Small notes on Object Oriented Analysis and Design

What is a System?

- A set or arrangement of entities so related or connected so as to form a unity or organic whole. (Iberall)
- A set of objects and relationships among the objects viewed as a whole and designed to achieve a purpose

What is subsystem?

- A subsystem is simply a system within a system**.**
  - Automobile is a system composed of subsystems:
    - Engine system, Body system, Frame system
  - Each of this subsystem is composed of sub-sub -systems.
    - Engine system: carburetor system, generator system, fuel system, and so on

Bad Systems

- Fail to meet requirements
- Poor performance
- Poor reliability
- Lack of usability
- Example difficulties:
  - Not to schedule
  - Not to budget
  - Runaway = 100% over budget or schedule
- Some problems are simply "wicked" problems

Reasons for Failure

- Complexity
  - Shifting requirements
  - Bad estimation
  - Bad management
  - New technology
- Must tackle complexity by, for example:
  - Structure partitioning of problem
  - Organized interaction of parts
  - Ensure you achieve the task
- Systems are subject to the need for continuing change

Important System Concepts

Decomposition

- The process of breaking down a system into smaller components
- Allows the systems analyst to:
  - Break a system into small, manageable subsystems
  - Focus on one area at a time
  - Concentrate on component pertinent to one group of users
  - Build different components at independent times

Modularity

  - Process of dividing a system into modules of a relatively uniform size
  - Modules simplify system design
- **Coupling**
  - Subsystems that are dependent upon each other are coupled
- **Cohesion**
  - Extent to which a subsystem performs a single function

What is an Information Systems?

- Interrelated components working together to
  - Collect
  - Process
  - Store
  - Disseminate information
- To support decision making, coordination, control, analysis and visualization in an organization

Information System Types

- Transaction Processing Systems (TPS)
- Management Information Systems (MIS)
- Decision Support Systems (DSS)
- Expert System and Artificial Intelligence (ES &AI)

Systems Analysts

- Systems analysts are the key individuals in the systems development process.
- A systems analyst studies the problems and needs of an organization to determine how people, data, processes, communications, and information technology can best accomplish improvements for the business.
- The organizational role most responsible for the analysis and design of information systems.

Skills of a Successful Systems Analyst

- **Analytical skills**
  - Understanding of organizations.
  - Problem solving skills
  - System thinking
    - Ability to see organizations and information systems as systems
- **Technical skills**
  - Understanding of potential and limitations of technology.
- **Managerial skills**
  - Ability to manage projects, resources, risk and change
- **Interpersonal skills**
  - Effective written and oral communication skills

- Help you work with end user as well as other system analysts and programmers

The analyst is responsible for:

- The efficient capture of data from its business source,
- The flow of that data to the computer,
- The processing and storage of that data by the computer, and
- The flow of useful and timely information back to the business and its people.

Skills Required by Systems Analysts

- Working knowledge of information technology
- Computer programming experience and expertise
- General business knowledge
- Problem-solving skills
- Interpersonal communication skills
- Interpersonal relations skills
- Flexibility and adaptability
- Character and ethics
- Systems analysis and design skills

System Analysis and Design (SAD)

Systems Analysis: understanding and specifying in detail <u>what</u> an information system should do

System Design: specifying in detail <u>how</u> the parts of an information system should be implemented

- Definition of SAD:
  - The complex organizational process whereby computer-based information systems are developed and maintained.
- Analysis: defining the problem
  - From requirements to specification
- Design: solving the problem
  - From specification to implementation

Views of Systems Analysis

- How to build information systems
- How to analysis information system needs
- How to design computer based information systems
- How to solve systems problems in organizations

System development methodology

- A standard process followed in an organization to conduct all the steps necessary to:
- Analyze
- Design

- Implement
- Maintain information system

Systems Development Life Cycle (SDLC)

- It is a common methodology for systems often follows for system development in much organization, featuring several phases that mark the progress of the systems analysis and design effort.

- **SDLC phases:**
  - 1-Project identification and selection
  - 2-Project initiation and planning
  - 3-Analysis
  - 4-Design
    - 4.1Logical design
    - 4.2Physical design
  - 5-Implementation
  - 6-Maintenance

1- Structured analysis and structured design

- More focus on reducing maintenances and time effort in system development
- Integrate change when needed

2- Object **O**riented **A**nalysis and **D**esign (**OOAD**)

- **A** more recent approach to system development that is becoming is object oriented analysis and design (OOAD).
- It is often called third approach to system development, after the process oriented and data oriented approaches
- **Definition: OOAD**
  - **It systems development methodologies and techniques base on objects rather than data or process**

What is Object-Oriented Analysis

Object-Oriented Analysis is a method of Analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain. (Booch)

What is Object-Oriented Design

Object-Oriented Design is a method of design encompassing the process of Object-Oriented decomposition and a notation for depicting logical and physical as well as static and dynamic models of the system under design. (Booch)

Two important Parts of the above definition:

1. leads to an Object-Oriented Decomposition and
2. Uses different Notations to express different models of the Logical (Class and Object structure) and Physical (Module and Process Architecture) Design of a system, in addition to the Static and Dynamic aspects of the System.

What is Object-Oriented Programming

a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships. (Booch)

UML – Definition

Unified Modelling Language
“UML is a language for specifying, visualizing, constructing, and documenting the artefacts of software systems, as well as for business modelling and other non-software systems."

The Object Management Group adopted UML in 1997 as a standard.

Documentation

Every engineering discipline has a standard method of documentation

- Architects have blueprints
- Electronic engineers have schematic diagrams
- Mechanical engineers have blueprints and mechanical diagrams
- Software industry now has UML

Benefits of UML

- Software system is professionally designed and documented before any code is written
- Reusable code is easily spotted and coded with the highest efficiency (since system design comes first)
- Lower development costs
- Logical ‘holes’ can be spotted in the design drawings

- Fewer surprises – software will behave as you expect it to
- Overall design will dictate the way software is developed - right decisions are made before any code is written
- UML lets you see the bigger picture
- More memory and processor efficient code is developed
- Modifications on the system are easier

- Less ‘re/learning’ of the system takes place
- Diagrams will quickly get any other developer up to speed

- More efficient to communicate with other programmers

Summary of Benefits

Using UML will result in lower overall costs, more reliable and efficient software, and a better relationship with all parties involved.
Software documentation with UML can be modified much more efficiently.

Parties Involved

The parties involved in developing a software system are:

- **Client** – the person who has the problem to be solved
- **Analyst** – documents client's problem and relays it to:
- **Developers** – programmers who build the software that solves the problem, test it, and deploy it on computer hardware.

Necessity of this Breakdown

Systems today are so complex that one person can't:

- Know all the facets of a business,
- Understand the problem,
- Design a solution,
- Translate it into a problem,
- Deploy the program onto hardware,
- Make sure the hardware components all work together correctly

Old Way of System Modelling

Waterfall Method:

Specifies the analysis, design, coding and deployment follow one after the other. Only when one is complete can the next one follow.

Drawbacks

If an analyst hands off analysis to a designer, who hands off a design to a developer, chances are the 3 team members will rarely work together and share important insights.
The waterfall method supports a big amount of project time given to coding. This takes valuable time away from analysis and design.

New Way of System Modelling

Continuing interplay among the stages of development is encouraged.
As understanding of the system grows, the team incorporates new ideas and builds a stronger system.

Rapid Application Development (RAD) consists of:

- Requirements gathering
- Analysis
- Design
- Development
- Deployment

I-Requirements Gathering

Understand what the client wants!

1. Discover Business Processes

Analysts gain an understanding of the client’s business processes by interviewing the client or having the client go through the relevant processes step-by-step.

Activity diagrams

2. Perform Domain Analysis

Analyst makes an understanding of the major entities in the client’s domain. The object modeller listens for nouns and starts by making each noun a class. They also listen for verbs which ultimately become operations of the classes.

High-level class diagrams and meeting notes

3. Identify Cooperating Systems

Development team finds out exactly which systems the new system will depend on, and which systems will depend on it. System engineer takes care of this action.

Deployment diagram

4. Discover System Requirements

Team goes through its first Joint Application Development (JAD) session. This brings together decision makers from client’s organisation, potential users and members of the development team. Facilitator elicits from decision-makers and users what they want the system to do, team members take notes and object modeler refines the class diagrams.

Package diagram

5. Present Results to Client

When the team finishes all the Requirements actions, the project manager presents the results to the client.

II-Analysis

6. Understanding System Usage

In JAD session, development team works with users to discover actors who initiate each use case from the Requirements JAD session (actor can be a system as well as a person)

Use case diagrams

7. Flesh Out Use Cases

Analyse each sequence of steps in each use case, while still working with the users

Text description of the steps in each use case diagram

8. Refine the Class Diagrams

During JAD sessions, object modeller listens to discussions and refines all the class diagrams. Fills in the names of the associations, abstract classes, multiplicities, generalisations and aggregations.
Refined class diagrams

9. Analyse Changes of State in Objects

Object modeller refines the model by showing changes of state whenever necessary.
State diagram

10. Define Interactions Among Objects

Development team has a set of use cases and refined class diagrams. Using these, they define how the objects interact. Object modeller develops a set of diagrams which include state changes.
Sequence and collaboration diagrams

11. Analyse Interaction with Cooperating Systems

System engineer uncovers specific details of the integration with the cooperating systems. What type of communication is involved, network architecture, access to databases, what kind of databases, etc.
Detailed deployment diagram and maybe data models

III-Design

12. Develop and Refine Object Diagrams

Programmers take class diagrams and generate any object diagrams by examining each operation and developing a corresponding activity diagram. These serve as the basis for much of the coding in the Development segment.
Activity diagrams

13. Develop Component Diagrams

Programmers visualise the components that will result from the next segment and show the dependencies among them.
Component diagrams

14. Plan for Deployment

System engineer begins planning for deployment and for integration with cooperating systems. Diagrams created here show where the components will reside
Part of the deployment diagram developed earlier

15. Design and Prototype User Interface

Another JAD session with the users, continuation of previous JAD sessions. Interplay between analysis and design. GUI analyst works with users to create paper prototypes of screens that correspond to groups of use cases
Screen shots of screen prototypes

16. Design Tests

Developer or test specialist from outside development team uses the use case diagrams to develop test scripts for automated test tools
Test scripts

17. Begin Documentation

Documentation specialists work with designers to begin documentation and arrive at high-level structure for each document
Document structure

IV-Development

18. Construct Code

With the class, object, activity and component diagrams in hand, programmers construct the code for the system.
The code

19. Test Code

This action feeds back into preceding action and vice versa, until all code passes all levels of testing, developed in previous segment (Design Tests).
Test results

20. Construct User Interfaces, Connect to Code and Test

Connection between Design and Development, GUI specialist constructs approved interface prototypes and connects them to code, also ensures that interfaces work correctly.
Functioning system, complete with user interfaces

21. Complete Documentation

Documentation experts work in parallel with programmers to complete documentation.
System documentation

V-Deployment

22. Plan for Backup and Recovery

System engineer creates a plan for steps to follow in case the system crashes, can start before the Development segment begins.
Crash recovery plan

23. Install the Finished System on Appropriate Hardware

System engineer will install the system with any necessary help from the programmers.
Fully deployed system

24. Test the Installed System

After installing the software the development team tests the system. Does it perform as it's supposed to? Does the backup and recovery plan work? Results of this test determine whether future refinement is necessary
Test results

Summary

A skeleton of development process is **GRAPPLE** (Guidelines for Rapid Application Engineering), which consists of 5 segments:

- Requirements gathering
- Analysis
- Design
- Development
- Deployment

Each segment consists of a number of actions, and each action results in a work-product.
UML diagrams are work products for many of these actions

UML Usage

In UML, a certain number of graphical elements are combined inot diagrams and because UML is a language, there are rules for combining these elements.
Purpose of diagrams is to present multiple views of a system. This set of multiple views is called a **model**.
UML model describes what a system is supposed to do, it doesn't tell how to implement the system.

UML Components

UML consists of 9 basic diagrams (hybrids of these diagrams are possible)

1. Class Diagram

Things naturally fall into categories (books, trees, computers,). We refer to these categories as classes

2. Object Diagram

An object is an instance of a class – a specific thing that has specific values of attributes, and behaviour

3. Use Case Diagram

A use case is a description of a system's behaviour from a user's standpoint. A use case diagram is useful for gathering system requirements from a user's point of view.

4. State Diagram

At any given time, an object is in a particular state. State diagrams represent these states and their changes over time.

5. Sequence Diagram

Class and object diagrams represent static information. In a system objects interact with one another and these interactions occur over time. Sequence diagrams show the time-based dynamics of interaction.

6. Activity Diagram

The activities that occur within a use case or within an object's behaviour typically, occur in sequence which is represented with activity diagrams.

7. Collaboration Diagram

The elements of a system work together to accomplish the system's objectives, this is depicted with the collaboration diagram.

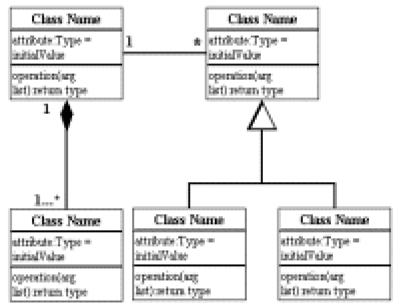
8. Component Diagram

Because large scale system development involves different people working on different components, it's important to have a diagram to model these.

9. Deployment Diagram

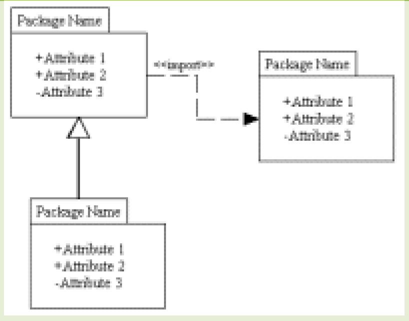
This shows the physical architecture of a computer based system. It can depict the computers and devices, show their connectivity, and show the software that sits on each machine.

UML provides a standard that enables the system analyst to provide a multi-faceted blueprint that's comprehensible to clients, programmers and everyone involved in the development process.


Class Name
attribute:Type = initialValue
operation(arg list) return type
1
*
1...*

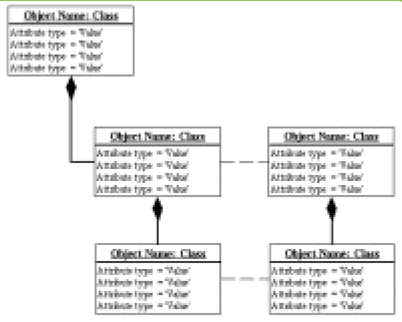
Class Diagrams

Class diagrams are the backbone of almost every object oriented method, including UML. They describe the static structure of a system. Things naturally fall into categories (books, trees, computers,). We refer to these categories as classes.



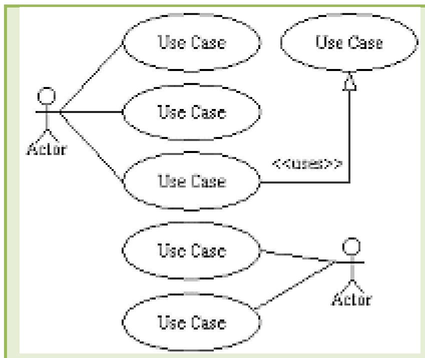
Package Diagrams

Package diagrams are a subset of class diagrams, but developers sometimes treat them as a separate technique. Package diagrams organize elements of a system into related groups to minimize dependencies between packages.



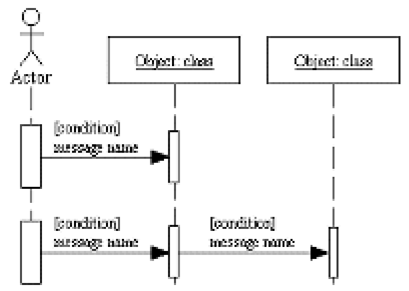
Object Diagrams

An object is an instance of a class – a specific thing that has specific values of attributes, and behaviour. Object diagrams describe the static structure of a system at a particular time. They can be used to test class diagrams for accuracy.



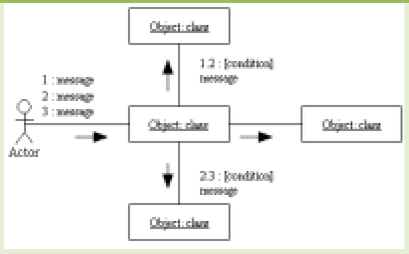
Use Case Diagrams

A use case is a description of a system's behaviour from a user's standpoint. A use case diagram is useful for gathering system requirements from a user's point of view. Use case diagrams model the functionality of system using actors and use cases.



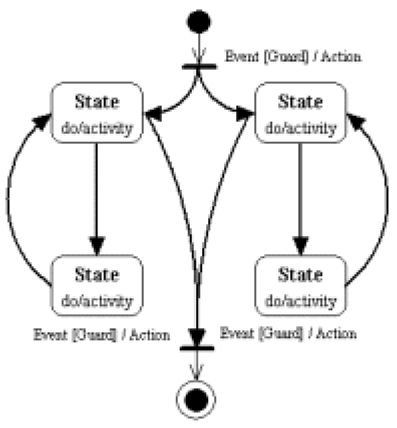
Sequence Diagrams

Class and object diagrams represent static information. In a system objects interact with one another and these interactions occur over time. Sequence diagrams show the time-based dynamics of interaction. Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.



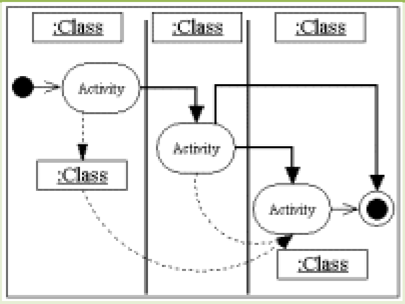
Collaboration Diagrams

Because large scale system development involves different people working on different components, it's important to have a diagram to model these. The elements of a system work together to accomplish the system's objectives; this is depicted with the collaboration diagram. Collaboration diagrams represent interactions between objects as a series of sequenced messages. Collaboration diagrams describe both the static structure and the dynamic behavior of a system.



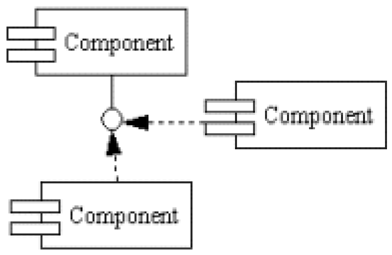
Statechart Diagrams

At any given time, an object is in a particular state. State diagrams represent these states and their changes over time. Statechart diagrams describe the dynamic behaviour of a system in response to external stimuli. Statechart diagrams are especially useful in modelling reactive objects whose states are triggered by specific events.



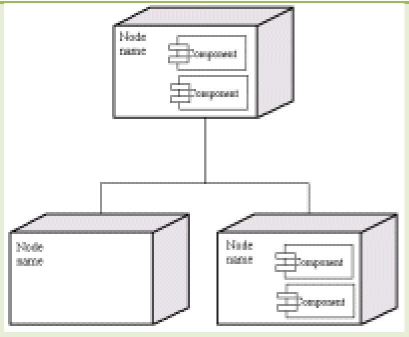
Activity Diagrams

The activities that occur within a use case or within an object's behaviour typically occur in sequence, which is represented with activity diagrams. Activity diagrams illustrate the dynamic nature of a system by modelling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.



Component Diagrams

Component diagrams describe the organization of physical software components, including source code, run-time (binary) code, and executable.



Deployment Diagrams

This shows the physical architecture of a computer based system. It can depict the computers and devices, show their connectivity, and show the software that sits on each machine. Deployment diagrams depict the physical resources in a system, including nodes, components, and connections.

What is a Requirement?

Requirement: A statement about the proposed system that all stakeholders agree must be made true in order for the customer's problem to be adequately solved.

- Short and concise piece of information
- Says something about the system
- All the stakeholders have agreed that it is valid
- It helps solve the customer's problem

A collection of requirements is a requirements document.

Types of Requirements

- Functional requirements

    - Describe **what** the system should do
- Non-functional requirements
    - **Constraints** that must be adhered to during development

Functional requirements

- What inputs the system should accept
- What outputs the system should produce
- What data the system should store that other systems
- might use
- What computations the system should perform
- The timing and synchronization of the above

Non-functional requirements

- All must be verifiable
- Three main types
    - 1. Categories reflecting: usability, efficiency, reliability, maintainability and reusability
- Response time
- Throughput
- Resource usage
- Reliability
- Availability
- Recovery from failure
- Allowances for maintainability and enhancement
- Allowances for reusability
- Categories constraining the environment and technologyof the system.
- Platform
- Technology to be used
- Categories constraining the project plan and development methods
- Development process (methodology) to be used
- Cost and delivery date
- Often put in contract or project plan instead

Some Techniques for Gathering and Analyzing Requirements

Observation

- Read documents and discuss requirements with users
- Shadowing important potential users as they do their work
    - ask the user to explain everything he or she is doing

Interviewing

- Conduct a series of interviews
    - Ask about specific details
    - Ask about the stakeholder's vision for the future
    - Ask if they have alternative ideas
    - Ask for other sources of information
    - Ask them to draw diagrams

Brainstorming

- Appoint an experienced moderator
- Arrange the attendees around a table
- Decide on a 'trigger question'
- Ask each participant to write an answer and pass the paper to its neighbor

Prototyping

- The simplest kind: paper prototype.
  - .a set of pictures of the system that are shown to users in sequence to explain what would happen
- The most common: a mock-up of the system's UI
  - Written in a rapid prototyping language
  - Does not normally perform any computations, access any
  - databases or interact with any other systems
  - May prototype a particular aspect of the system

Use case analysis

- Determine the classes of users that will use the facilities of this system (actors)
- Determine the tasks that each actor will need to do with the system

____________________End of Notes__________________