

BAB II

KAJIAN PUSTAKA

Bab ini berisi penjelasan tentang kajian berbagai pustaka yang digunakan dalam penyusunan Tugas Akhir ini. Kajian pustaka akan dilakukan terhadap beberapa literatur mengenai data *poller*, *graph plotter*, *SMS gateway* yang digunakan, *design pattern*, *Hallway usability testing*, dan lingkungan pengujian VNUML.

2.1 Packet Internet Grouper

Packet Internet grouper atau lebih terkenal dengan nama *ping* adalah salah satu utilitas Internet yang paling tua. Ping secara sederhana menanyakan ketersediaan (*availability*) *host* yang lain dan menyimpan *round-trip time* antara paket *request* dan *reply* [FLA77]. Ping dibuat oleh Marjorie Flack terilhami dari prinsip *echo-location* pada pemodelan sistem sonar dan radar. Ping menggunakan paket *Internet Control Message Protocol* (ICMP) ECHO_REQUEST dan ECHO_REPLY untuk menentukan jarak virtual ke target.

Ketika dijalankan *ping* akan menyiapkan paket untuk *host* tujuan, mengirimkan paket tersebut, kemudian menyimpan waktu yang dibutuhkan agar paket mencapai *host* tujuan. Paket yang dibangkitkan adalah paket ICMP ECHO_REQUEST. Jika *host* tujuan menerima paket tersebut, *host* tujuan akan membangkitkan sebuah paket ICMP ECHO_REPLY. Mekanisme sederhana ini dapat menyediakan informasi sederhana mengenai performansi jaringan komputer yakni *network latency*, waktu yang dibutuhkan paket untuk sampai ke tujuan.

Secara umum, paket ICMP ECHO_REQUEST dan ECHO_REPLY dilewatkan oleh *router* dan *firewall*. Akan tetapi karena keberadaan *Trojan* dan *Distibuted Denial of Service* (DDoS) yang juga memanfaatkan paket ICMP, beberapa jaringan komputer memblok paket ini. *Round-trip time* yang disimpan oleh paket ICMP ECHO_REPLY juga tidak selamanya merepresentasikan kondisi jaringan komputer yang akurat. Karena beberapa *network device* mungkin saja memberikan prioritas berbeda paket yang akan dilewatkan. Dengan demikian ping tidak dapat dijadikan sebagai satu-satunya alat utama untuk menentukan ketersediaan suatu *host*.

2.2 Simple Network Management Protocol

Simple network management protocol (SNMP) adalah sebuah protokol untuk *Internet Network Management Service* yang menyediakan himpunan operasi yang memungkinkan sebuah perangkat jaringan dikelola secara jarak jauh (*remote*) [MAU01][CIK03].

2.1.1. Arsitektur Utama SNMP

Sebuah sistem manajemen SNMP terdiri dari:

1. Satu atau lebih simpul, setiap simpul mempunyai entitas SNMP yang berisi *command responder* dan aplikasi *notification originator* yang memiliki akses ke sebuah instrumen manajemen.

Entitas ini dinamakan agen. Agen dapat didefinisikan juga sebagai sebuah perangkat lunak yang berjalan diatas perangkat yang dimonitor. Perangkat lunak ini dapat berupa sebuah program *daemon* ataupun terintegrasi pada sistem operasi.

2. Satu atau lebih entitas SNMP yang berisi *command generator* dan ataupun aplikasi *notification receiver*.

Entitas ini dinamakan manajer atau sering disebut *Network Management Stations* (NMS). NMS bertanggung jawab untuk melakukan *query* informasi dan menerima *trap* dari agen. *Trap* adalah sebuah mekanisme agar agen dapat melakukan notifikasi ke NMS bahwa sesuatu telah terjadi dan dikirimkan oleh agen secara asinkron.

3. Protokol manajemen yang digunakan untuk mendapatkan informasi antar entitas SNMP

2.1.2. Struktur Manajemen Informasi

Definisi suatu objek yang dikelola dalam sebuah sistem SNMP dapat diuraikan menjadi 3 atribut:

1. Nama

Nama atau *object identifiers* (OID), secara unik mendefinisikan objek yang dikelola dalam sebuah sistem SNMP. Terdapat dua bentuk OID yakni OID numerik yang direpresentasikan dengan bilangan integer, dan OID yang direpresentasikan dengan string atau *human readable* OID.

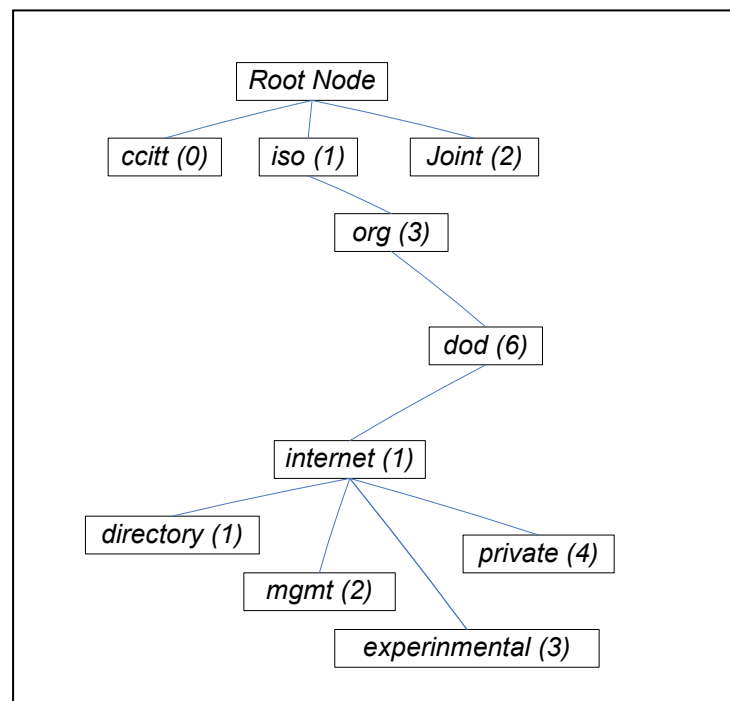
2. Tipe dan sintaks

Tipe data untuk objek yang didefinisikan menggunakan subset dari *Abstract Syntax Notation One* (ASN.1). ASN.1 adalah suatu mekanisme menspesifikasikan representasi transmisi data antara agen dan manajer dalam konteks SNMP. Notasi dari ASN. 1 bersifat *machine-independent*.

3. Encoding

Instansiasi sebuah objek dienkodkan dalam string oktet menggunakan *Basic Encoding Rules* (BER).

Objek diorganisasikan dalam struktur hierarki pohon yang dinamakan pohon *Structure of Management Information* (SMI). Struktur ini adalah dasar dari skema penamaan SNMP. Sebuah objek ID dibentuk dari sekumpulan *integer* berdasarkan simpul dalam pohon tersebut dan dipisahkan dengan titik (.). Gambar II-1 menunjukkan sebuah struktur pohon objek SMI.



Gambar II-1 Pohon SMI

Dalam SNMP, sub-pohon yang menjadi perhatian adalah sub-pohon yang direpresentasikan dengan OID *1.3.6.1* atau sebagai *iso.org.dod.internet*. Setiap objek yang dikelola memiliki numerikal OID dan nama tekstual.

Management Information Base II (MIB-II) adalah sebuah grup manajemen yang sangat penting, karena setiap perangkat yang mendukung SNMP haruslah mendukung MIB-II [MAU01]. *RFC1213-MIB* mendefinisikan OID untuk sub-pohon MIB-II sebagai berikut:

Tabel II-1 MIB-II Group

Sub-pohon	OID	Deskripsi
<i>System</i>	<i>1.3.6.1.2.1.1</i>	Mendefinisikan objek yang berhubungan dengan sistem operasi seperti <i>system uptime</i> , <i>system contact</i> , dan <i>system name</i> .

<i>Interfaces</i>	<i>1.3.6.1.2.1.2</i>	Menyimpan <i>track status</i> untuk setiap antarmuka pada entitas yang dikelola. Grup ini memonitor antarmuka yang berfungsi dan yang tidak.
<i>At</i>	<i>1.3.6.1.2.1.3</i>	Grup <i>address translation (at)</i> bersifat <i>deprecated</i> , disediakan hanya sebagai <i>backward compatibility</i> .
<i>Ip</i>	<i>1.3.6.1.2.1.4</i>	Menyimpan <i>track</i> tentang hal-hal yang berhubungan dengan IP, termasuk di dalamnya adalah <i>IP routing</i> .
<i>Icmp</i>	<i>1.3.6.1.2.1.5</i>	ICMP <i>errors, discards</i> , dsb.
<i>Tcp</i>	<i>1.3.6.1.2.1.6</i>	Menyimpan status koneksi TCP (e.g., <i>closed, listen, synSent, dst</i>).
<i>Udp</i>	<i>1.3.6.1.2.1.7</i>	Menyimpan statistik UDP, datagram yang diterima dan keluar, dsb.
<i>Egp</i>	<i>1.3.6.1.2.1.8</i>	Menyimpan statistik tentang <i>Exterior Gateway Protocol (EGP)</i> dan menyimpan tabel EGP tetangga.

2.1.3. Operasi SNMP

Protocol Data Unit (PDU) adalah format pesan yang digunakan oleh manajer dan agen untuk menerima dan mengirimkan informasi. Terdapat standar format PDU untuk setiap operasi SNMP, akan tetapi hanya beberapa operasi SNMP yang biasa digunakan yakni operasi `get`, `set`, dan `trap`.

Operasi `get` adalah permintaan yang diinisiasi oleh manajer yang dikirimkan ke agen. Agen menerima permintaan tersebut dan memprosesnya selengkap mungkin. Beberapa perangkat jaringan dalam keadaan beban puncak, seperti *router* mungkin tidak dapat merespon permintaan tersebut dan mengabaikannya. Jika agen berhasil mendapatkan informasi yang diminta, agen akan mengirim balasan ke manajer. Contoh eksekusi operasi ini adalah sebagai berikut:

```
$ snmpget cisco.ora.com public .1.3.6.1.2.1.1.6.0
system.sysLocation.0 = ""
```

Perintah di atas menggunakan `snmpget` dengan tiga parameter utama yakni nama *device* yang akan dilakukan *query (cisco.ora.com)*, komunitas string untuk *read-only (public)*, dan OID yang ingin diambil informasinya (*1.3.6.1.2.1.1.6.0*), bentuk *human-readable* dari OID tersebut adalah `sysLocation`.

Operasi set digunakan untuk mengubah nilai objek atau untuk membuat nilai baru dalam sebuah jika berbentuk sebuah tabel. Untuk dapat dikenakan operasi set, objek haruslah bersifat *read-write* atau *write-only*. Contoh berikut adalah perintah menggunakan Net-SNMP untuk mengubah nilai dari variabel *sysLocation*:

```
$ snmpset cisco.ora.com private system.sysLocation.0 s
"Atlanta, GA"
system.sysLocation.0 = "Atlanta, GA"
$ snmpget cisco.ora.com public system.sysLocation.0
system.sysLocation.0 = "Atlanta, GA"
```

untuk dapat mengetahui properti suatu objek, dokumen RFC 1213 dapat digunakan sebagai rujukan.

Agen pada dasarnya hanya dapat dikelola pada operasi-operasi yang terbatas. Untuk melakukan suatu fungsionalitas tambahan di luar spesifikasi standar dari SNMP, sebuah agen dapat diubah menjadi sebuah *extensible agent* dengan mengubah atau menambahkan definisi MIB untuk agen tersebut. Net-SNMP adalah contoh dari sebuah *extensible agent* yang paling sederhana yang akan digunakan pada pengerjaan Tugas Akhir ini.

Beberapa contoh yang ditawarkan oleh Net-SNMP adalah pemeriksaan banyaknya proses yang berjalan menggunakan perintah *proc*, menjalankan *command line* dengan perintah *exec*, dan pemeriksaan penggunaan disk dengan perintah *disk*.

Perintah *exec* dapat digunakan untuk meminta agen menjalankan perintah yang telah terdefinisi untuk agen tersebut. File konfigurasi utama dari Net-SNMP dapat dilihat pada file *snmpd.conf*. Contoh penggunaan *exec* dapat dilihat pada cuplikan file *snmpd.conf* berikut:

```
# Filename: $NET_SNMP_HOME/share/snmp/snmpd.conf
# Return the value from the executed program with a passed parm.
# Items in here will appear in the ucdavis.extTable
exec FileCheck /opt/local/shell_scripts/filecheck.sh
/tmp/vxprint.error
```

Pada contoh di atas, *filecheck.sh* telah didefinisikan untuk menjalankan perintah *ls -al* pada nama file yang dilewatkan sebagai argumen pertama. Detail *filecheck.sh* adalah sebagai berikut:

```
#!/bin/sh
# FileName: /opt/local/shell_scripts/filecheck.sh
if [ -f $1 ]; then
ls -la $1
exit 0
fi
exit 1
```

output yang akan dikembalikan oleh agen akan muncul dalam `ucdavis.extTable (.1.3.6.1.4.1.2021.8)`. Contoh dari pemanggilan menggunakan `snmpget` dari script tersebut adalah sebagai berikut:

```
enterprises.ucdavis.extTable.extEntry.extOutput.1 = \
" 16 -rw-r--r-- 1 root other 2476 Feb 3 17:13
/tmp/vxprint.error."
```

Dari kemampuan untuk melewati sebuah script ke agen yang dimonitor, sistem dapat melakukan kontrol terhadap agen seperti halnya mengetikkan perintah pada lokal terminal agen.

2.3. Gnokii SMS Gateway

Gnokii adalah sebuah perangkat yang menyediakan *user space* modem *driver* untuk telepon genggam sekaligus perangkat pengaksesannya, yang dikeluarkan dengan *GNU Public License* (GPL) untuk *platform* Linux, MacOS, dan Win32 [GNO08]. Tujuan Gnokii adalah menyediakan *device driver* untuk mendukung perangkat mobil Nokia ataupun perangkat mobil lain yang mendukung *AT command*. Mulai versi 0.6.1, Gnokii mulai mendukung perangkat mobil berbasis Symbian.

Proyek awal Nokia 3810/3110/8110 dicetuskan dari diskusi antara Francois Dessart dan Hugh Blemings. Tujuan utama dari proyek ini adalah mengganti perangkat lunak Nokia *Cellular Data Suite* (NCDS) yang berjalan di Linux. Proyek serupa dikembangkan oleh Staffan Ulfberg untuk menyediakan perangkat lunak bagi Nokia 6110 dan model telepon genggam yang serupa. Proyek ini diorientasikan untuk mendukung *platform* yang tidak didukung oleh NCDS. Pada akhir Februari 1999, kedua proyek tersebut digabung menjadi satu dibawah nama proyek Gnokii. Tujuan utama penggabungan ini adalah menghindari duplikasi usaha pembuatan perangkat lunak yang sama dan untuk menyediakan satu *milist* untuk berbagi informasi mengenai telepon dalam paradigma *open-source*. Pemimpin proyek pengembangan Gnokii saat ini adalah Pawel Kot dan Borbely Zoltan. [GNO08]

Terdapat *Perlmodule* yang tersedia untuk Gnokii. Module tersebut memberikan akses ke semua fungsi yang terdefinisi dalam `gsm-common.h` pada Gnokii. Gnokii juga memfasilitasi *command line* yang dapat langsung mengakses fungsionalitas Gnokii. Pada umumnya Gnokii digunakan untuk membaca ataupun menulis daftar buku telepon, dan mengirim ataupun menerima SMS. Contoh penggunaan Gnokii untuk mengirimkan sebuah SMS adalah sebagai berikut:

```
#!/usr/bin/perl
use strict;

my $message = "test sending sms using gnokii";
my $destination = "08157534221";
my $to_exec = "echo $message | /usr/local/bin/gnokii --sendsms
$destination ";
exec($to_exec);
```

2.4. JpGraph *Graph Plotter*

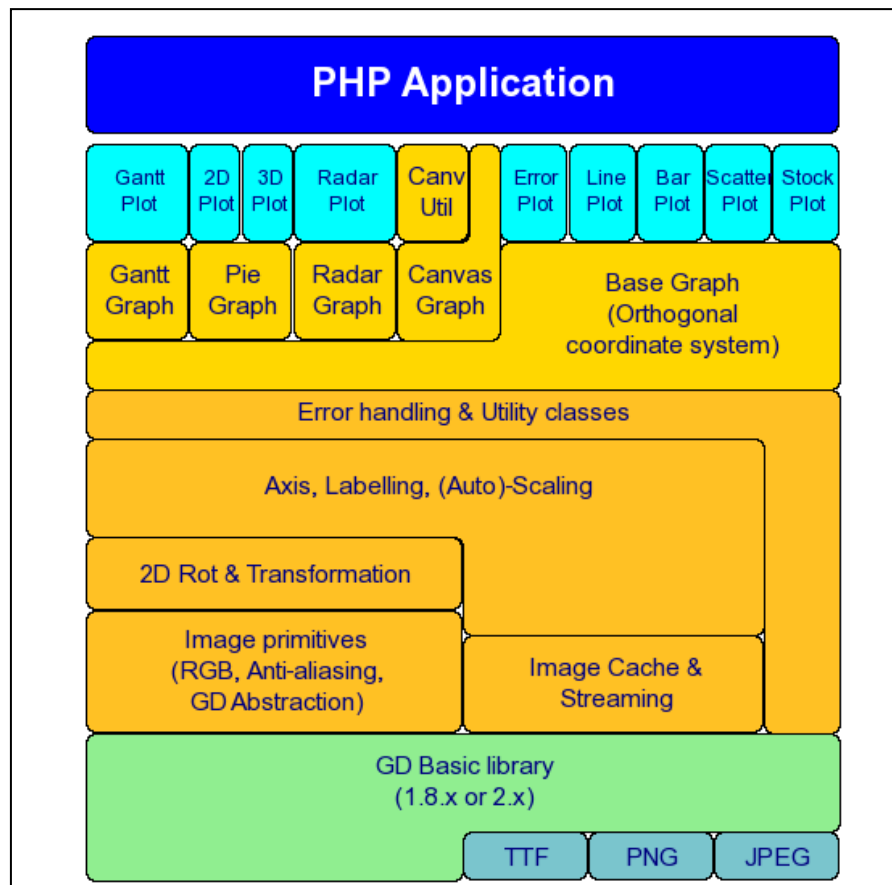
Jpgraph adalah pustaka PHP yang dibangun dengan berorientasikan objek untuk menciptakan grafik dalam format PNG, GIF, dan JPEG dengan lisensi QPL 1.0 (*Qt-License*) [ADI05]. Seluruh pustaka ditulis dalam PHP dan dapat langsung digunakan untuk script PHP apapun. JpGraph mensyaratkan PHP minimal dengan versi 4.3.1 dan pustaka GD baik versi 1 ataupun 2. JpGraph memungkinkan untuk menggambar grafik dengan pendekatan “*quick and dirty*” dengan kode yang minimum dan mudah ditelaah.

JpGraph bersifat independen terhadap sistem operasi, karena pustaka ini mampu mendiagnosa sistem operasi yang digunakan.

Beberapa fitur dari JpGraph adalah sebagai berikut:

- Rata-rata ukuran *file* < 5Kb, dengan demikian cukup ringan untuk sebuah dokumen web.
- Kompatibel dengan pustaka GD versi 1 dan 2. Secara otomatis JpGraph akan mendeteksi versi dari pustaka GD yang digunakan.
- *Advanced interpolation* untuk menghasilkan kurva yang halus.
- Mendukung bermacam tipe plot.
- Mendukung *internal caching* sehingga tidak membebani *web server* yang digunakan.

Arsitektur pustaka JpGraph dapat dilihat pada Gambar II-2.

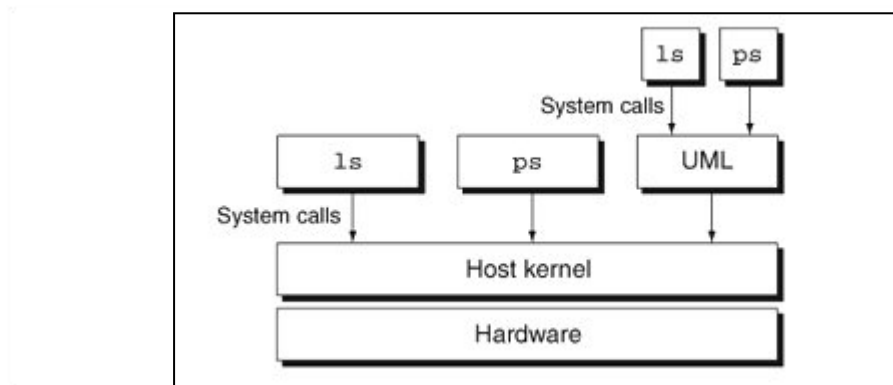


Gambar II-2 Arsitektur Pustaka JpGraph

2.5 Virtual Network - User Mode Linux (VNUML)

User Mode Linux (UML) adalah virtual mesin Linux yang berjalan di atas Linux. Secara teknis, UML adalah hasil porting Linux ke Linux [DIK06]. UML berjalan menggunakan satu media penyimpanan yang keseluruhannya terletak pada satu file pada mesin fisik. Satu instansiasi UML dapat diberikan satu atau lebih akses virtualisasi. Dengan pemberian akses yang benar, pengguna yang menjalankan UML tidak akan merubah ataupun merusak komputer fisik.

Interaksi UML dengan mesin fisik dapat dilihat pada Gambar II-3.



Gambar II-3 Interaksi antara Instansiasi UML, Host Kernel, dan proses UML

Virtual Network User Mode Linux (VNUML) adalah perangkat virtualisasi *open source* yang dirancang untuk dapat secara cepat mendefinisikan dan menguji skenario jaringan yang kompleks berdasarkan UML [FER06].

VNUML terdiri dari dua komponen utama; bahasa VNUML untuk mendefinisikan simulasi dalam XML dan sebuah interpreter (perintah `vnuml`) yang membangun dan mengelola simulasi. Dengan demikian VNUML mempermudah pengguna sehingga tidak perlu tahu detail kompleks cara kerja UML.

2.6 Hallway *Usability Testing*

Usability testing adalah teknik yang digunakan dalam perancangan untuk meningkatkan kualitas produk dari segi *usability* atau *user-friendliness* [DUM99]. *Usability testing* secara umum digunakan untuk mengukur tingkat keberhasilan objek yang diuji dalam 4 kriteria, yakni:

1. Waktu

Pengujian dilakukan dengan mengukur banyaknya waktu yang digunakan untuk menyelesaikan sebuah operasi dasar secara lengkap.

2. Akurasi

Pengujian dilakukan dengan mengukur banyaknya kesalahan yang dilakukan pengguna dan respons sistem terhadap kesalahan tersebut.

3. *Recall*

Pengujian dilakukan dengan mengukur seberapa banyak pengguna mengingat operasi yang telah dilakukan setelah n period tidak melakukan operasi tersebut.

4. Respons emosional.

Pengujian dilakukan dengan mengukur emosi pengguna setelah menyelesaikan satu tugas. Emosi yang diukur misalnya, tertekan, keyakinan keberhasilan sistem, dsb.

Hallway testing adalah metodologi spesifik yang digunakan untuk *usability testing* perangkat lunak yang mengambil lima sampai enam pengguna secara random. Perlakuan random ini mengindikasikan bahwa pengujian mengambil latar belakang pengguna yang berbeda-beda. Nama dari teknik ini merujuk pada awal pengujian bahwa pengguna diambil secara random dari *hallway*. Secara teori, teknik ini berdasarkan pada riset Jakob Nielsen bahwa 95% dari permasalahan *usability* dapat ditemukan dengan teknik ini [NIE00].

Rata-rata waktu yang digunakan untuk aksi antarmuka komputer dapat dilihat pada tabel II-2 [OLS90].

Tabel II-2 Rata-Rata Waktu Aksi Antarmuka Komputer

Aksi	Waktu (detik)	Keterangan
Menekan satu <i>keyboard</i>	0,28	Bervariasi dari 0,07 detik untuk yang mahir mengetik, 0,2 detik untuk tipikal pengguna, dan 1 detik untuk pengguna yang buruk
Menggunakan <i>mouse</i> untuk <i>pointing</i>	1,5	
Perpindahan <i>mouse</i> dan <i>keyboard</i>	0,3	
Respon ke warna terang	0,1	Bervariasi dari intensitas cahaya
Mengambil perintah sederhana dari ingatan jangka panjang	1,2	Contoh mengingat fungsi <i>dir</i>
Belajar suatu langkah dalam prosedur	25	Penelitian menunjukkan, 10 dan 15 adalah waktu minimum
Pemilihan di antara method yang ada	1,2	Bervariasi dari 0,06 hingga 1,8 detik bergantung dari kerumitan

2.7 Design Pattern

Design pattern didefinisikan sebagai solusi perancangan yang berulang pada domain permasalahan yang umum ditemui pada perancangan perangkat lunak [SHA04]. *Design pattern* membantu developer perangkat lunak untuk selalu berpikir *reusable* dan sebagai

notasi baru untuk berkomunikasi dengan developer yang lain. Selain itu, *design pattern* juga membantu memberikan perspektif yang lebih tinggi pada tahapan analisa dan perancangan.

Beberapa *pattern* yang digunakan adalah *singleton pattern*, *template factory*, *observer pattern*, dan *strategy pattern*.

Singleton pattern memastikan bahwa kelas hanya memiliki satu instansiasi dan harus menyediakan fungsi global untuk mengaksesnya [GAM95]. Contoh penggunaan *pattern* ini adalah kelas pengaksesan basis data.

Template factory memastikan adanya standardisasi pada pembangunan kelas. Tujuan dari *pattern* ini adalah mendefinisikan antarmuka untuk membangun suatu objek, dengan mendelegasikan sub kelas untuk memutuskan kelas yang akan diinstansiasi [GAM95].

Tujuan dari *observer pattern* adalah untuk mendefinisikan relasi kebergantungan antara banyak objek sehingga ketika status objek berubah semua objek yang bergantung dengannya secara otomatis menerima notifikasi dan ikut berubah [GAM95].

Jika masing-masing kelas memiliki abstraksi yang sama namun dengan implementasi algoritma yang berbeda, maka *strategy pattern* adalah solusi yang tepat untuk permasalahan tersebut. Tujuan dari *strategy pattern* adalah mendefinisikan kelompok besar algoritma, membungkus tiap algoritma tersebut dan membuatnya dapat saling bertukar [GAM95].