provided and/or the problems to be solved.

To simplify reality to better understand the system to be developed.

Object-oriented SDLC (OOSDLC) Methodologies

Object-oriented design

To create an architecture for implementation,

Represents in terms of objects and classes and the relationship among them.

Implementation

To implement the design by using and object-oriented programming language (e.g. Java).

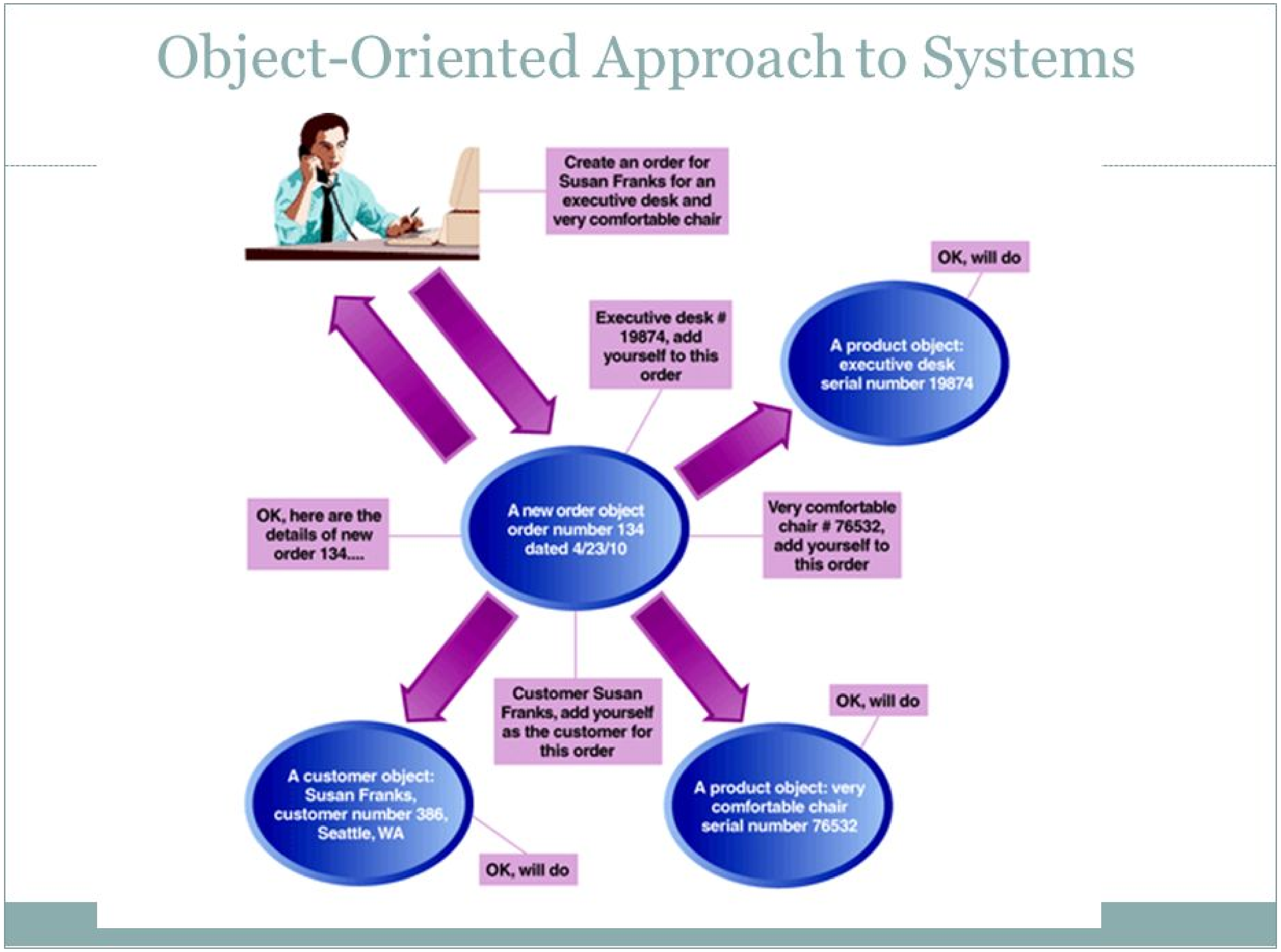
Maintenance

To manage post delivery evolution effectively.

Object-oriented SDLC (OOSDLC) Methodologies

- Completely different approach to Information Systems
- Views information system as collection of interacting **objects** that work together to accomplish tasks
- Objects – things in computer system that can respond to messages
- Conceptually, no processes, programs, data entities or files are defined – just objects

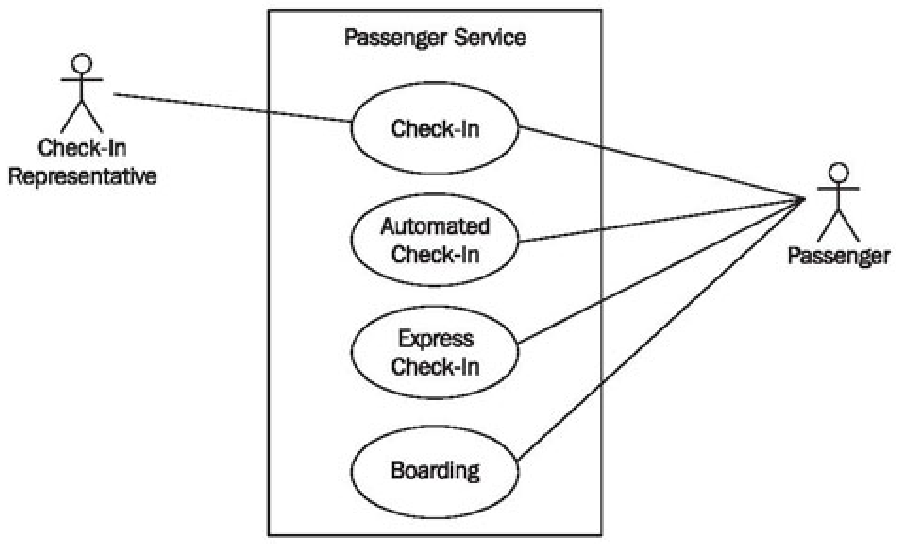
Example

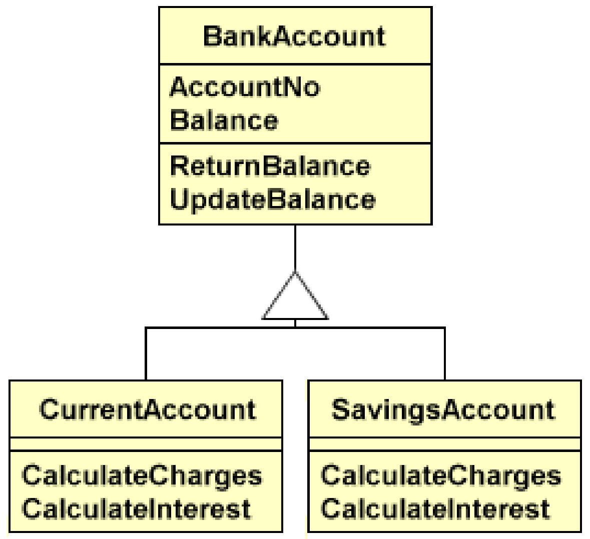

Object-Oriented Approach to Systems
Create an order for Susan Franks for an executive desk and very comfortable chair
OK, will do
Executive desk # 19874, add yourself to this order
A product object: executive desk serial number 19874
OK, here are the details of new order 134....
A new order object order number 134 dated 4/23/10
Very comfortable chair # 76532, add yourself to this order
Customer Susan Franks, add yourself as the customer for this order
OK, will do
A customer object: Susan Franks, customer number 386, Seattle, WA
OK, will do
A product object: very comfortable chair serial number 76532

Object-Oriented Approach to Systems

- **Object-oriented analysis (OOA)**
  - → *Defines types of objects users deal with*
  - → *Shows use cases are required to complete tasks*
- **Object-oriented design (OOD)**
  - → *Defines object types needed to communicate with people and devices in system*
  - → *Refines each type of object for implementation with specific language of environment*
- **Object-oriented programming (OOP)**
  - → *Writing statements in programming language to define what each type of object does*

Examples: Use Case Diagram and Class Diagram





Computer Aided System Engineering (CASE) Tools

- CASE tools are software tools designed to help **Systems Analyst** to complete development tasks.
- The CASE tool contains a database of information called a repository.
- The repository stores information about the system, including models, descriptions and references that link the various model together.
- Information stored in repository can be used in a variety of ways by the development team.

Computer Aided System Engineering (CASE) Tools

- Every time a team member adds information about the system, it is immediately available for everyone else.

- CASE tools can check the models to make sure they are complete and follow the correct diagramming rules.

- CASE tools can check one model against another to make sure they are consistent.

CASE Tools - Examples

- Microsoft Visio
- Rational Rose
- **StarUML**

Rational Rose

StarUML™
The Open Source UML/MDA Platform

Note:

The above are drawing tools suitable for any system model comes with a collection of drawing templates (including symbols used in a variety of business and engineering applications: flowcharts, DFDs, ERDs, UML diagrams).

- Oracle Designer *(A tool set for recording definitions and automating the rapid constructions of flexible, graphical client-server applications)*
- C++, JAVA – OO Programming Languages

Next Week

UML