BAB III

OBJEK DAN METODE PENELITIAN

3.1. Sejarah Singkat Perpustakaan

Perpustakaan SMU Muhammadiyah mulai berdiri pada tahun 1990 yaitu setahun setelah sekolah SMU Muhammadiyah didirikan. Pada saat itu kepala sekolah SMU Muhammadiyah masih berperan penting dalam memimpin perpustakaan ini. Perpustakaan ini pun masih berlokasi didalam ruang lingkup sekolah SMU Muhammadiyah hingga kini, yaitu di Jl. Aggrek No. 86 Cileungsi, Bogor. Pada saat perpustakaan ini baru didirikan, perpustakaan SMU Muhammadiyah hanya memiliki koleksi buku sejumlah 150 buku pelajaran yang hanya dapat dipinjam oleh siswa atau siswi SMU Muhammadiyah cileungsi saja. Tetapi seiring dengan berkembangnya sekolah SMU Muhammadiyah cileungsi saat ini maka perpustakaanpun ikut merasakan perkembangannya, dengan memiliki 650 jumlah buku saat ini, kepadatan pengunjung pun semakin meningkat, sehingga SMU Muhammadiyah menghadapi kesulitan dalam menyediakan sarana tempat untuk pengunjung perpustakaan yang hendak membaca ataupun meminjam buku diperpustakaan ini.

Member-member yang ada pada perpustakaan SMU Muhammadiyah cileungsi ini tidak lain adalah pelajar ataupun guru di SMU Muhammadiyah, dan selalu dikunjungi oleh kurang lebih 60 pengunjung setiap harinya, hampir 60

persen pengunjung yang datang ke perpustakaan bertujuan hendak meminjam buku yang ada diperpustakaan ini, sedangkan sisanya hanya membaca buku ataupun mencari informasi yang tersedia diperpustakaan ini.

Dengan menimbang sasaran pengunjung yaitu para pelajar SMU maka transaksi peminjaman buku di perpustakaan SMU Muhammadiyah cileungsi ini tidak dipungut biaya untuk setiap peminjaman buku, dan lamanya peminjaman buku selama tiga hari peminjaman, dan jika pengunjung terlambat dalam pengembalian buku maka perpustakaan SMU Muhammadiyah Cileungsi memberikan denda sebesar Rp.500,- untuk satu hari keterlambatannya.

3.2. **Visi dan Misi Perpustakaan**

Adapun Visi serta Misi yang ingin dicapai oleh Perpustakaan SMU Muhammadiyah adalah sebagai berikut :

3.2.1. Visi

Visi Perpustakaan SMU Muhammadiyah Cileungsi adalah ” Perpustakaan sebagai pusat informasi ilmu pengetahuan, dengan mendukung kegiatan belajar mengajar, serta menyediakan fasilitas informasi yang memungkinkan pengguna mengakses informasi secara nyaman diperpustakaan.”

3.2.2. Misi

1. Menyediakan informasi-informasi berkualitas untuk mendukung kegiatan pendidikan disekolah.

2. Menyediakan tempat yang nyaman bagi pengguna perpustakaan.

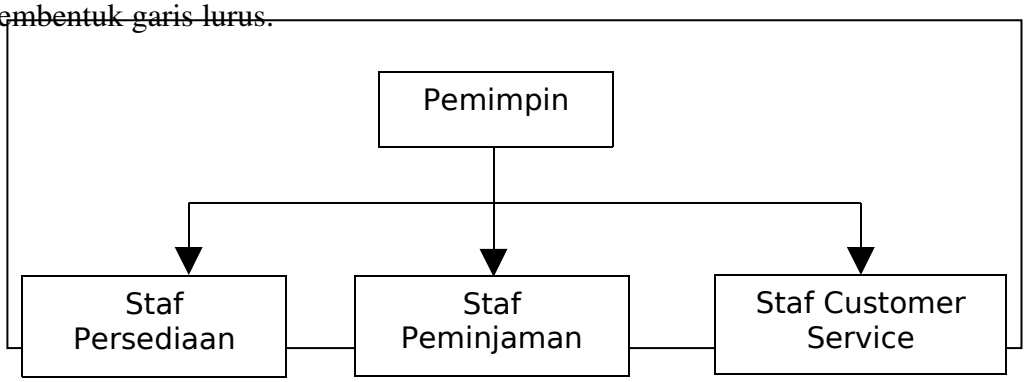
3. Menyediakan layanan khusus bagi pengguna perpustakaan.

4. Menyelenggarakan pendidikan pemakai bagi pengguna.

3.3. Struktur Organisasi Perpustakaan

Dalam melakukan aktivitas sehari-hari, Perpustakaan SMU Muhammadiyah cileungsi perlu menyusun struktur organisasi, karena dengan adanya struktur organisasi dapat membantu perpustakaan untuk melakukan penetapan pembagian kerja, pendelegasian wewenang dan penetapan pegawai pada posisi tertentu agar kegiatan dapat berjalan sesuai dengan yang diharapkan.

Struktur organisasi dibentuk agar wewenang dan tanggung jawab masing-masing bagian menjadi jelas, sehingga tidak menimbulkan kesimpangsiuran dalam pembagian tugas dan pemberian perintah. Selain itu struktur organisasi dapat mempermudah pimpinan untuk melakukan dan meminta pertanggung jawaban atas tugas yang telah dibebankan pada 8 orang pegawainya, yaitu 1 orang pemimpin 4 orang bagian staf peminjaman 2 orang staf persediaan buku serta 2 orang staf bagian customer service. Sedangkan struktur organisasi yang digunakan oleh Perpustakaan SMU Muhammadiyah cileungsi adalah berbentuk garis, dimana wewenang yang ada berjalan dari atasan sampai bawahan, sedangkan untuk pertanggungjawaban bergerak dari bawah menuju atasan sehingga membentuk garis lurus.



Gambar 3.1 Struktur Organisasi Perpustakaan

(Sumber : Perpustakaan SMU Muhammadiyah Cileungsi)

3.4. Deskripsi Pekerjaan (*Job Description*)

1. Pemimpin adalah sebagai penanggungjawab perpustakaan dan sebagai orang yang menerima hasil laporan transaksi yang terjadi diperpustakaan SMU Muhammadiyah Cileungsi.

2. Persediaan barang (buku) mempunyai wewenang mencatat dan melaporkan persediaan buku kepada pemimpin.

3. Bagian peminjaman adalah orang yang mempunyai tanggung jawab dalam mengelola laporan keuangan, transaksi peminjaman buku, dan hal-hal administrasi lainnya.

4. Staf *customer service* hanya melayani konsumen dalam mencari buku yang diinginkan pengunjung, pembuatan id pengunjung, serta yang menerima keluhan-keluhan para pengunjung.

3.5. Metode Pengumpulan Data

Pengumpulan data yang lengkap berhubungan dengan penelitian ini dilakukan dengan survei. Data yang diperlukan adalah data primer dan data sekunder. Kedua jenis data tersebut akan dikumpulkan melalui teknik sebagai berikut.

3.5.1. Sumber Data Primer

Sumber data / informasi dalam penelitian ini berdasarkan kepada jenis data yang diperlukan. Data primer yaitu data yang diperoleh dari responden secara langsung yang dikumpulkan melalui survei lapangan dengan menggunakan teknik pengumpulan data tertentu yang dibuat secara khusus untuk itu [Sek00].

a. Wawancara

Dilakukan dengan pihak-pihak terkait yaitu bagian peminjaman, bagian persediaan buku dan pemimpin perpustakaan dengan harapan dapat melengkapi data-data yang diperlukan dalam penelitian ini.

Dalam penelitian ini, ada beberapa pertimbangan untuk menentukan seseorang sebagai sumber informasi. Dalam menentukan sumber informasi pertimbangannya yaitu.

1. Keakuratan dan validitas informasi yang diperoleh. Berdasarkan hal ini maka jumlah pemberi informasi sangat tergantung pada hasil yang dikehendaki. Bila mereka yang menjadi sumber pemberi informasi adalah orang-orang yang benar-benar menguasi masalah yang diteliti, maka informasi tersebut dijadikan bahan analisis.
2. Jumlah sumber pemberi informasi sangat bergantung pada pencapaian tujuan penelitian, artinya bila masalah-masalah dalam penelitian yang diajukan sudah terjawab dari 5 sumber informasi, maka jumlah tersebut adalah jumlah yang tepat.

3. Peneliti diberi kewenangan dalam menentukan siapa saja yang menjadi sumber pemberi informasi, tidak terpengaruh jabatan seseorang. Bisa saja peneliti membuang pemberi sumber informasi yang dianggap tidak layak.

b. Observasi

Peneliti langsung mengunjungi lokasi penelitian ke bagian peminjaman dan bagian persediaan buku yang dijadikan objek penelitian, hal ini dilakukan untuk melihat dari dekat masalah-masalah yang berhubungan dengan pokok bahasan yang diperlukan dalam penelitian ini.

3.5.2. Sumber Data Sekunder (*Dokumentasi*)

Sumber data sekunder diperoleh dengan cara melakukan kajian kepustakaan, yaitu mempelajari buku-buku literatur dan informasi melalui internet yang relevan dengan variabel yang diteliti, serta laporan-laporan triwulan, tahunan tentang perkembangan. Selain itu sumber data juga diperoleh dari instansi terkait yang ada hubungannya dengan variabel yang digunakan dalam penelitian ini.

Dokumentasi dilakukan dengan menelaah data sekunder yang digunakan untuk menunjang, melengkapi dan menyempurnakan data primer yang diperoleh dari dokumentasi perusahaan, kepustakaan, internet, serta hasil penelitian sebelumnya yang berhubungan dengan masalah dalam penelitian ini. Beberapa dokumen yang telah diberikan seperti:

1. Jumlah buku yang tersedia
2. Kartu anggota perpustakaan

3. Bukti peminjaman buku

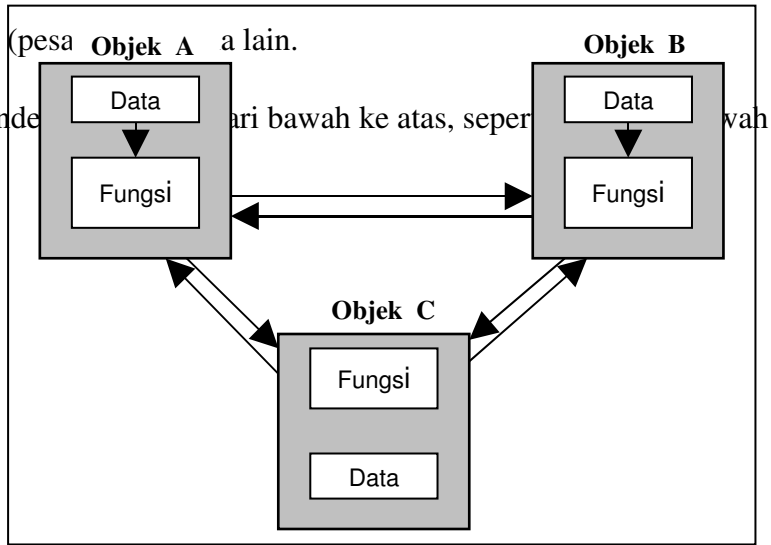
3.6. Metode Pendekatan/Pengembangan Sistem

Metode pendekatan dan pengembangan suatu sistem harus diperhatikan dalam mengerjakan suatu proses pembuatan perangkat lunak, karena dampak dari pendekatan dan pengembangan ini akan mempengaruhi terhadap kesuluruhan sistem.

3.6.1. Metode Pendekatan Sistem

Menurut [Adi05] ada beberapa karakterisitik yang menjadi ciri-ciri dari pendekatan berorientasi objek adalah :

1. Pendekatan lebih pada data dan bukannya pada prosedur/fungsi.
2. Program besar dibagi pada apa yang dinamakan objek-objek.
3. Sturktur data dirancang dan menjadi karakteristik dari objek-objek.
4. Fungsi-fungsi yang mengoperasikan data tergabung dalam suatu objek yang sama.
5. Data tersembunyi dan terlindung dari prosedur/fungsi yang ada di luar.
6. Objek-objek dapat saling berkomunikasi dengan saling mengirim *message* (pesa a lain.
7. Pende ari bawah ke atas, seper wah ini :





Gambar 3.2 Pengorganisasian Data Serta Fungsi Pendekatan Berorientasi Objek (Adi, Nugroho." *Analisis dan Perancangan Sistem Informasi dengan Metodologi Berorientasi Objek*",[ADI05])

Pendekatan beorientasi objek membuat data terbungkus pada setiap fungsi/ prosedur dan melindunginya terhadap perubahan tidak dikehendaki dari fungsi yang berada di luar.

[Lar04] mendefinisikan analisis adalah proses mendekomposisi atau menginvestigasi *domain problem* atau *requirements* menjadi bagian-bagian yang bisa dimengerti. Proses analisis lebih menekankan kepada pertanyaan seperti apakah sistem yang akan dikerjakan, bagaimana sistem tersebut digunakan sehari-hari oleh user.

Perancangan adalah melakukan sintesa model *software* dari masalah yang di analisis, menterjemahkan *domain* model kepada *software* model sehingga bisa dimengerti dan dikomunikasikan kepada anggota tim lain dan juga para profesional *software* lainnya. Bisa disebut juga analisis adalah menentukan apa yang seharusnya dikerjakan, sedangkan perancangan adalah menentukan cara

terbaik untuk mengerjakan sesuatu dengan benar setelah diketahui apa yang harus dikerjakan. Terlihat bahwa analisis dilakukan sebelum perancangan. Analisis berorientasi objek menekankan kepada menemukan dan menjelaskan objek-objek dan interelasinya yang menjadi bagian atau pemain dalam sebuah sistem. Perancangan berorientasi objek memodelkan objek-objek yang diidentifikasi pada proses analisis dan interaksi antar objek ke dalam model *software*. Proses selanjutnya bisa mengidentifikasi atribut dan operasi objek-objek lainnya.

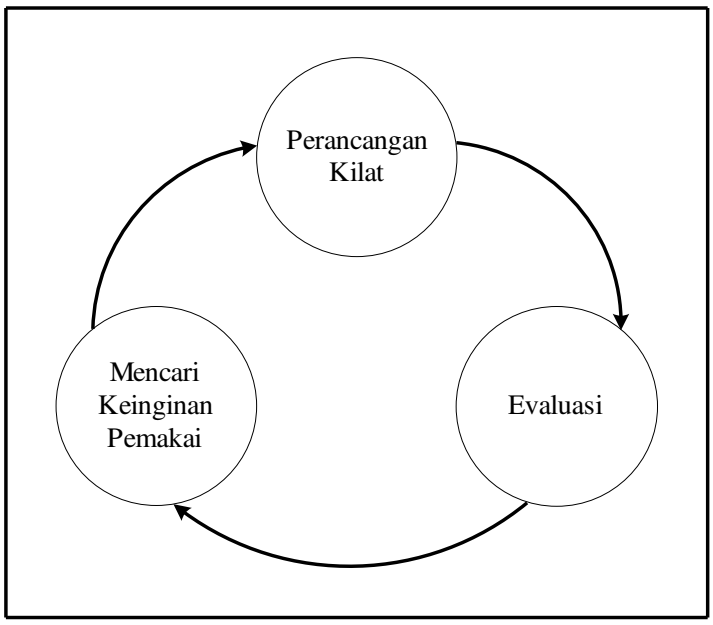
3.6.2. Metode Pengembangan Sistem

Metode pengembangan perangkat lunak yang digunakan dalam penyusunan skripsi ini adalah prototipe, yang dimulai dengan pengumpulan kebutuhan, kemudian pengembang dan pemakai bertemu dan mendefinisikan objektif keseluruhan dari perangkat lunak, mengidentifikasi segala kebutuhan yang diketahui dan area garis besar dimana definisi lebih jauh merupakan keharusan, kemudian dilakukan perancangan kilat. Perancangan kilat membawa kepada konstruksi sebuah prototipe. Prototipe tersebut dievaluasi dan dipakai untuk menyaring kebutuhan pengembang perangkat lunak. Interaksi terjadi pada saat prototipe dirancang untuk memenuhi kebutuhan pelanggan dan pada saat yang sama memungkinkan pengembang untuk secara lebih baik memahami apa yang harus dilakukannya.

Secara ideal prototipe berfungsi sebagai sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Bila prototipe yang sedang bekerja dibangun, pengembang harus mempergunakan fragmen-fragmen program yang

ada sehingga memungkinkan program yang bekerja untuk dimunculkan secara cepat.

Dalam pembangunan sistem, representasi pengetahuan ini terlebih dahulu dilakukan peninjauan keinginan persyaratan objektif para pemakai terhadap perangkat lunak yang akan dirancang. Setelah tahapan tersebut kemudian dilanjutkan dengan perancangan perangkat lunak yang memungkinkan untuk diperbaharui kembali melalui evaluasi program aplikasi yang dibangun dari perspektif pemakai dengan metode sampling.



Gambar 3.3. Model Proses Prototipe

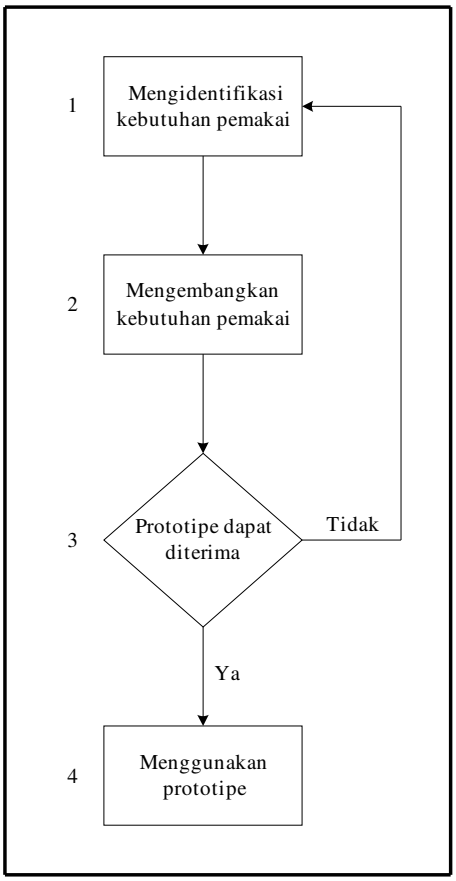
(Pressman, "Rekayasa Perangkat Lunak Pendekatan Praktisi", [Pre02])

Alasan mengapa penulis memakai metode prototipe ini adalah karena metode ini cukup efektif dalam mendapatkan kebutuhan secara cepat yang diperlukan untuk merancang suatu aplikasi khususnya aplikasi berbasis *web site*, serta dapat disetujui pelanggan ataupun pembuat perangkat lunak itu sendiri.

Walaupun pada umumnya prototipe ini akan dihilangkan dan dibuat perangkat yang sebenarnya. Dengan prototipe ini juga, ketika diperlihatkan *working version*-nya pelanggan bisa langsung merasakan seakan-akan itu adalah sistem yang sebenarnya.

Prototyping adalah proses iteratif dalam pengembangan sistem dimana kebutuhan diubah ke dalam sistem yang bekerja *(working system)* yang secara terus menerus diperbaiki melalui kerjasama antara pengguna dan analis.

Adapun langkah-langkah pengembangan sistem metode prototipe dapat dilihat pada gambar 3.4 di bawah ini.



Gambar 3.4. Tahapan Pengembangan Prototipe

Uraian tahapan pengembangan perangkat lunak menggunakan paradigma prototipe dapat dijelaskan sebagai berikut :

1) Mengidentifikasi kebutuhan pemakai

   Pada tahap ini, analisis sistem akan melakukan studi kelayakan dan studi terhadap kebutuhan pemakai baik yang meliputi model *interface,* teknik prosedural, maupun dalam teknologi yang akan digunakan. Pada tahap ini penulis melakukan penelitian terhadap sistem yang sedang berjalan dan menganalisis kebutuhan pemakai, apa saja yang kira-kira akan menjadi kebutuhan pemakai dalam menjalankan aplikasi berbasis *web* yang akan penulis rancang.

2) Mengembangkan kebutuhan pemakai

   Pada tahap kedua ini, analis sistem bekerjasama dengan programer mengembangkan prototipe sistem untuk memperlihatkan kepada pemesan pemodelan atau pihak yang membutuhkan rancangan aplikasi sistem yang akan dibangun. Seperti kegiatan mendefinisikan masalah yang mencakup pendefinisian *input*, proses dan *output* dari sistem yang akan dibangun serta memilih alternatif yang akan dibangun sesuai dengan kebutuhan pemakai. Mengenai implementasi alternatif yang telah dipilih pada tahap ini penulis mulai membuat desain sistem sesuai dengan kebutuhan pemakai yang telah dikembangkan.

3) Menentukan prototipe

Prototipe yang dibangun sudah mencapai tahap apakah akan diterima atau tidak oleh pihak pemakai aplikasi. Analisis sistem pada tahap ini akan mendeteksi dan mengidentifikasi sejauh mana pemodelan yang dibuat dapat diterima oleh pengguna. Perbaikan-perbaikan apa yang diinginkan pengguna atau bahkan harus merubah secara keseluruhan.

4) Menggunakan prototipe

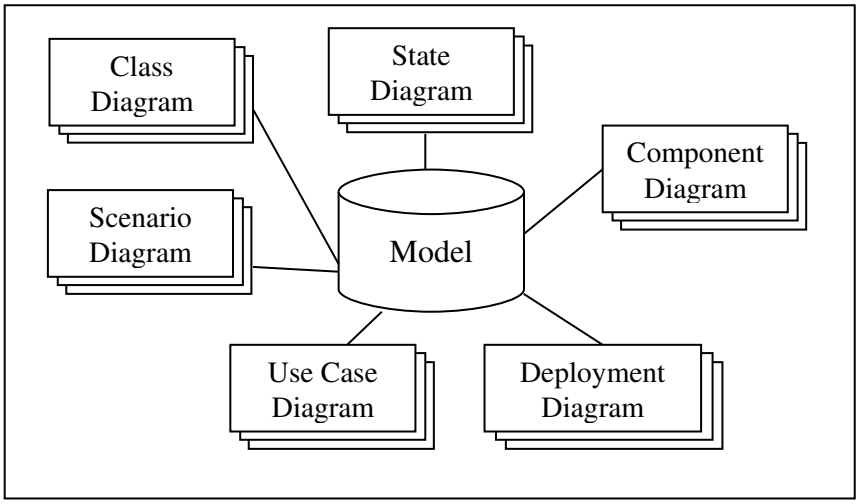
Pada tahap akhir ini, analis sistem akan menyerahkan kepada programer untuk mengimplementasikan pemodelan yang sudah dibuat menjadi suatu sistem. Penulis akan melakukan beberapa test, seperti memasukan sejumlah data ke dalam sistem dan dilihat hasilnya serta cara pemrosesan yang dilakukan oleh aplikasi sistem berbasis *web site* ini.

Dengan langkah-langkah seperti yang disebutkan diatas diharapkan sistem perpustakaan *online* dan sistem otomasi perpustakaan yang dibangun dapat berjalan sesuai dengan yang dikehendaki. Program yang digunakan dapat berupa sebuah web site perpustakaan yang kemudian dikembangkan dan didesign sesuai dengan keinginan perpustakaan maupun program yang dikembangkan secara mandiri oleh tim teknologi informasi perpustakaan.

3.6.3. Alat Bantu Pemodelan

Pemodelan adalah sebuah aktivitas yang telah berlangsung bertahun-tahun dalam pengembangan perangkat lunak. Sebuah model masih diperlukan untuk menulis sebuah aplikasi baik menggunakan bahasa pemograman yang paling

sederhana sampai bahasa yang kompleks. Pemodelan dapat berupa menggambar diagram alir (*flowchart*) ataupun dengan menggunakan *Unified Modeling Language* (UML) yang menunjukkan langkah-langkah yang dikerjakan sebuah aplikasi. Pemodelan sangatlah diperlukan karena membantu tim pengembang untuk visualisasi, spesifikasi, dan mendokumentasikan struktur dan perilaku dari arsitektur sistem. Dengan menggunakan pemodelan yang baku seperti UML, anggota tim pengembang tidak akan ada keraguan dalam mengkomunikasikan keputusan mereka satu dengan yang lainnya.



Gambar 3.5 Model Diagram

(Munawar, "Pemodelan Visual", [MUN05])

Menurut [SH02] pemodelan harus memenuhi syarat berikut.

1. *Readability*

   Readability membawa kejelasan – kemudahan dalam memahami. Memahami sistem adalah langkah pertama baik dalam membangun atau mengembangkan sebuah sistem yang telah ada. Hal ini melibatkan pengetahuan tentang bagaimana sistem dibuat, bagaimana sistem berperilaku dan sebagainya. Memodelkan sebuah sistem membuat sistem mudah dibaca dan yang paling penting, mudah dalam pendokumentasian. Penggambaran sebuah sistem agar mudah dibaca melibatkan pemahaman struktur sistem dan tingkah laku sistem.

2. *Reusability*

   Reusability adalah efek dari membuat sebuah sistem mudah dibaca. Bila sebuah sistem telah dimodelkan sedemikian rupa sehingga mudah dipahami, karena para pengembang cenderung untuk mencari kesamaan atau pengulangan, baik dalam hal fungsionalitas, fitur ataupun struktur.

3.6.3.1. Konsep Dasar UML (*Unified Modelling Language*)

UML (Unified Modelling Language) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan

mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

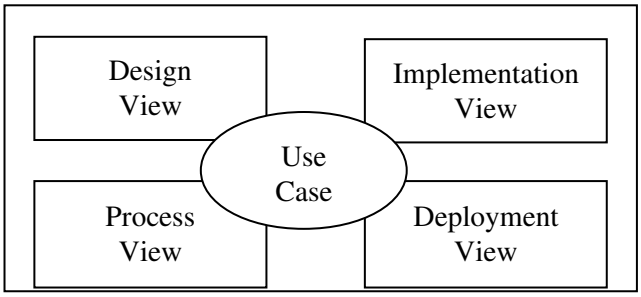
UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT), dan *Object Oriented Software Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan design ke dalam empat tahapan iteratif, yaitu :

1. Identifikasi kelas-kelas dan obyek-obyek
2. Identifikasi semantic dari hubungan obyek dan kelas tersebut
3. Perincian interface
4. Implementasi

Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan *entity-relationship*.

Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih member penekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, design dan implementasi, dan model pengujian (*test model*). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahwa struktur sebuah sistem dideskripsikan dalam 5 *view* dimana salah satu diantaranya *use case view. Use case view* ini memegang peran khusus untuk mengintegrasikan *content* ke *view* yang lain.



Gambar 3.6 Model 4+1 *View*

(Munawar, "Pemodelan Visual", [MUN05])

Kelima view terserbut tidak berhubungan dengan diagram yang dideskripsikan di UML. Setiap view berhubungan dengan perspektif tertentu dimana sistem akan diuji. View yng berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili ketertarikan sekelompok stakeholder tertentu. Penjelasan lengkap tentang sistem bias dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima view tersebut.

Use case view mendefinisikan perilaku eksternal sistem. Hal ini menjadi daya tarik bagi end user, analis dan tester. Pandangan ini mendefinisikan kebutuhan sistem karena mengandung semua view yang lain yang mendeskripsikan aspek-aspek tertentu dan rancangan sistem. Itulah sebabnya use case view menjadi pusat peran dan sering dikatakan yang mendrive proses pengembangan perangkat lunak.

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di use case. Design view ini berisi definisi komponen program, class-class utama bersama-sama dengan spesifikasi data, perilaku dan interaksiny. Informasi yang terkandung di view ini menjadi perhatian para programmer karena menjelaskan secara detil bagaimana fungsionalitas sistem akan diimplementasikan.

Implementation view menjelaskan komponen-komponen fisik dari sistem yang akan dibangun. Hal ini berbeda dengan komponen logic yang dideskripsikan pada design view. Termasuk disini diantaranya file exe, library dan database. Informasi yang ada di view ini relefan dengan aktifitas-aktifitas seperti manajemen konfigurasi dan integrasi sistem.

Process view berhubungan dengan hal-hal yang berkaitan dengan *concurrency* didalam sistem. Sedangkan deployment view menjelaskan bagaimana komponen-komponen fisik didistribusikan ke lingkungan fisik seperti jaringan computer dimana sisetm akan dijalankan. Kedua view ini menunjukan kebutuhan non fungsional dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja.

3.6.3.2. Model – Model Diagram UML

UML terdiri atas banyak elemen-elemen grafis yang digabungkan membentuk diagram. Tujuan representasi elemen-elemen grafis ke dalam diagram adalah untuk menyajikan beragam sudut pandang dari sebuah sistem berdasarkan fungsi masing-masing diagram tersebut. Kumpulan dari beragam sudut pandang

inilah yang disebut sebuah model. Pada saat ini UML telah mencapai versi 2.1 memiliki banyak perubahan dari versi sebelumnya seperti diagram kolobarasi diganti dengan diagram komunikasi dan terdapat tiga tambahan diagram lainnya. Diagram di klasifikasikan berdasarkan tiga jenis diagram sebagai berikut.

Structural Diagram	*1. Class Diagram* *2. Composite Structure Diagram* *3. Object Diagram* *4. Component Diagram* *5. Deployment Diagram* *6. Package Diagram*
Behavioral Diagram	*1. Use Case Diagram* *2. Activity Diagram* *3. State Machine Diagram*
Interaction Diagram	*1. Sequence Diagram* *2. Communication Diagram* *3. Interaction Overview Diagram* *4. Timing Diagram*

Gambar 3.7 Klasifikasi Jenis Diagram UML

(Martin, Flower. "*UML Distilled, Edisi Tiga,"*[MAR05])

1 Class Diagram

Diagram *Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan *object* beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Sebuah *Class* memiliki tiga area pokok yaitu:

1. Nama, merupakan nama dari sebuah kelas.
2. Atribut, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada objek dari *class* .
3. Operasi, adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class*.

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.
4. *Package*, hanya dapat dipanggil oleh *instance* sebuah *class* pada paket yang sama.

Class juga dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *runtime*.

A. Relasi Pada Diagram *Class*

Relasi atau *relationship* merupakan keterhubungan antar kelas yang muncul pada saat sebuah kelas berinteraksi dengan kelas-kelas lainnya. Didalam diagram *class*, setiap kelas pasti akan berinteraksi dengan baik satu maupun lebih kelas. Relasi yang muncul pada setiap keterhubungan antar kelas juga akan memiliki atribut-atribut yang akan lebih menjelaskan definisi dari sebuah relasi

yang terjadi.

Berikut relasi yang terdapat pada diagram *Class* :

1. Asosiasi (*Association*)

   Para pemodel menggunakan pemahaman asosiasi adalah pada saat beberapa kelas saling terhubung satu sama lain secara konseptual. Asosiasi juga dapat menjadi lebih kompleks pada saat beberapa class terhubung ke satu class.

2. *Constraint* Pada Asosiasi

   Kadangkala sebuah asosiasi diantara dua *class* harus mengikuti sebuah aturan dan aturan ini bisa diletakkan dalam sebuah *constraint* pada garis asosiasi dan diletakkan dalam kurung kurawal. Bentuk lain dari tipe *constraint* adalah relasi OR yang ditulis dengan {or} dalam garis putusputus yang menghubungkan dua garis asosiasi. Kondisi OR ini menghadapkan kepada keadaan bahwa sebuah kelas terhubung dengan dua kelas tetapi pada saat mendefinisikan relasinya kelas harus memilih salah satu dari kedua kelas tersebut, sedangkan kondisi untuk memilih keduanya adalah *invalid*.

3. Kelas Asosiasi (*Associations Class*)

   Sebuah asosiasi dapat memiliki atribut dan operasi seperti halnya sebuah *class*. Sebuah *association class* sebenarnya diperlukan apabila salah satu dari kelas yang terhubung mempunyai sebuah atau beberapa atribut yang tidak layak dimiliki oleh kelas tersebut, karena secara logis atribut tersebut lebih layak dimiliki oleh asosiasi yang menghubungkan kedua kelas tersebut.

4. Multiplisitas (*Multiplicity*)

Multiplisitas atau kardinalitas adalah jumlah banyaknya objek sebuah *class* yang berelasi dengan sebuah objek lain pada *class* lain yang berasosiasi dengan *class* tersebut. Untuk menyatakan multiplisitas anda dapat meletakkannya diatas garis asosiasi berdekatan dengan *class* yang sesuai. Ada banyak multiplisitas yang mungkin untuk dipakai.

Tabel berikut menjabarkan multiplisitas yang dapat digunakan. :

Tabel 3.1 Nilai Multiplisitas Potensial

(Munawar, "Pemodelan Visual", [MUN05])

Indicator	**Meaning**
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One to more
N	Only n (where n > 1)
0..n	Zero to n (where n > 1)
1..n	One to n (where n > 1)

B. Generalization & Inheritance

Sebuah *class* (*child class* atau *subclass*) dapat mewarisi atribut-atribut dan operasioperasi dari *class* lainnya (*parent class* atau *super class*) dimana *parent class* bersifat lebih umum daripada *child class*.

Generalisasi pada konsep *Object Oriented* digunakan untuk menjelaskan hubungan kesamaan diantara *class*. Dengan menggunakan generalisasi bisa dibangun struktur logis yang bisa menampilkan derajat kesamaan atau perbedaan diantara *class-class*. Manfaat lain dari struktur hirarkis juga memungkinkan untuk penambahan *subclass* (*child class*) baru tanpa harus merubah struktur yang sudah ada.

Inheritance adalah sebuah mekanisme pengimplementasian generalisasi dan spesialisasi. Aturan inheritance dapat secara umum bisa diklasifikasikan sebagai berikut :

1. *Subclass* selalu mewarisi semua sifat dari *superclass*-nya.
2. Definisi *subclass* selalu mencakup paling tidak satu detil yang tidak diturunkan dari *superclass*-nya.

Inheritance sangat dekat dengan asosiasinya dengan generalisasi. Generalisasi menjelaskan hubungan logis antar elemen-elemen yang mempunyai karakteristik yang sama. Sedangkan *inheritance* menerangkan mekanisme agar bagi pakai (*sharing*) bisa terjadi.

C. Kelas Ketergantungan (*Dependant Class*)

Pada penggunaan relasi kadangkala satu *class* menggunakan *class* yang lain, hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan

untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Notasi untuk *dependency* pada UML dapat menggunakan garis putus-putus dan tanda panah pada ujungnya.

D. Agregasi

Didalam UML, ada relasi dengan perlakukan khusus yang disebut dengan "bagian dari (*part of*)" yang menangani antar objek-objek dimana salah satunya adalah bagian dari yang lain. Dengan kata lain sebuah objek terdiri atas objek-objek yang lain. Hal seperti ini di dalam UML disebut agregasi. Sebuah agregasi adalah kasus khusus dari asosiasi. Agregasi disimbolkan dengan jajaran genjang yang diletakkan pada *class* yang mengandung objek. Kadangkala relasi OR harus digunakan pada agregasi. Untuk memodelkan hal tersebut, sebuah constraint – dengan kalimat atau dalam tanda kurung kurawal pada garis putus-putus harus dibuat untuk menghubungkan dua objek.

Komposit (*composite*) adalah sebuah tipe agregasi yang kuat dimana bagian dari objek tergantung penuh/secara keseluruhan terhadap objeknya sehingga bila sebuah objek komposit dibuang maka bagian yang tergantung pada komponen tersebut akan terbuang juga pada saat yang bersamaan. Notasi komposit sama seperti agregasi hanya saja jajaran genjangnya terisi (solid).

E. *Interface* & Realisasi

Interface adalah satu set *operation* yang memberikan spesifikasi beberapa aspek dari perilaku dan operasi disuatu *class* ke *class* yang lain. Contohnya,

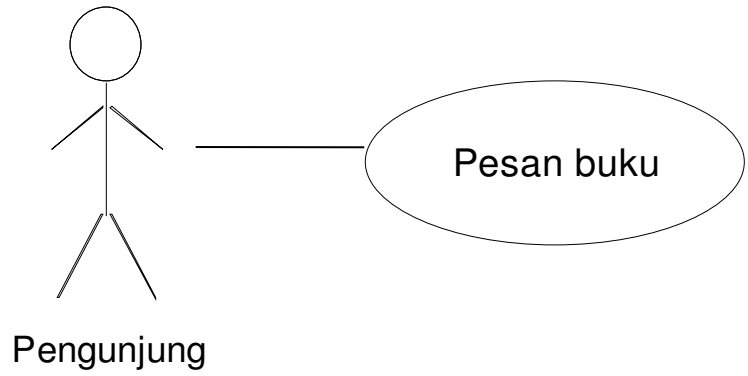
keyboard pada komputer sebenarnya merupakan *interface* yang bisa dipakai ulang karena tombol-tombol *keyboard* sebenarnya berasal dari mesin ketik, hanya saja mungkin ada beberapa operasionalisasi tombol-tombol yang berbeda yang sudah ditransfer ke sistem yang lain. Pemodelan *interface* sama dengan pemodelan *class* hanya saja pada *interface* tidak mempunya atribut dan pada penamaannya perlu ditambahkan *stereotype* <<interface>> atau ditambahkan huruf "I" di depan nama interface. Relasi antara *class* dan *interface* disebut *Realization*. Realisasi dituliskan dengan garis putus-putus dengan segitiga yang mengarah ke *interface*.

2 Use Case Diagram

Diagram *use case* adalah deskripsi fungsi dari sebuah sistem dari perspektif/sudut pandang para pengguna sistem. *Use case* mendefinisikan "apa" yang dilakukan oleh sistem dan elemen-elemennya, bukan "bagaimana" sistem dan elemen-elemennya saling berinteraksi. *Use case* bekerja dengan menggunakan "*scenario*", yaitu deskripsi urutan-urutan langkah yang menerangkan apa yang dilakukan penggunan terhadap sistem maupun sebaliknya. Diagram *use case* mengidentifikasikan fungsionalitas yang dipunyai oleh sistem (*use case*), user yang berinteraksi dengan sistem (*actor*) dan asosiasi/keterhubungan antara user dengan fungsionalitas sistem. Komponen notasi dasar yang dipunyai oleh diagram *use case* adalah *actor*, *use-case*, dan *association*. Berikut adalah notasi yang terdapat pada diagram *use case*. Diagram *use case* memiliki sebuah model khusus yang terbatas untuk kondisi tertentu yang disebut *stereotype*. Untuk menunjukkan *stereotype* digunakan symbol "<<"

diawalnya dan ditutup dengan “>>” diakhirnya. Terdapat dua *stereotype* paling sering digunakan dalam *use case diagram* yaitu <<extend>> dan <<include>>. <<extend>> digunakan untuk menunjukkan bahwa satu *use case* merupakan tambahan fungsional dari *use case* yang lain jika kondisi atau syarat tertentu dipenuhi. Sedangkan <<include>> digunakan untuk menggambarkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya.

Contoh salah satu *use case* model pada sebuah pemesanan buku diperpustakaan :



Gambar 3.8 Model *use case* pemesanan buku

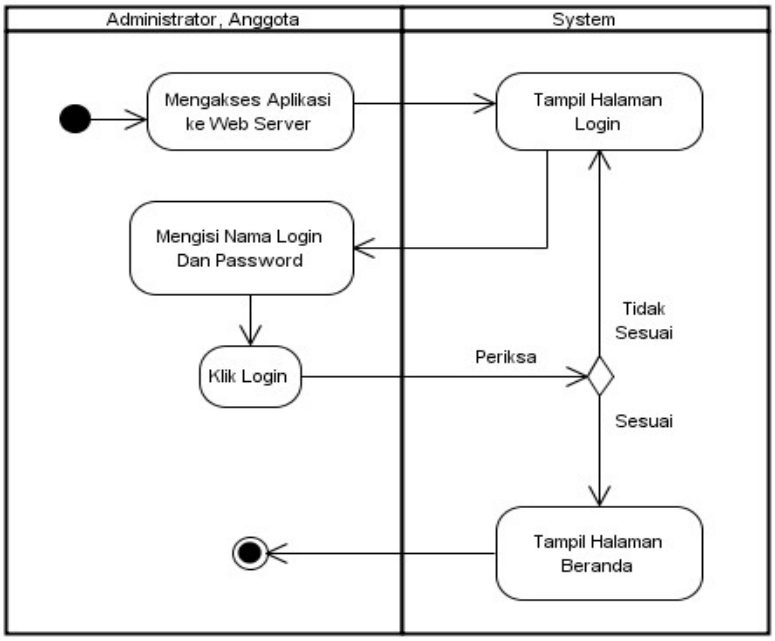
Pada model diatas, digambarkan jika pengunjung perpustakaan dapat melakukan pemesanan buku didalam sebuah aplikasi perpustakaan *online* berbasis website.

3 Activity Diagram

Diagram *Sequence* mendokumentasikan komunikasi/interaksi antar kelas-kelas. Diagram ini menunjukkan sejumlah objek dan *message* (pesan) – yang diletakkan diantara objek-objek didalam *use case*. Perlu diingat bahwa di dalam diagram ini, kelas-kelas dan aktor-aktor diletakkan dibagian atas diagram dengan

urutan dari kiri ke kanan dengan garis *lifeline* yang diletakkan secara vertikal terhadap kelas dan aktor.

Contoh salah satu Activity Diagram pada Activity diagram login :



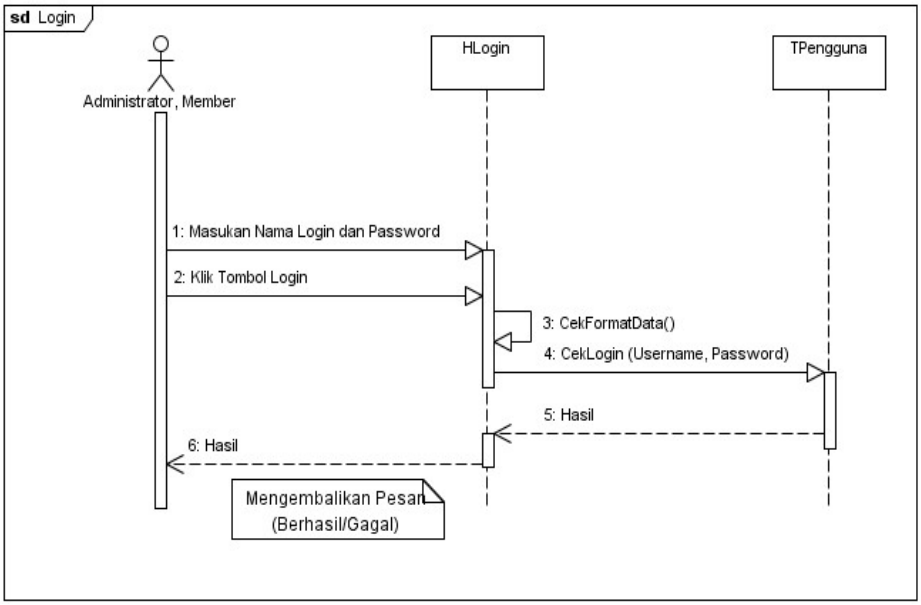
Gambar 3.9 Activity Diagram Login

4 Sequence Diagram

Diagram *Sequence* mendokumentasikan komunikasi/interaksi antar kelas-kelas. Diagram ini menunjukkan sejumlah objek dan *message* (pesan) – yang diletakkan diantara objek-objek didalam *use case*. Perlu diingat bahwa di dalam diagram ini, kelas-kelas dan aktor-aktor diletakkan dibagian atas diagram dengan urutan dari kiri ke kanan dengan garis *lifeline* yang diletakkan secara vertikal terhadap kelas dan aktor.

Diagram sequence merupakan diagram untuk menggambarkan perilaku aktor pada sistem tersebut berdasarkan garis waktu.

Dibawah ini adalah Gambar dari *Sequence Diagram* Login :



Gambar 3.10 *Sequence diagram* login

3.7. Batasan dan Asumsi

Pembatasan masalah ini dilakukan agar dalam pengolahan sistem dapat lebih terarah dari tujuan yang telah ditetapkan. Berdasarakan permasalahan yang timbul, penyusun membatasi beberapa permasalahan sebagai berikut:

1. Sistem yang dirancang hanya membahas prosedure aktivasi pemakaian website perpustakaan *online* hanya ditujukan kepada siswa/siswi serta guru yang berada di SMU Muhammadiyah cileungsi.
2. Sistem yang dirancang hanya membahas pemesanan buku yang dapat dilakukan secara *online* di website perpustakaan *online* SMU Muhammadiyah cileungsi.

3. Sistem yang dirancang hanya membahas pencarian buku perpustakaan dapat langsung dilakukan di website ini berdasarkan data buku yang telah disediakan.
4. Sistem yang dirancang hanya membahas pencarian ebook/artikel atau video tutorial berdasarkan data ebook/artikel dan video tutorial yang disediakan.
5. Sistem yang dirancang hanya membahas download ebook/artikel atau video tutorial yang tersedia di website perpustakaan *online* SMU Muhammadiyah Cileungsi.
6. Sistem yang dirancang hanya menampilkan tampilan data transaksi peminjaman buku.
7. Website perpustakaan *online* ini tidak membahas pengadaan buku yang disediakan pada perpustakaan SMU Muhammadiyah. Baik itu sumber pemasok maupun buku yang akan disediakan.
8. Sistem tidak membahas tentang pembayaran secara *online* pada pemesanan buku yang ada di perpustakaan *online* ini, dikarenakan permintaan dari pihak perpustakaan yang ingin pembayaran dilakukan dengan cara langsung.

Adapun penulis mengasumsikan :

1. Pemesanan buku langsung dilakukan oleh guru ataupun siswa/siswi sebagai usaha untuk menjaga rahasia password user yang dibuat oleh member itu sendiri.

2. Peraturan peminjaman buku sudah ditentukan oleh perpustakaan yang bersangkutan oleh karena itu aplikasi hanya menyediakan input pemesanan serta data buku yang telah dipesan secara *online*.
3. Pengecekan persediaan buku menggunakan sistem yang ada pada aplikasi sehingga admin serta member tidak perlu memeriksa secara langsung.
4. Pembatalan peminjaman akan dikenakan denda sesuai dengan peraturan yang ditetapkan oleh perpustakaan.
5. Peminjaman buku hanya ditetapkan satu judul untuk satu member sehingga tidak terjadi peminjaman yang berulang untuk satu judul buku kepada satu member yang telah meminjam atau memesannya.
6. Maksimal buku yang dapat dipinjam hanya tiga buku untuk setiap member untuk tiga hari peminjamannya.