IMPLEMENTASI POLA DESAIN UNTUK PERANCANGAN SISTEM INFORMASI PERPUSTAKAAN

Oleh:

Jason Pratama Sunarji[1*], Mohammad Fajar2, Junaedy3

Sistem Informasi, STMIK Kharisma Makassar

Abstrak: Penelitian ini bertujuan untuk mengimplementasikan pola desain pada sistem informasi perpustakaan dan menganalisis apakah penggunaan pola desain mudah untuk digunakan kembali. Pola desain yang diterapkan terdiri dari *Factory Pattern* dan *Façade Pattern*. *Factory Pattern* digunakan untuk memisahkan proses pembuatan sebuah objek dari objek lain yang menggunakannya. *Façade Pattern* digunakan untuk mengurangi keterhubungan antar kelas dan mengurangi kompleksitas sistem yang dibangun. Untuk mengevaluasi pola desain digunakan object-oriented metrics baik pada sistem yang tidak menerapkan pola desain maupun sistem yang diusulkan (menggunakan pola desain). Hasil analisis Object Oriented Metrics menggunakan parameter Coupling Between Object (CBO) memperlihatkan pengurangan keterhubungan antar kelas sehingga membuat sistem mudah untuk di reusable, dengan parameter Response For a Class (RFC) memperlihatkan pengurangan method di dalam kelas pada sistem yang menerapkan pola desain sehingga membuat sistem mudah untuk di mengerti dan di testing. Selain itu menggunakan parameter Line of Code (LoC) menunjukkan pengurangan jumlah baris program sebesar 228 baris kode, dari 4410 menjadi 4182 sehingga sistem tersebut membuat ukuran program menjadi lebih kecil dan membuat sistem informasi perpustakaan mudah untuk di maintenance, di reuse dan dimengerti.

Kata kunci : Pola Desain, Sistem Informasi Perpustakaan, *Factory Pattern*, Facade Pattern, Object Oriented Metrics.

Abstract: *The aims of this study are to apply design patterns for developing library information system and analyze them from re-usability aspect. The applied design pattern consists of Factory Patterns and Façade Patterns. Factory Patterns are used to separate the process of making objects from other purchased objects. The Façade Patterns are used to reduce connectivity among classes and the complexity of the system being built. To evaluate the patterns, object-oriented metrics were used, both system that do not apply the design pattern and the proposed system (system that apply the design pattern). The results of the Object Oriented Metrics analysis using the Coupling Between Object parameter showed the decrease of the connectivity among classes so that the system is easy to reuse, in which the parameter Response for a Class (RFC) showed the reduction of the method in the class on a system that applied the design pattern which made it easier to be understood and tested. In addition, using the parameter Line of Code (LoC) showed the reduction of 228 lines code of program, from 4410 to 4182 so that the system made the program size smaller and the library information system become easier to be maintained, reused, and understood.*

Keywords : Design Patterns, Library Information Systems, Factory Pattern, Facade Pattern, Object Oriented Metrics

[*]Corresponding author : Jason Pratama Sunarji (jasonpratama_16@kharisma.ac.id)

PENDAHULUAN

Telah banyak studi yang membahas tentang pengembangan sistem informasi perpustakaan yang sejenis namun berbeda hanya pada tempat seperti (Wafiroh, Mulyono, & Hutabri), (Hendrianto, 2014), (Ermatita, 2016), (Kasmirin, Yusman, & Adipribadi, 2016), (Firdaus, Sakethi & Rosman, 2015). Akan tetapi, studi – studi ini hanya membahas pengembangan sistem informasi perpustakaan secara ad-hoc tanpa menggunakan teknik atau pendekatan pengembangan sistem atau rekayasa sistem. Teknik atau pengembangan dibutuhkan untuk menangani kompleksitas sistem (Sommerville, 2007) dan menyederhanakan proses konfigurasi sistem apabila sistem informasi perpustakaan yang telah dikembangkan di suatu tempat ingin diterapkan di tempat lain. Selain itu, tanpa pemakaian teknik atau metode pengembangan sistem, proses analisa yang sesuai dengan kebutuhannya dan proses implementasi jika ingin dikembangkan sering kali membuat seseorang bingung dengan kode yang telah dibuatnya sehingga membutuhkan waktu yang lama untuk memahami alur prosesnya dan bahkan kode program yang telah dibuat tidak bisa digunakan kembali dan adanya kompleksitas di dalam sistem yang dibuat sehingga membuat programmer menjadi kesulitan untuk melakukan maintenance sistem yang sudah dikembangkan. Keterhubungan antar kode menjadikan kode tersebut sangat rumit dan sulit untuk dipahami. Kerumitan biasanya terjadi saat sistem akan menjalankan suatu fungsionalitas. Pemanggilan fungsionalitas dilakukan melalui kelas GUI, kelas GUI akan memanggil method yang berada di kelas logik. Pemanggilan method dilakukan berulang kali, sehingga terjadi banyak hubungan antara kelas-kelas GUI dan kelas-kelas logik.

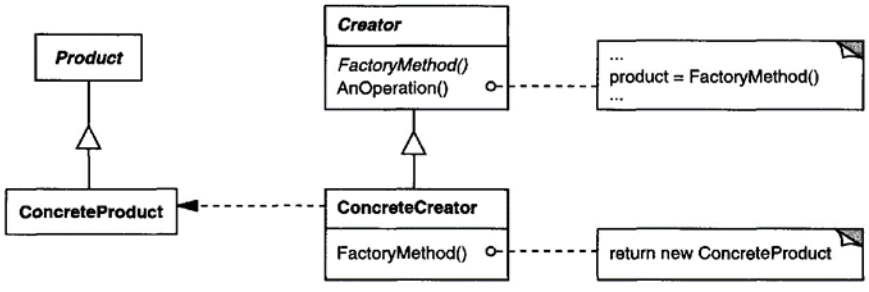
LANDASAN TEORI

1. Pola Desain

Pola desain adalah sebuah solusi terhadap masalah-masalah umum dalam rekayasa perangkat lunak yang dapat digunakan berulang kali. Pola desain ini merupakan sebuah template yang menunjukkan bagaimana sebuah masalah diselesaikan dan dapat digunakan kembali dalam situasi yang berbeda. (Gamma, 1994)

2. *Factory Pattern*

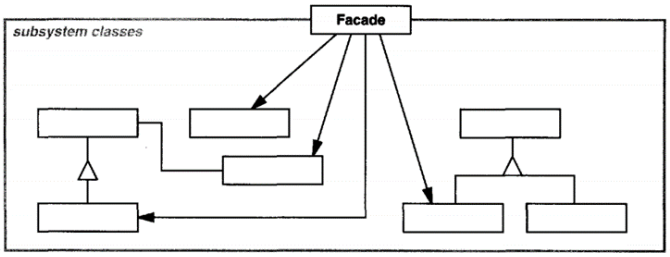
Pola Pabrik adalah pola yang menentukan antarmuka untuk membuat objek, tetapi membiarkan subclass menentukan kelas mana yang akan dipakai. Pola Pabrik memungkinkan menunda instansiasi kelas ke subkelas. (Gamma, 1994)



Gambar 1 Struktur *Factory Pattern*

3. *Façade Pattern*

Pola Fasad adalah sebuah pola desain yang menyembunyikan kompleksitas sistem dan menyediakan antarmuka yang telah disederhanakan kepada klien untuk mengakses sistem. (Gamma, 1994)



Gambar 2 Struktur *Facade Pattern*

4. Object Oriented Metrics

a. *Coupling Between Object* (CBO)

Coupling between object adalah jumlah kelas yang terhubung atau berpasangan, tanpa terdapat inheritance di dalamnya atau dapat dikatakan jika suatu method atau atribut mengakses method atau atribut lain dari kelas lainnya. Kondisi *couple* terjadi apabila *method* pada suatu kelas dibentuk dari *method* kelas lainnya. Pengukuran *Coupling Between Object* (CBO) dapat menunjukkan tingkat kompleksitas dan *reuse* dari suatu program.

b. Response For a Class (RFC)

Response For a Class adalah *metrics* yang digunakan untuk menghitung banyaknya jumlah *method* yang digunakan oleh suatu kelas. RFC dapat diartikan serangkaian metode yang dapat berpotensi dieksekusi sebagai respon terhadap pesan yang diterima oleh objek dari kelas itu (Aggarwal, Singh, Kaur, & Malhotra, 2006). Perhitungan RFC dapat dirumuskan dengan rumus berikut ini.

$$\mathrm{RS} = M_i \cup_{all\ j} \{R_{ij}\}$$

RS = *Response For a Class*

Mi = *Method* suatu kelas yang digunakan oleh kelas itu sendiri,

Rij = *Method* dari kelas lain yang digunakan oleh suatu kelas

Contoh:

A:a1( ) memanggil B:b2( ), A:a2( ) memanggil A:a1( ), A:a3( ) memanggil B:b3( )

RFC = {A:a1, A:a2: A:a3} U {B:b2} U {A:a1} U {B:b3}

= {A:a1, A:a2, A:a3, B:b2, B:b3} = 5

c. Line of Code (LoC)

Line of Code (LOC), adalah banyaknya baris program yang menyusun suatu aplikasi. Semua baris program akan dihitung kecuali *comment* dan baris yang kosong. Berkurangnya jumlah baris kode implementasi suatu program, maka semakin efisien (tepat guna) program tersebut.

ANALISIS DAN DESAIN SISTEM

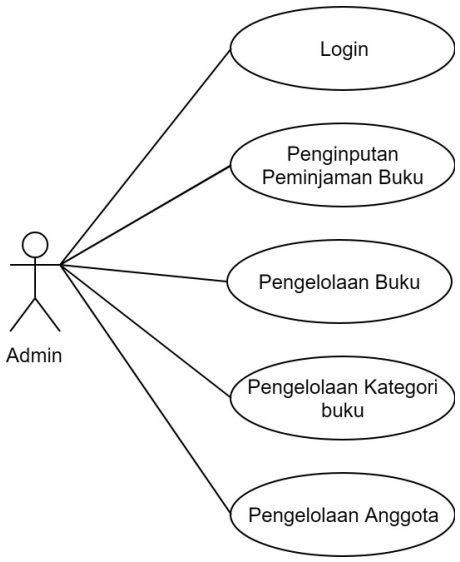
1. Analisis Kondisi Awal

Analisis yang dilakukan adalah analisis terhadap sistem pada suatu perpustakaan. Untuk mendapatkan kebutuhan fungsional sistem perpustakaan dilakukan wawancara pada admin perpustakaan STMIK Kharisma Makassar. Hasil dari wawancara dapat didefinisikan bahwa sistem perpustakaan harus memiliki proses penginputan data buku teks, penginputan data anggota, penginputan peminjaman hingga setting pada perpustakaan. Terdapat dua buah sistem yang akan dirancang pada tugas akhir ini dimana bertujuan untuk mendapatkan hasil evaluasi yang diharapkan. Sistem-sistem tersebut sebagai berikut:

a. Sistem perpustakaan tanpa *Factory Pattern* dan *Façade Pattern*
b. Sistem perpustakaan menggunakan *Factory Pattern* dan *Façade Pattern*

2. Rancangan Sistem

a. Usecase Sistem Informasi Perpustakaan



Gambar 1 *Usecase* Sistem Informasi Perpustakaan

Tabel 1 Tabel Identifikasi Aktor

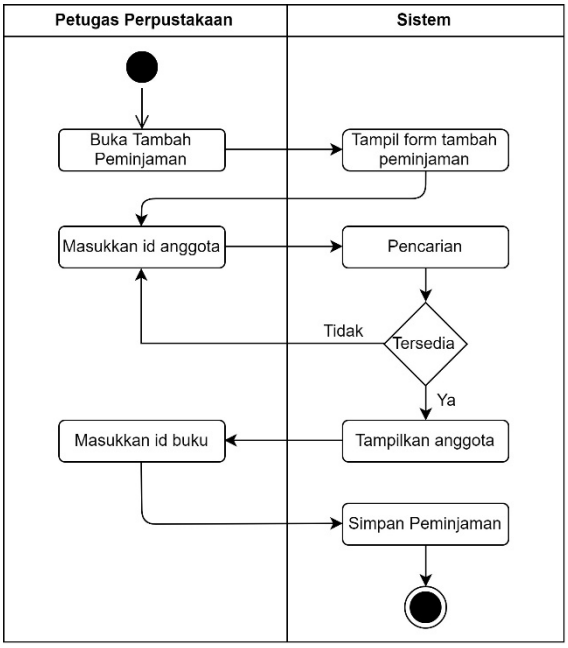
No	Nama Aktor	Deskripsi
1	Admin	Orang yang akan mengelola/menggunakan sistem informasi perpustakaan dengan diberikan akses yang full untuk semua fitur yang ada pada sistem.

Tabel 2 Tabel Identifikasi Use Case

No	*Use Case*	Deskripsi	Aktor
1	*Login*	Pada *use case* ini, admin akan melakukan proses *login* terlebih dahulu untuk dapat menggunakan sistem perpustakaan.	admin
2	Pengelolaan Kategori Buku	Pada *use case* ini, admin akan melakukan pengelolaan kategori buku yang ada di perpustakaan untuk disimpan ke dalam sistem dimana use case ini terdapat proses simpan,	admin

No	*Use Case*	Deskripsi	Aktor
		tampil, dan hapus kategori buku.	
3	Pengelolaan Buku	Pada *use case* ini, admin akan melakukan pengelolaan data buku yang ada di perpustakaan untuk disimpan ke dalam sistem dimana *use case* ini terdapat proses simpan, tampil, edit dan hapus buku.	admin
4	Pengelolaan Anggota	Pada *use case* ini, admin akan melakukan pengelolaan data anggota yang ada di perpustakaan untuk disimpan ke dalam sistem dimana *use case* ini terdapat proses simpan, tampil, edit dan hapus anggota.	admin
5	Penginputan Peminjaman buku	Pada *use case* ini, admin akan melakukan penginputan data buku yang ingin dipinjam oleh anggota. Pada proses ini admin akan menginput data buku dan input anggota yang akan meminjam.	admin

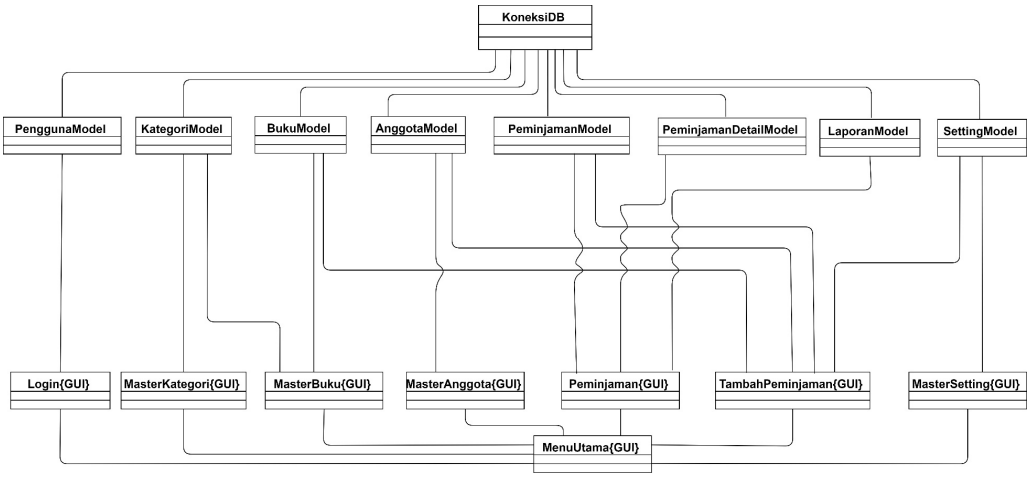
b. Activity Diagram Tambah Peminjaman



Gambar 3 *Activity Diagram* Tambah Peminjaman

Pada gambar 2, menjelaskan bahwa petugas perpustakaan saat membuka form tambah peminjaman akan menampilkan tampilan form tambah peminjaman. Setelah itu petugas perpustakaan akan memasukkan id anggota yang ingin meminjam buku dan akan dicek dari sistem apakah id anggota telah terdaftar atau belum. Jika belum maka akan kembali memasukkan id anggota yang telah ada, jika tersedia akan menampilkan nama anggota dan petugas perpustakaan akan memasukkan id buku yang akan dipinjam serta akan menyimpan data ke dalam sistem.

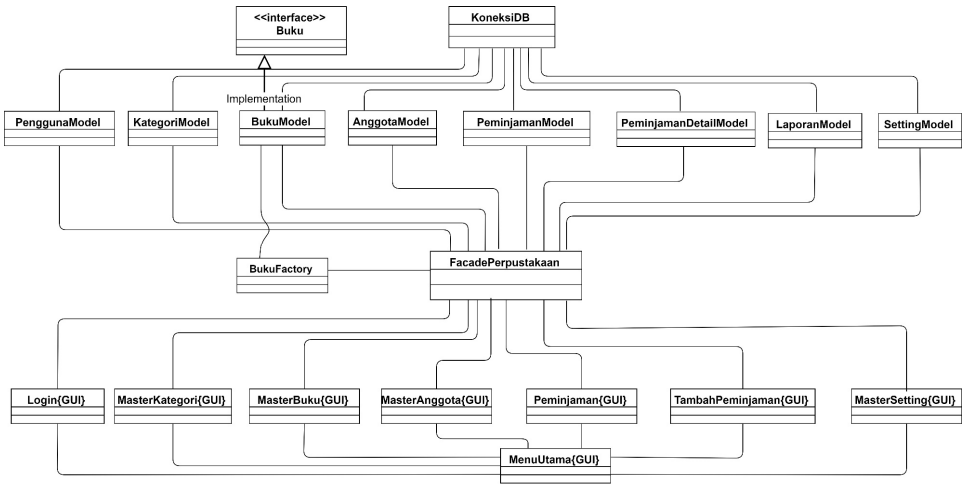
3. Perancangan Sistem Informasi Perpustakaan Tanpa Desain Pattern



Gambar 4 *Class Diagram* Sistem Perpustakaan Tanpa *Design Pattern*

Pada gambar 3, terdapat banyak keterhubungan antara kelas logik dan kelas GUI untuk pembuatan sistem perpustakaan. Banyaknya keterhubungan menjadikan sistem semakin rumit dan sulit untuk digunakan kembali apabila ada penambahan fitur yang diinginkan dalam sistem. Kerumitan tersebut akan juga mengakibatkan sistem menjadi lebih kompleks dan membutuhkan waktu yang sangat banyak untuk *maintainance* jika terdapat kerusakan pada sistem.

4. Perancangan Sistem Informasi Perpustakaan Menggunakan Desain Pattern



Gambar 5 *Class Diagram Sistem Perpustakaan Menggunakan Design Pattern*

Pada gambar 4, terdapat sebuah kelas baru yaitu kelas FacadePerpustakaan yang merupakan sebuah kelas façade dan terdapat kelas Buku, kelas BukuFactory yang merupakan sebuah kelas factory. Pada kelas BukuFactory ini akan digunakan untuk proses penciptaan objek-objek buku seperti pada gambar tersebut sehingga mengurangi tinggal keterhubungan di dalam proses penciptaan objek baru. Pada kelas

FacadePerpustakaan sangat bermanfaat untuk sebagai perantara antara kelas logic dan kelas GUI sehingga mengurangi keterhubungan langsung dengan bagian GUI dan juga proses perubahan menjadi lebih mudah karena perubahan terjadi pada kelas FacadePerpustakaan bukan pada kelas GUI. Dari hasil penerapan *Factory Pattern* dan *Façade Pattern*, membuat sistem informasi perpustakaan menjadi lebih mudah untuk di lakukan penambahan atau proses maintenance apabila terjadi perubahan sehingga tidak memakan waktu yang lama untuk membuatnya. Class diagram tersebut juga akan menjadi lebih mudah untuk dimengerti oleh bagian programmer karena telah mengetahui alur-alur yang ada di dalam sistem informasi perpustakaan.

PENGUJIAN SISTEM

1. Perhitungan dan Analisis Coupling Between Object (CBO)

ada beberapa kelas yang mengalami penurunan pada nilai metriknya pada kelas yang ada pada sistem informasi perpustakaan, yaitu:

Tabel 3 Perhitungan *Coupling Between Object* (CBO)

Nama Kelas	Tanpa Design Pattern	Menggunakan Design Pattern	Selisih
MasterBuku	3	2	1
Peminjaman	6	2	4
TambahPeminjaman	5	2	3

Dengan melihat nilai CBO pada kedua sistem perpustakaan, menunjukkan bahwa nilai CBO setelah menerapkan *Factory Pattern* dan *Façade Pattern* memiliki jumlah yang lebih kecil dibandingkan dengan sistem perpustakaan yang tidak menerapkan *Factory Pattern* dan *Façade Pattern*. Hal ini menunjukkan bahwa dengan menggunakan *Façade Pattern* pada sistem perpustakaan memiliki keterhubungan dengan kelas lain yang lebih sedikit dibandingkan dengan sistem perpustakaan yang tidak menerapkan *Façade Pattern*. Dari hasil tersebut dapat menarik kesimpulan bahwa sistem informasi perpustakaan mudah untuk *reuse* karena kurangnya keterkaitan yang banyak dalam suatu kelas.

2. Perhitungan dan Analisis *Response For a Class* (RFC)

Tabel 4 Perhitungan *Response For a Class (RFC)*

Nama Kelas	Tanpa Design Pattern	Menggunakan Design Pattern	Selisih
MasterAnggota	13	11	2
MasterBuku	17	15	2
MasterKategori	10	7	3
MenuUtama	9	8	1
Peminjaman	10	8	2
TambahPeminjaman	25	17	8

Dengan melihat nilai RFC pada kedua sistem perpustakaan, menunjukkan bahwa nilai RFC setelah menerapkan *Factory Pattern* dan *Façade Pattern* memiliki jumlah yang lebih kecil dibandingkan dengan sistem perpustakaan yang tidak menerapkan *Factory Pattern* dan *Façade Pattern*. Hal ini menunjukkan bahwa dengan menggunakan *Façade Pattern* pada sistem perpustakaan memiliki penggunaan *method* yang lebih sedikit dibandingkan dengan sistem perpustakaan yang tidak menerapkan *Façade Pattern*. Dari hasil tersebut menarik kesimpulan bahwa sistem informasi perpustakaan mudah untuk dimengerti atau understandability.

3. Perhitungan dan Analisis *Line of Code* (LoC)

Tabel 5 Perhitungan *Line of Code (LoC)*

Tanpa Design Pattern	Menggunakan Design Pattern	Selisih
4410	4182	828

Terdapat jumlah baris kode dari sistem informasi perpustakaan yang tidak menggunakan design pattern sebanyak 4410 baris kode yang lebih besar dibandingkan dengan yang menggunakan design pattern memiliki jumlah baris kode sebanyak 4182 sehingga terjadi penyusutan kode sebanyak 228 baris kode. Hal ini berarti bahwa pada perangkat lunak yang sudah menerapkan *Factory Pattern* dan *Façade Pattern* memiliki ukuran program yang lebih kecil jika dibandingkan dengan perangkat lunak sebelum diterapkan design pattern tersebut dan juga membuat sistem tersebut menjadi mudah untuk di maintainance dan juga efisien dalam peyusunan kode program.

KESIMPULAN

Berdasarkan hasil pembahasan dan informasi yang diperoleh, maka penulis dapat menarik beberapa kesimpulan:

1. Telah diimplementasikan pola desain yang terdiri dari *Factory Pattern* dan *Façade Pattern*. *Factory Pattern* digunakan untuk memisahkan proses pembuatan objek dari objek lain yang menggunakannya. *Façade Pattern* digunakan untuk mengurangi keterhubungan antar kelas dan mengurangi kompleksitas sistem informasi perpustakaan yang dibangun. Pemetaan dari proses perancangan desain sistem ke implementasi dapat dilakukan dengan melihat artifak *class diagram* untuk pemetaan kode kelas, mekanisme penggunaan pola desain juga dapat diterapkan ke dalam bahasa pemrograman Java.
2. Dari evaluasi menggunakan *Object Oriented Metrics*, sistem informasi perpustakaan yang menerapkan *Factory Pattern* dan *Façade Pattern* menggunakan parameter CBO menunjukkan pengurangan keterhubungan antar kelas sehingga membuat sistem mudah untuk di *reusable*, dengan parameter RFC memiliki nilai yang lebih kecil dibandingkan dengan sistem yang tidak menerapkan *design pattern* tersebut disebabkan karena terjadi pengurangan pada method di dalam kelas sehingga sistem perpustakaan mudah untuk di mengerti dan di *testing*, dan menggunakan parameter LoC menunjukkan pengurangan jumlah baris program sebesar 228 baris kode, dari 4410 menjadi 4182 sehingga membuat ukuran program menjadi lebih kecil dan sistem mudah untuk di *maintenance*, di *reuse* dan di mengerti.

DAFTAR PUSTAKA

[1] Aggarwal, K., Singh, Y., Kaur, A., & Malhotra, R. (2006). Empirical Study of Object-Oriented. *Journal Of Object Technology*, 1490173.

[2] Ermatita. (2016). Analisis Dan Perancangan Sistem Informasi Perpustakaan. *Jurnal Sistem Informasi (JSI)*, 966-977.

[3] Firdaus, R., Sakethi, D., & Rosman, F. (2015). Rancang Bangun Sistem Informasi Perpustakaan Berbasis Web. *Jurnal Komputasi*, 85-94.

[4] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Pattern : Element of Reusable Object-oriented.* Addison Wesley.

[5] Hendrianto, D. E. (2014). Pembuatan Sistem Informasi Perpustakaan Berbasis Website. *Indonesian Journal on Networking and Security*, 57-64.

[6] Kasmirin, A. R., Yusman, M., & Adipribadi, I. (2016). Perancangan Sistem Informasi Perpustakaan Berbasis Web (Studi Kasus Sman 1 Penengahan). *Jurnal Komputasi*, 104-108.

[7] Lukman, A. M. (2017). Pengembangan Sistem Informasi Perpustakaan Umum Berbasis Web Menggunakan Inlislite 3.0. *ILKOM Jurnal*, 70-77.

[8] Sommerville, I. (2007). *Software Engineering.* New York: Addison-Wesley.

[9] Wafiroh, Mulyono, H., & Hutabri, E. (n.d.). Perancangan Sistem Informasi Perpustakaan