



ANALISIS DAN PERANCANGAN PERANGKAT LUNAK (APPL)

11 UML - Usecase



Nurfitria Khoirunnisa

D4 Teknologi Rekayasa Perangkat Lunak
Politeknik Negeri Subang

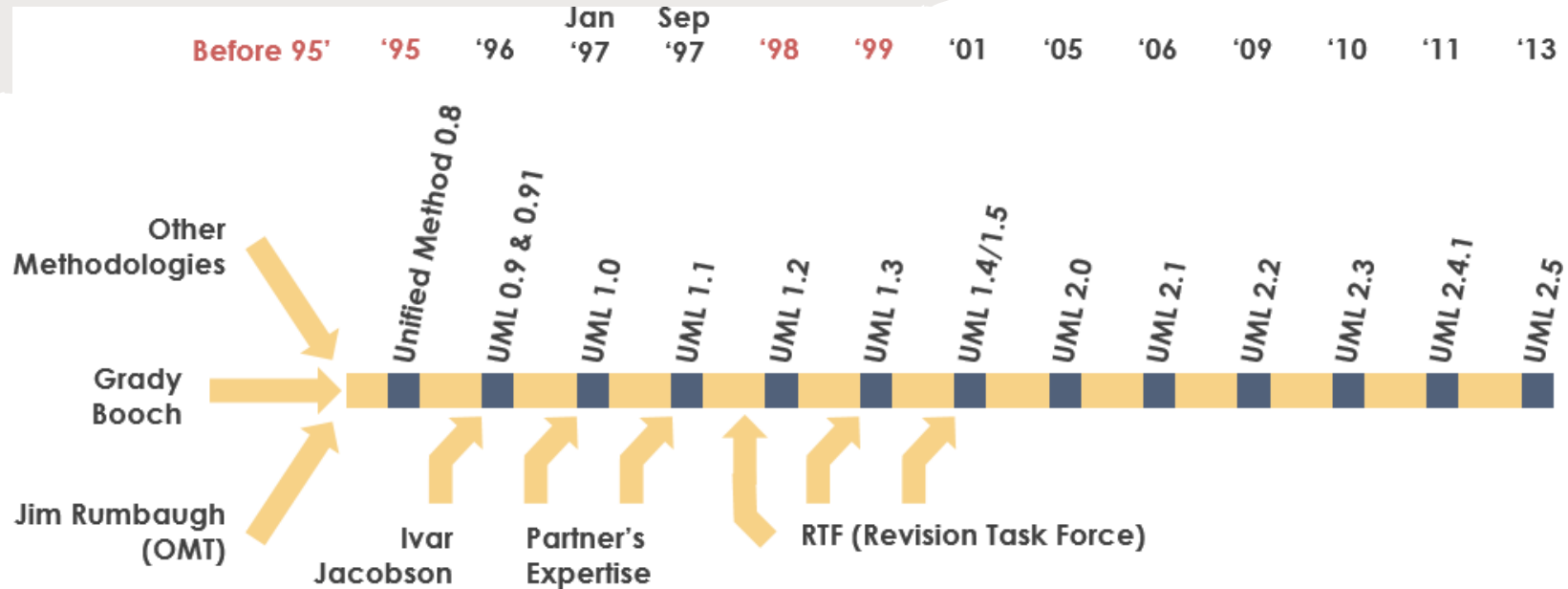
OVERVIEW

1. Pengenalan UML sebagai alat bantu pemodelan pada pembangunan perangkat lunak menggunakan pendekatan analisis dan desain berorientasi objek
2. Notasi, Semantik, dan Stereotype pada UML
3. Komponen pada UML
4. Pengenalan Use Case Diagram sebagai Use Case View pada UML
5. Penjelasan Aktor, Generalisasi Aktor, dan Use Case
6. Penjelasan Use Case Scenario
7. Use Case Refinement (Relasi Include, Extends, dan Generalisasi Use Case)

WHAT IS UML?

1. The Unified Modeling Language (UML) is the **standard modeling language** for software and systems development.
2. A model is an **abstraction of the real thing**.
3. When you model a system, you abstract away any details that are irrelevant or potentially confusing.
4. **Your model is a simplification of the real system**, so it allows the design and viability of a system to be understood, evaluated, and criticized quicker than if you had to dig through the actual system itself.
5. To effectively model a system, you need one very important thing: a language with which the model can be described. And here's where **UML comes in**.

STORY OF UML



Before 95' - Fragmentation



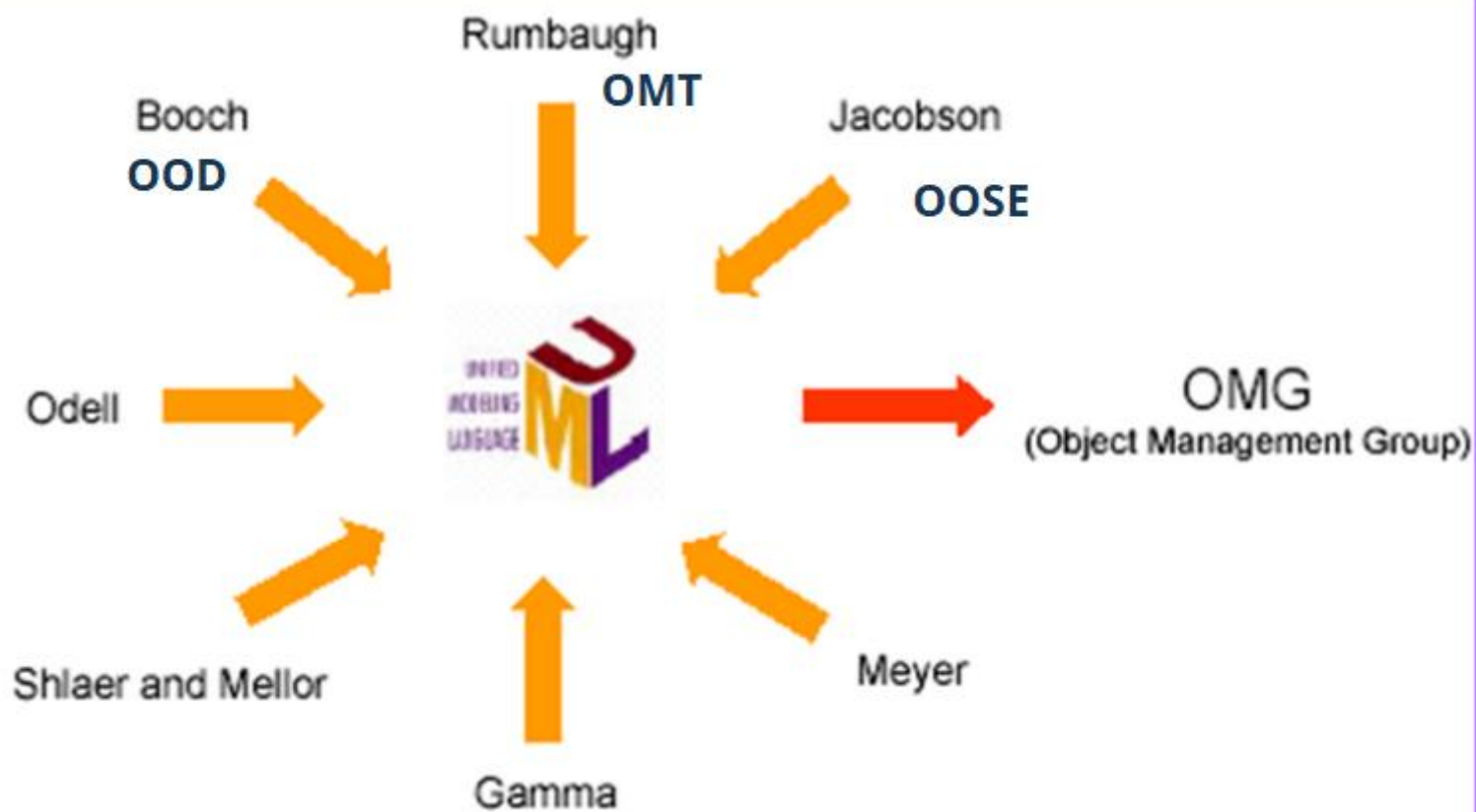
95' - Unification



98' - Standardization



99' - Industrialization



ADVANTAGE OF UML?

1. IT'S A FORMAL LANGUAGE

Each element of the language has a strongly defined meaning, so you can be confident that when you model a particular facet of your system it will not be misunderstood.

2. IT'S CONCISE

The entire language is made up of simple and straightforward notation.

3. IT'S COMPREHENSIVE

It describes all important aspects of a system.

5. IT'S BUILT ON LESSONS LEARNED

UML is the culmination of best practices in the object-oriented community during the past 15 years.

6. IT'S THE STANDARD

UML is controlled by an open standards group with active contributions from a worldwide group of vendors and academics, which fends off "vendor lock-in." The standard ensures UML's transformability and interoperability, which means you aren't tied to a particular product.

UML & SOFTWARE DEVELOPMENT PROCESS

WATERFALL

The waterfall method attempts to pin down the requirements early in the project life cycle. After gathering requirements, software design is performed in full. Once the design is complete, the software is implemented. The problem with this method is that if a change in requirements occurs, the impact can be devastating.

AGILE METHODS

Agile methods **use iterations in extremely short bursts and attempt to minimize risk** by always having a working system of expanding capabilities. Methodologies under this category have introduced some of the more interesting development practices, such as pair programming and test-driven development. **Agile methods emphasize using UML as a sketch.**

VIEW OF THE MODEL

LOGICAL VIEW

Describes the abstract descriptions of a system's parts. Used to model what a system is made up of and **how the parts interact** with each other. The types of UML diagrams that typically make up this view include **class, object, state machine, and interaction diagrams**.

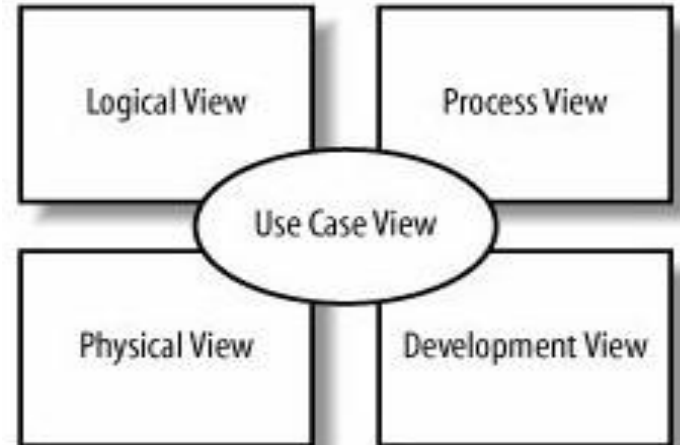
PROCESS VIEW

Describes **the processes within your system**. It is particularly helpful when visualizing what must happen within your system. This view typically contains **activity diagrams**.

DEVELOPMENT VIEW

Describes **how your system's parts are organized into modules and components**. It is very useful to manage layers within your system's architecture. This view typically **contains package and component diagrams**.

Philippe Kruchten's 4+1 view model



VIEW OF THE MODEL

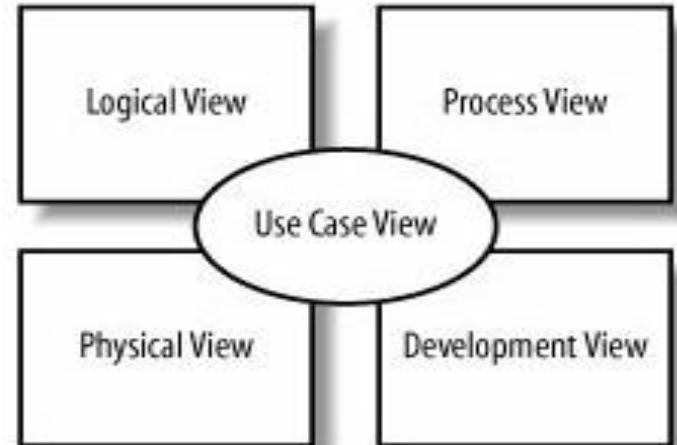
PHYSICAL VIEW

Describes **how the system's design, as described in the three previous views, is then brought to life as a set of real-world entities.** The diagrams in this view show how the abstract parts map into the final deployed system. **This view typically contains deployment diagrams.**

USE CASE VIEW

Describes the functionality of the system being modeled from the perspective of the outside world. This view is needed to describe what the system is supposed to do. All of the other views rely on the use case view to guide them that's why the model is called 4+1. This view **typically contains use case diagrams, descriptions, and overview diagrams.**

Philippe Kruchten's 4+1 view model

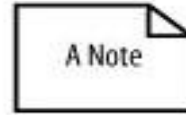


ELEMENT NOTATION OF UML

The elements that make up a modeling language are called its notation

1. NOTES

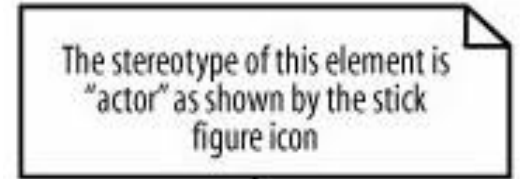
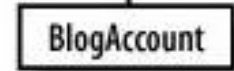
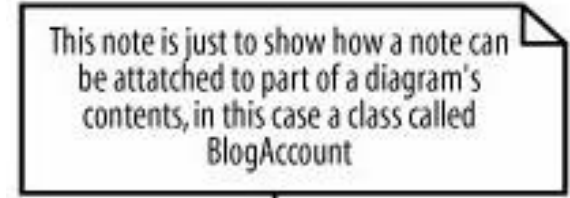
Notes allow you to enter additional comments that aren't captured in your diagrams. You can write anything you want in a note to explain your diagram, similar to a comment in code.



2. STEREOTYPES

Stereotypes signify a special use or intent and can be applied to almost any element of UML notation. Stereotypes modify the meaning of an element and describe the element's role within your model.

A stereotype sometimes has an associated icon



Administrator

BUILDING BLOCKS OF UML

1. Things – Important modelling concepts
2. Relationships – tying individual things
3. Diagram – Grouping interrelated collections of things and relationships

THINGS COMPONENT

| UML Elements | Specific UML Details |
|---------------------|----------------------|
| Structural Things | Classes |
| | Interfaces |
| | Collaborations |
| | Use Cases |
| | Active Classes |
| | Components |
| | Nodes |
| Behavioral Things | Interactions |
| | State Machines |
| Grouping Things | Packages |
| Annotational Things | Notes |

BUILDING BLOCKS OF UML

RELATIONSHIP COMPONENT

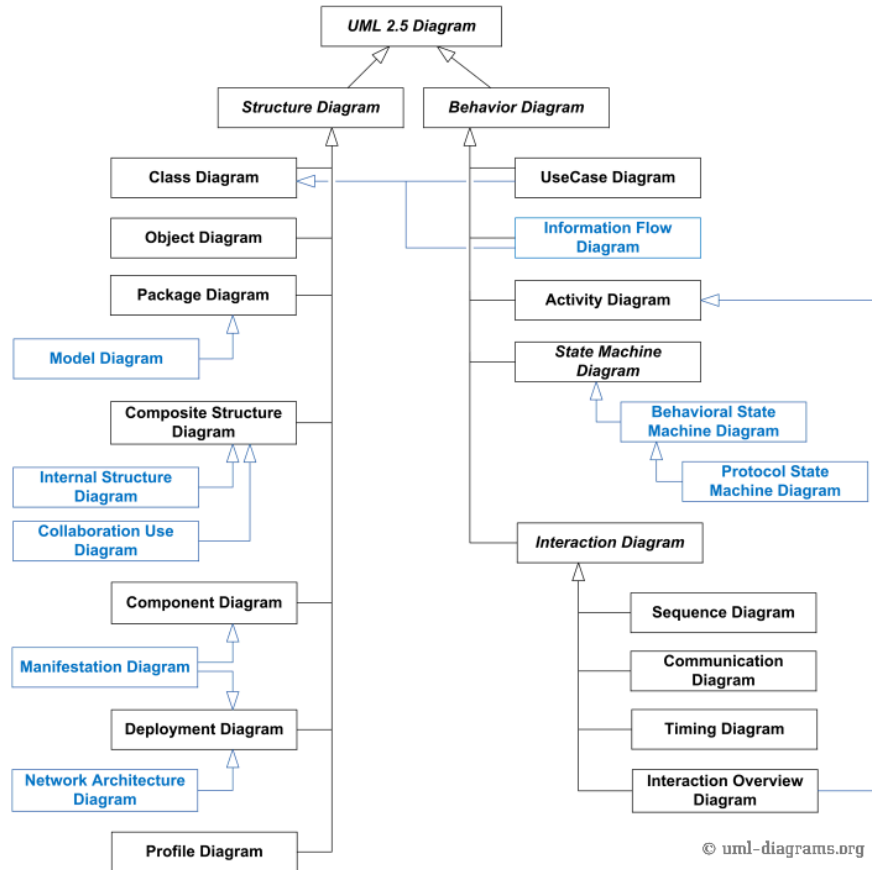
| UML Elements | Specific UML Details |
|--------------------------|----------------------|
| Structural Relationships | Dependencies |
| | Aggregations |
| | Associations |
| | Generalizations |
| Behavioral Relationships | Communicates |
| | Includes |
| | Extends |
| | Generalizes |

1. STRUKTURAL DIAGRAM

Digunakan untuk mendeskripsikan relasi antar kelas

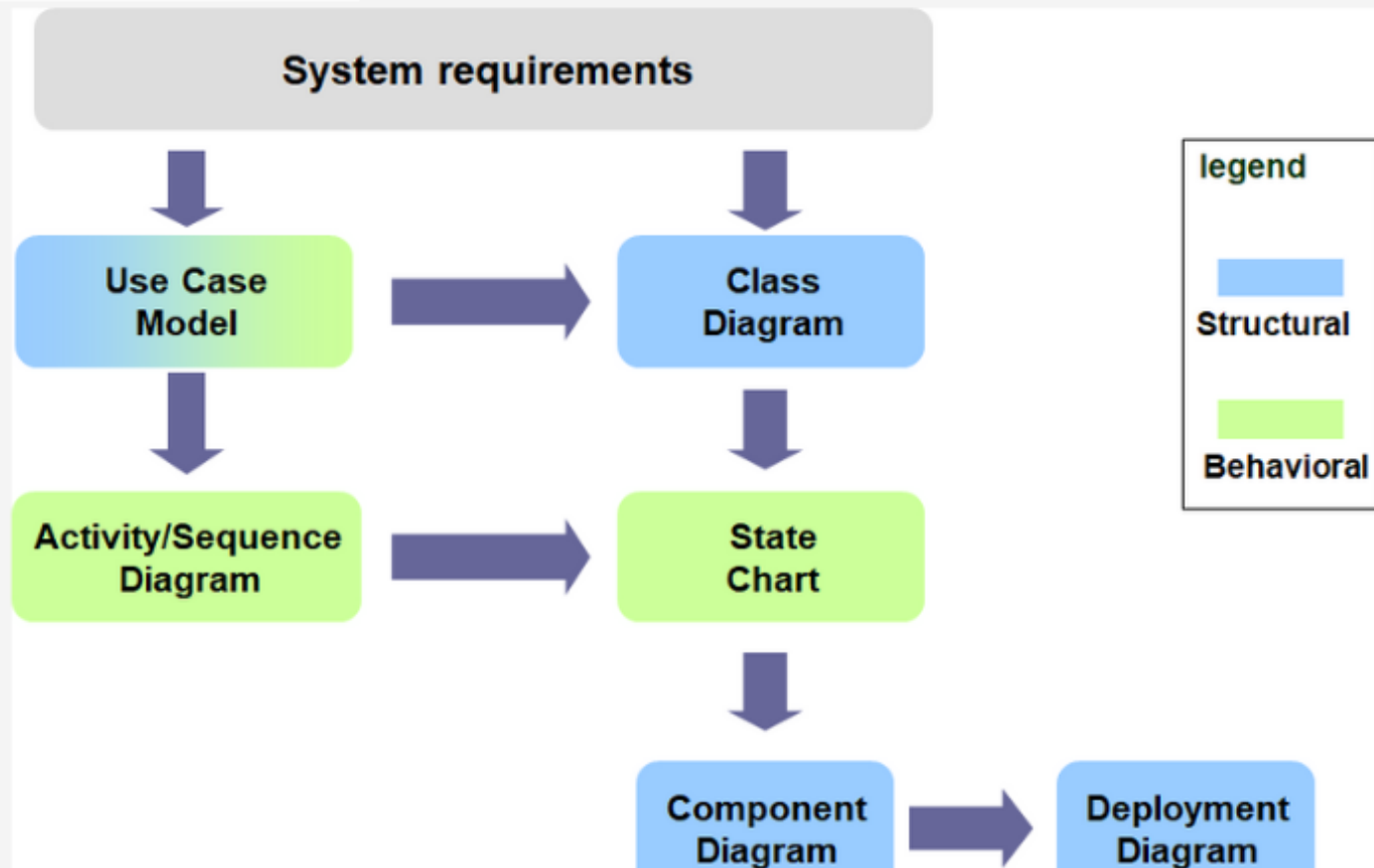
2. BEHAVIORAL DIAGRAM

Digunakan untuk mendeskripsikan interaksi antara aktor dan sebuah use case (bagaimana aktor menggunakan sistem)



Analysis and Design Process

Zachman Framework

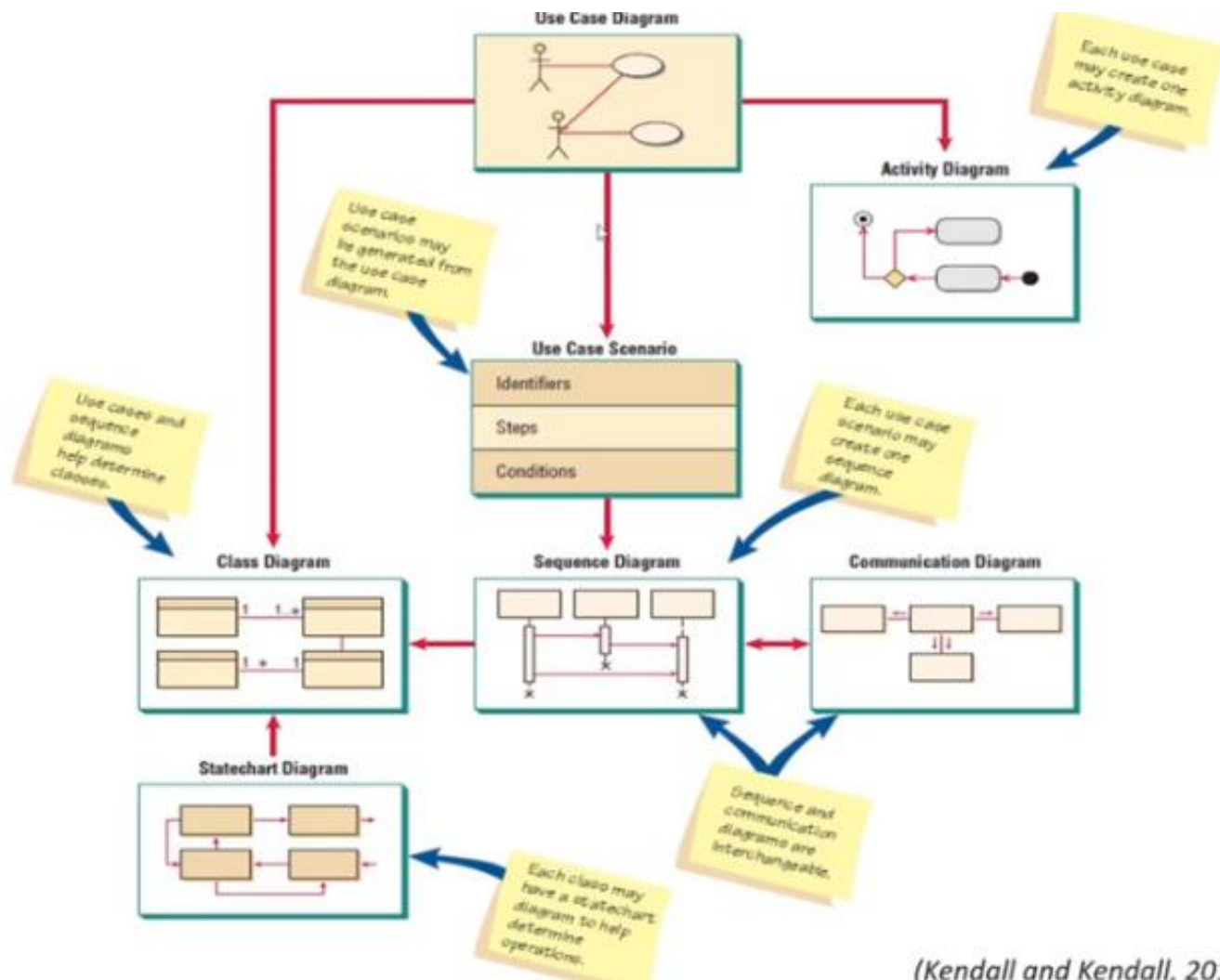


UML based Software Analysis and Design (Sparx Systems EA)

1. Display the boundary of a system and its major functions using **use cases and actors**
2. Model the organization's business process with **activity diagram**
3. Illustrate use case realizations with **sequence diagrams**
4. Represent a static structure of a system using **class diagrams**
5. Reveal the physical implementation architecture with **deployment diagrams**

UML based Software Analysis and Design (Kendal, 2011)

1. A **use case diagram**, describing how the system is used. **Analysts start with a use case diagram**
2. An **activity diagram**, illustrating the overall flow of activities. **Each use case may create one activity diagram**
3. **Sequence diagrams**, showing the sequence of activities and class relationships. **Each use case may create one or more sequence diagrams**
4. **Class diagrams**, showing the classes and relationships. **Sequence diagrams are used to determine classes**
5. **Statechart diagrams**, showing the state transitions. Each class may create a statechart diagram, which is **useful for determining class methods**



(Kendall and Kendall, 2011)

UML based Software Analysis and Design

(Wahono, 2009)

1. Systems Analysis

1.1 Identifikasi Proses Bisnis
dengan **Use Case Diagram**

1.2 Pemodelan Proses Bisnis
dengan **Activity Diagram** atau
BPMN

1.3 Realisasi Proses Bisnis
dengan **Sequence Diagram**
(**Boundary** - **Control** - **Entity**)



2. Systems Design

2.1 Pemodelan **Class Diagram**

2.2 Pemodelan **User Interface Design**

2.3 Pemodelan **Data Model**

2.4 Pemodelan **Deployment Diagram**

WHAT IS USE CASES?

- A use case is a case (or situation) where your system is used to fulfill one or more of your user's requirements; a use case captures a piece of functionality that the system provides.
- Use cases are at the heart of your model
- Use cases specify only what your system is supposed to do, i.e., the system's functional requirements.
- They do not specify what the system shall not do, i.e., the system's nonfunctional requirements. Nonfunctional requirements often include performance targets and programming languages, etc.

USECASE DIAGRAM

Pemodelan untuk menggambarkan kelakuan (behavior) sistem yang akan dibuat

Usecase adalah deskripsi fungsi dari sebuah system dari perspektif pengguna



- Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”.
- Urutan langkah-langkah yang menerangkan antara aktor dengan sistem disebut **skenario**. Setiap skenario mendeskripsikan urutan kejadian.

USECASE DIAGRAM

Usecase merupakan abstraksi dari interaksi antara system dan aktor.




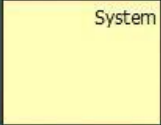
Usecase dibuat berdasarkan keperluan aktor.



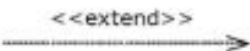
Setiap usecase harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan aktor.

Nama usecase boleh terdiri dari beberapa kata dan tidak boleh ada dua use case yang memiliki nama yang sama


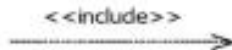



SYMBOL USECASE DIAGRAM

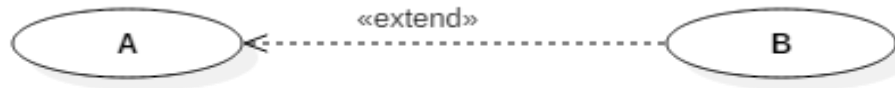
| SIMBOL | NAMA SIMBOL | FUNGSI |
|---|-----------------|---|
|  | Aktor | Pihak yang mengakses use case |
|  | Use Case | Mewakili apa yang sistem bisa lakukan |
|  | Association | Merelasikan aktor dengan use case |
|  | System Boundary | Menggambarkan batasan sistem terhadap lingkungannya |

| Simbol | Deskripsi |
|--|--|
|  | <p>biasanya dinyatakan menggunakan kata benda di awal frase nama aktor</p> |
| <p>Asosiasi / association</p>  | <p>komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor</p> |
| <p>Ekstensi / extend</p>  | <p>relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, misal</p> <div data-bbox="454 682 714 982"> </div> <p>arah panah mengarah pada use case yang ditambahkan</p> |

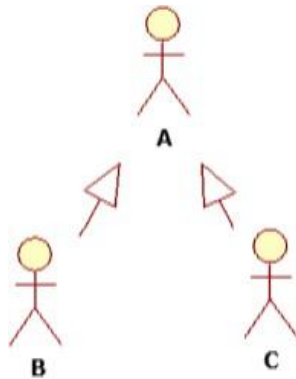
extend dan include termasuk stereotype (model khusus yang terbatas pada kondisi tertentu). Digunakan simbol "<<" dan ">>"

| Simbol | Deskripsi |
|---|--|
|  | <p>dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:</p> <div data-bbox="1410 125 1671 420"> </div> <p>arah panah mengarah pada use case yang menjadi generalisasinya (umum)</p> |
| <p>Menggunakan / include / uses</p>   | <p>relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini</p> <p>ada dua sudut pandang yang cukup besar mengenai include di use case:</p> <ul style="list-style-type: none"> include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut: <div data-bbox="1535 895 1787 1053"> </div> |

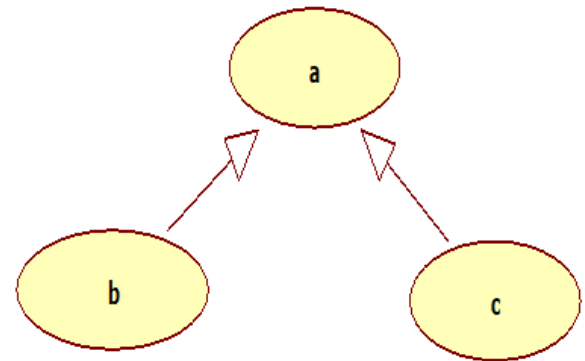
USE CASE RELATIONSHIP



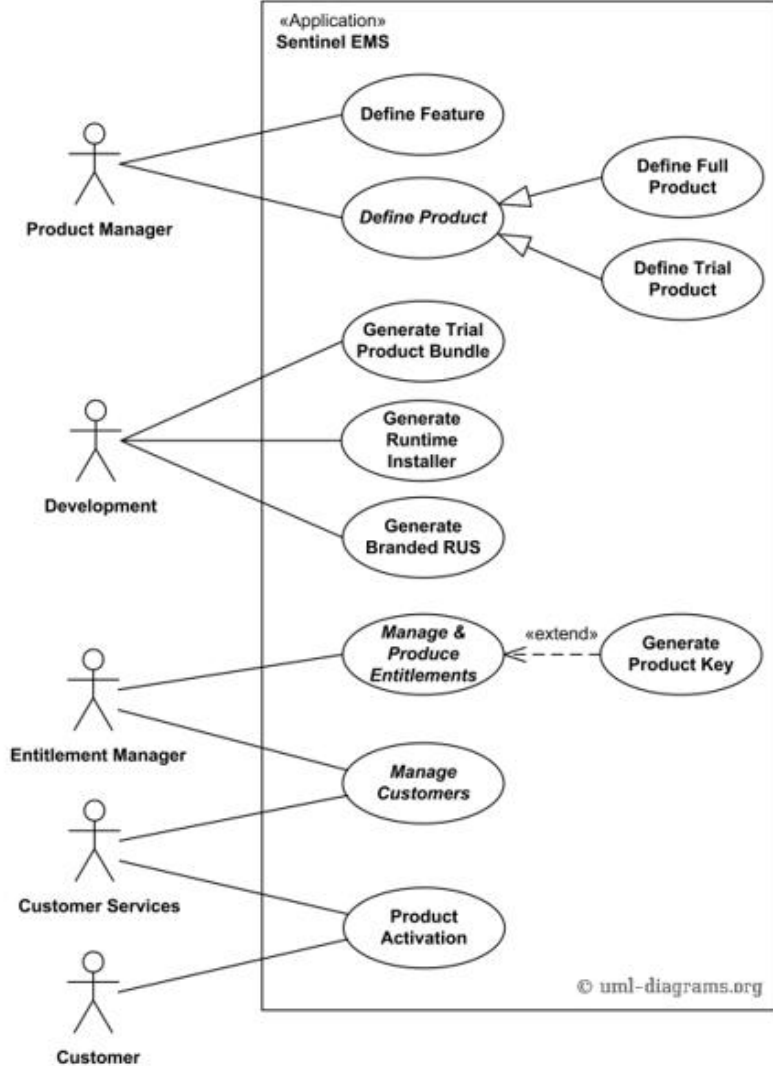
1. INCLUDE
2. EXTEND
3. GENERALIZATION

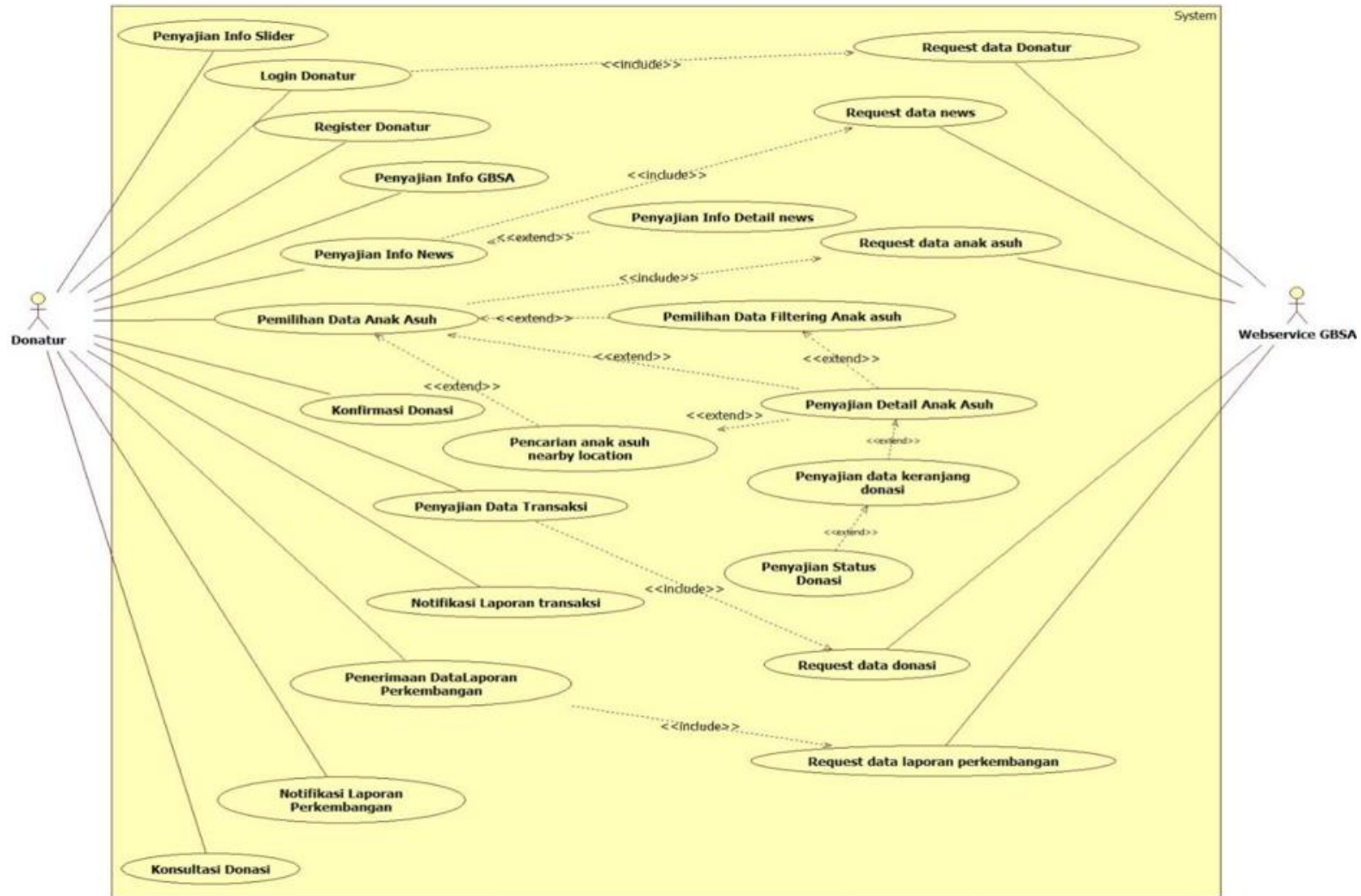


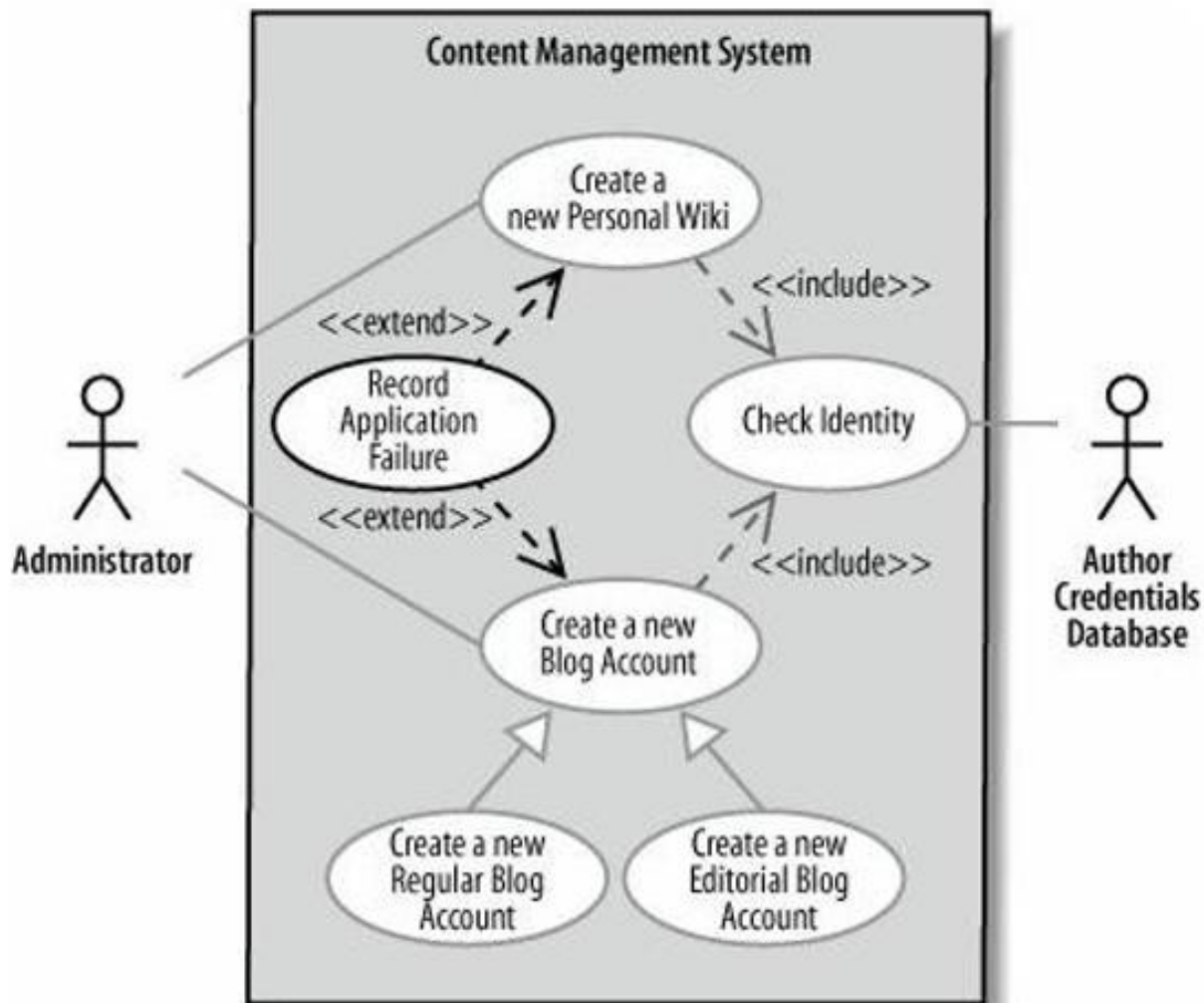
ACTOR GENERALIZATION



USECASE GENERALIZATION







STUDI KASUS 1

Sistem informasi manajemen perpustakaan merupakan sebuah sistem informasi untuk mengelola informasi yang diperlukan dalam suatu perpustakaan yang meliputi pendaftaran Pustaka, anggota, dan proses peminjaman Pustaka. Aturan perpustakaan yang harus diatasi pada sistem informasi manajemen perpustakaan yang akan dimodelkan adalah sebagai berikut :

- 1. Pustaka dapat memiliki lebih dari satu pengarang**
- 2. Anggota dapat memiliki lebih dari satu nomor telepon**
- 3. Seorang anggota dapat melakukan sebuah peminjaman dalam satu waktu dan boleh lebih dari satu Pustaka**
- 4. Seorang anggota dapat mengembalikan Pustaka yang dipinjam tidak dalam waktu yang bersamaan walaupun Pustaka-Pustaka itu dipinjam pada waktu yang sama**
- 5. Pengunjung yang bukan anggota tidak diperbolehkan meminjam Pustaka**
- 6. Proses pendaftaran Pustaka, anggota, dan peminjaman dilakukan oleh petugas perpustakaan**
- 7. Anggota dan pengunjung dapat melakukan pencarian pustaka**

STUDI KASUS 1

Sistem informasi yang akan dibuatkan adalah aplikasi berbasis web. Manajemen Perpustakaan meliputi fungsi-fungsi sebagai berikut:

- 1. Mengelola data Pustaka, meliputi :**
 - a. Memasukkan data Pustaka**
 - b. Mengubah data Pustaka**
 - c. Menghapus data pustaka**
- 2. Mengelola data anggota, meliputi :**
 - a. Memasukkan data anggota**
 - b. Mengubah data anggota**
 - c. Menghapus data anggota**
- 3. Mengelola data peminjaman, meliputi :**
 - a. Memasukkan data peminjaman**
 - b. Mengubah datta peminjaman (mekanisme pengembalian pustaka)**
- 4. Mencari pustaka**

Pemecahan studi kasus tahap pertama yaitu melakukan pencarian aktor. Mulailah bertanya dengan **SIAPA**, **PERAN** dan **NILAI** apa yang akan didapatkan.

| No | Aktor | Deskripsi |
|----|---------------------------------|---|
| 1. | Petugas perpustakaan | orang yang bertugas dan memiliki hak akses untuk melakukan operasi pengelolaan data pustaka, anggota, dan proses pemiinjaman pustaka |
| 2. | Anggota/pengunjung perpustakaan | anggota adalah orang yang diperbolehkan meminjam pustaka sesuai dengan hak aksesnya, sedangkan pengunjung hanya memiliki hak akses melihat pustaka dan membaca di perpustakaan tanpa memiliki hak untuk meminjam pustaka. |

Tahap kedua adalah menemukan use case. Mulailah bertanya dengan INFORMASI apa yang akan diberikan oleh sistem kepada aktor.

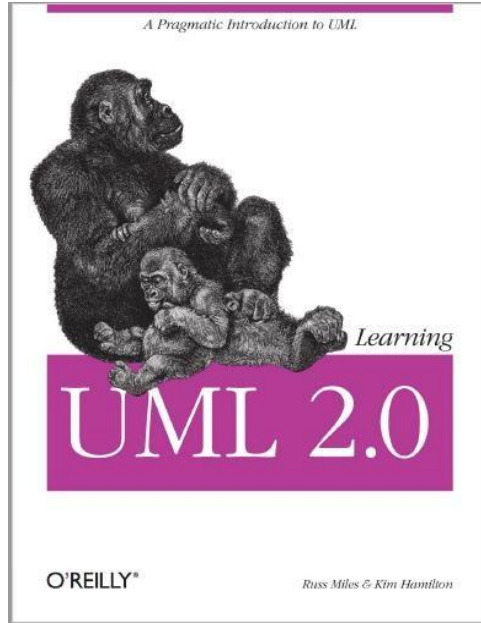
| No | Use case | Deskripsi |
|----|----------------------------|---|
| 1. | Memasukkan data pustaka | merupakan proses memasukkan data pustaka ke dalam basis data |
| 2. | Memasukkan data anggota | merupakan proses memasukkan data anggota ke dalam basis data |
| 3. | Memasukkan data peminjaman | merupakan proses memasukkan data peminjaman ketika ada anggota yang meminjam pustaka |
| 4. | Mencari pustaka | mencari pustaka berdasarkan judul, nama pengarang, jenis, dan kode pustaka dimana akan menampilkan data pustaka yang dicari |

STUDI KASUS 2

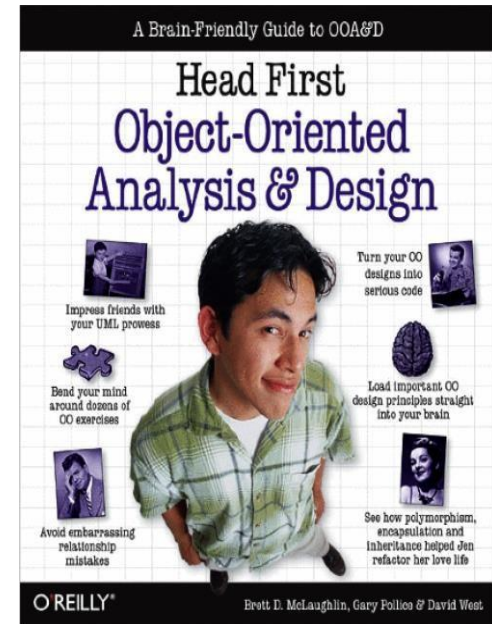
Sistem informasi pengolahan data nilai mahasiswa merupakan sebuah sistem yang mengelola data nilai mahasiswa. Melalui sistem ini mahasiswa dapat mengetahui hasil akhir setelah menyelesaikan ujian akhir semester secara online. Sistem ini dikelola oleh staf prodi dan data-data yang dikelola adalah data mahasiswa, data dosen, data matakuliah, dan data nilai. Aturan penggunaan sistem dapat dimodelkan sebagai berikut :

- 1. Staf prodi dan mahasiswa harus registrasi terlebih dahulu untuk mendapatkan hak akses**
- 2. Proses registrasi mahasiswa hanya dapat dikelola oleh staf prodi**
- 3. Apabila mahasiswa lupa username dan password dapat menghubungi staf prodi untuk mendapatkan hak akses kembali**
- 4. Staf prodi mengelola semua data yang berhubungan dengan nilai mahasiswa**
- 5. Mahasiswa hanya dapat melihat hasil nilai pada sistem dan mencetak nilai tersebut**

BUKU REFERENSI



Miles, Russ & Hamilton, Kim, 2006,
Oreilly Publisher , Learning UML 2.0
259 Page



McLaughlin, Bratt, Police, Gary, and West,
David, 2007,
Oreilly Publisher, Head First Object
Oriented Analysis & Design, 589 Page



THANKS !