# Rancang Bangun Arsitektur Modular Album Foto Digital 'Fotokita' berbasis *Desktop*

A. Heynoum Dala Rif'at, Rizky Januar Akbar, dan R.V. Hari Ginardi
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
*e-mail*: rizky@if.its.ac.id

***Abstrak*—Seiring dengan fenomena semakin maraknya aktifitas pengabadian momen, dibutuhkannya teknologi yang memadai untuk mendukung aktifitas tersebut. Beberapa solusi telah ditawarkan oleh percetakan atau studio foto untuk menawarkan jasa membuat album foto. Tetapi di Indonesia belum ada aplikasi album digital berbasis *desktop* yang dapat memudahkan pengguna secara langsung untuk membuat album fotonya sendiri.**

**Pada makalah ini, permasalahan tersebut akan ditangani dengan membuat album foto digital yang bersifat modular yang dapat mengakomodasi kebutuhan pengguna dan ketidakpastian perkembangan fitur di masa depan, misalnya penambahan *frame,* efek dan *tools* pada aplikasi. Perangkat lunak ini bersifat *desktop* dan terdapat abstraksi/aturan yang perlu diimplementasikan agar modul dapat diintegrasi. Perangkat lunak akan dapat menambah, menghapus instalasi atau mengubah status modul-modul dari album foto digital tersebut tanpa melakukan perubahan pada modul lain.**

**Pengujian dilakukan dengan melakukan penambahan, penghapusan instalasi dan pengubahan status modul. Dari hasil pengujian, aplikasi yang dirancang dan diimplementasikan telah memenuhi semua kebutuhan fungsional.**

***Kata Kunci*—Album Foto Digital, Perangkat Lunak Modular**

## I. PENDAHULUAN

PERKEMBANGAN teknologi di bidang digital sudah berkembang dengan sangat pesat, terutama dalam bidang fotografi. Dalam hal mencetak foto, yang dulu harus menggunakan film dan kemudian mencetaknya melalui proses yang rumit, sekarang bisa dilakukan dengan sangat mudah dengan memindahkan berkas yang ada di dalam kartu memori ke dalam komputer dan mencetaknya pun hanya membutuhkan sebuah mesin cetak (*printer*) dan kertas foto, dan foto-foto tersebut lalu diletakkan ke dalam album foto. Proses tersebut pada saat ini dapat dilakukan dengan menggunakan komputer melalui aplikasi.

Seiring dengan fenomena semakin maraknya pengabadian momen, dibutuhkannya teknologi yang memadai untuk mendukung kebutuhan. Beberapa percetakan atau studio foto sudah menyediakan layanan untuk melakukan pengeditan foto dan album, tetapi terkadang hasil yang ada tidak sesuai dengan harapan klien dan memakan biaya lebih. Solusi untuk klien yang menginginkan untuk mencetak foto album sesuai dengan selera masing-masing sudah tersedia, tetapi untuk di Indonesia sendiri sarana yang tersedia masih berupa pengeditanalbum foto secara *online*, sedangkan kecepatan jaringan belum bisa dikatakan mampu untuk melakukan semua kegiatan secara *online*.

Aplikasi album foto digital berbasis *desktop* diperlukan untuk memudahkan pengerjaan tersebut. Selain itu, untuk dapat menarik perhatian pengguna, sebuah aplikasi harus mengikuti perkembangan zaman dan mode, dan dikarenakan adanya ketidakpastian perkembangan fitur, misalnya penambahan *frame, layout*, efek dan tema sesuai dengan kebutuhan dan selera pengguna, maka dibutuhkan aplikasi album foto digital yang modular, dimana nantinya modul-modul dalam album foto digital dapat diperbaharui, dihapus, atau diganti dengan mudah.

Tujuan dikembangkannya aplikasi ini adalah untuk memberikan solusi kemudahan dalam melakukan evolusi dan modularitas sistem, sehingga dapat mengakomodasi kebutuhan pengguna kedepannya.

## II. TINJAUAN PUSTAKA

### A. Album Foto Digital

Album foto digital adalah sekumpulan foto yang dibukukan dalam suatu wadah. Album foto digital yang digital adalah aplikasi perangkat lunak dimana pengguna dapat memindahkan berkas foto dari hardisk ke dalam basis data utama aplikasi tersebut. Perangkat lunak Album foto digital biasanya memungkinkan pengguna untuk melihat, mengubah, dan mengatur foto dengan menggunakan antarmuka buku seperti yang menyerupai album foto tradisional.

Pada saat ini di Indonesia sudah terdapat beberapa usaha yang bergerak dalam bidang album foto digital seperti *Photobook* Indonesia[1], *Picbit Photobook*[2] dan *SnappyPhotobook*[3] tetapi masih terdapat beberapa kekurangan, yaitu:

- *Editor* berbasis *web* sehingga membutuhkan koneksi internet untuk mengedit album foto.
- Waktu untuk memuat halaman *editor* relatif lama sehingga tidak memungkinkan untuk mengedit foto dengan koneksi internet yang lambat.

[1] http://www.photobookindonesia.com/
[2] http://picbitphotobook.com/
[3] http://www.snapyphotobook.com/site/

### *B. Modularitas*

Secara umum, pemrograman yang modular adalah teknik perancangan perangkat lunak yang menekankan kepada pemisahan fungsi program ke modul-modul yang mandiri yang dapat dintegrasikan ke aplikasi yang lebih besar.

Menurut sudut pandang ilmu teknik, modularitas secara umum memiliki tiga tujuan, yaitu:

- Untuk mengelola kompleksitas suatu pengembangan
- Untuk memungkinkan pengerjaan secara pararel
- Untuk mengakomodasi ketidakpastian yang akan terjadi di masa depan.

Modularitas mengakomodasi ketidakpastian yang akan terjadi di masa depan karena elemen tertentu pada rencang bangun modular dapat diubah selama aturan-aturan perancangan ditaati. Oleh karena itu, dalam arsitektur modular, modul baru dapat mengganti modul lama dengan mudah dan dengan biaya yang rendah [1].

Modularitas terdiri dari dua aspek: *model runtime* dan *model development*. *Model runtime* fokus kepada bagaimana mengatur sistem perangkat lunak pada saat runtime, sementara *model development* adalah bagaimana para pengembang menggunakan kerangka kerja untuk membangun perangkat lunak mereka.

Dalam pembuatan modularitas pada album foto digital ini, pemrograman berbasis *interface (interface-based programming*) diperlukan. Pemrograman berbasis *interface* adalah pola arsitektur untuk melaksanakan pemrograman modular pada tingkat komponen dalam bahasa pemrograman berorientasi objek yang tidak memiliki sistem modul, sebagai contoh *Java*. *Java* tidak memiliki sistem modul pada tingkat komponen. *Java* memiliki sistem *package*, tetapi komponen perangkat lunak *Java* biasanya terdiri dari beberapa *packages Java*. Pemrograman berbasis *interface* memberikan keuntungan.

*Interface* modul menyatakan unsur-unsur yang akan disediakan dan dibutuhkan oleh modul. Implementasinya berupa kode yang bekerja yang merespon unsur-unsur yang terdapat pada *interface*.

### *C. Java Service Loader*

Kelas *ServiceLoader* merupakan kelas dari *java*.util yang dapat membaca konfigurasi berkas yang tersimpan dalam berkas arsip *Java* (JAR) dan menemukan implementasi dari sebuah implementasi, kemudian membuat implementasi tersebut tersedia sebagai daftar objek untuk dipilih [2]. Untuk tujuan pemuatan, *service* direpresentasikan oleh satu jenis *interface* atau *abstract class* tunggal. *Concrete class* juga dapat digunakan, tetapi tidak direkomendasikan. Sebuah penyedia *service* mengandung satu atau lebih *concrete class* yang meng*extend* tipe servis ini dengan data dan kode yang spesifik kepada penyedia [3].

Kelas penyedia biasanya tidak bukan seluruh penyedia itu sendiri melainkan proksi yang berisi informasi yang cukup untuk menentukan apakan layanan mampu memenuhi permintaan tertentu bersama dengan kode yang dapat membuat penyedia yang sebenarnya pada permintaan.

### *D. Java Standard Widget Toolkit (SWT)*

*Standard Widget Toolkit* (SWT) adalah sebuah *widget toolkit* grafis yang digunakan pada platform *Java*. SWT *Java* merupakan salah satu alternatif untuk menampilkan elemen GUI selain *Java Absract Window Toolkit* (AWT) dan *Swing*.

Untuk menampilkan elemen GUI, implementasi SWT mengakses pustaka asli GUI dari sistem operasi menggunakan JNI (*Java Native Interface*). Program yang menggunakan SWT portabel, tetapi implem entasi dari *toolkit* unik pada setiap platform [4].

### *E. Java Properties*

Kelas properti pada *java* merupakan satu set yang persistent dari properti. Properti dapat disimpan dan dimuat dari sebuah *stream*. Setiap kunci dan nilai yang dikandung dalam daftar properti adalah *string* [5]. Kelas properti merupakan kelas yang aman dari *thread*. Beberapa *thread* dapat berbagi properti tunggal tanpa perlu sinkronisasi eksternal.

Kelas properti digunakan oleh banyak kelas yang lain [6]. Beberapa metode dan konstruktor pada properti yang digunakan pada tugas akhir ini adalah:

- *Properties*()
- *String getProperty*(*String key*)
- *void load(InputStream streamIN) throws IOException*
- *Object setProperty(String key, String value)*
- *void store(OutputStream streamOut, String description)*

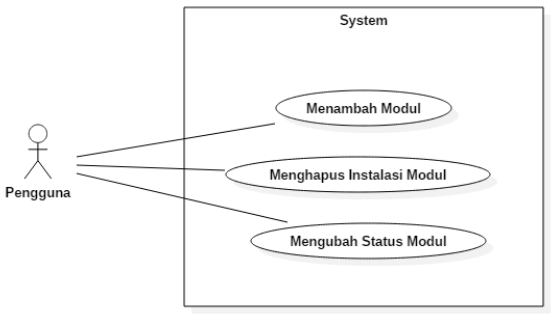
## III. ANALISIS DAN PERANCANGAN SISTEM

### *A. Analisis*

Fotokita terdiri dari dua *platform* yaitu *desktop* dan *website*. Aplikasi berbasis *desktop* berfungsi sebagai *editor* foto album dan *website* berfungsi untuk menangani proses bisnis yang terjadi antara pelanggan dan percetakan dalam proses pencetakan album foto.

Fotokita modular merupakan perangkat lunak yang mengintegrasikan modul-modul bawaan pada aplikasi Fotokita yaitu modul *frame*, modul *tool* dan modul efek. Fitur administratif perangkat lunak utama merupakan fitur bagi pengguna untuk menambah modul, menghapus instalasi modul dan mengaktifkan/mengnonaktifkan modul.

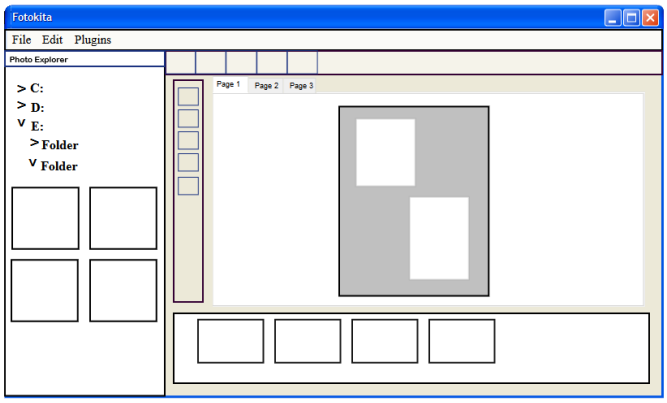
Kasus penggunaan pada aplikasi ini adalah menambah modul, menghapus instalasi modul dan mengubah status modul seperti pada Gambar 1.




Gambar 1. Diagram kasus penggunaan.

### B. *Perancangan Interface Fotokita*

Fotokita merupakan aplikasi standalone yang dapat dijalankan mandiri. Untuk membuat class diagram dan juga menentukan bagian apa saja yang bisa dijadikan *plugin* dan juga atribut yang dibutuhkan untuk mengembangkan modul, rancangan aplikasi Fotokita dipecah menjadi beberapa bagian. Secara umum, fotokita memiliki tampilan antarmuka seperti pada Gambar 2.

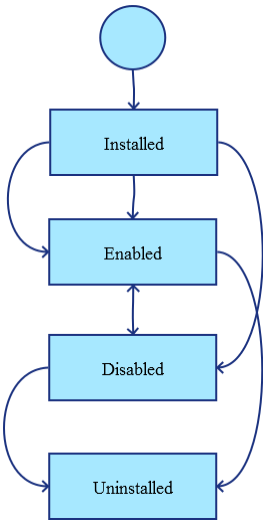



Gambar 2. Rancangan Antarmuka Album Foto Digital Fotokita

Album Foto Digital Fotokita dapat dibagi menjadi beberapa bagian yaitu *menubar*, *toolbar*, *toolbox*, *photo explorer*, *project explorer* dan obyek yang bisa diletakkan pada *page*. Setiap bagian dapat dipecaha menjadi bagian-bagian kecil lagi. *Menubar* memiliki *menu* dan *menu items*, *toolbox* terdiri dari *tools*, obyek yang bisa diletakkan pada *page* adalah *frame*, *layout* dan foto. Masing-masing bagian beserta pecahannya dibuatkan *interface*.

### C. *Perancangan Modularitas*

Daur hidup modul pada Fotokita ditunjukkan pada Gambar 3. Daur hidup modul pada Fotokita adalah *installed*, *enabled*, *disabled* dan *uninstalled*. *Installed* adalah kondisi ketika modul sudah berhasil dipasang, *enabled* adalah kondisi ketka modul terpasang dan aktif, *disabled* adalah kondisi ketika modul terpasang tetapi tidak aktif dan *uninstalled* adalah kondisi ketika modul telah terhapus instalasinya.




Gambar 3. Daur Hidup Modul

Modul yang dikembangkan wajib memiliki komponen yang mendefinisikan properti modul. Modul akan menggunakan beberapa kelas pada perangkat lunak utama. Karena itu, modul memiliki ketergantungan/dependensi dengan komponen lain dari perangkat utama. Komponen yang menjadi dependensi yaitu:

1. fotokita.*plugin*
   Modul yang dibuat wajib memiliki suatu kelas yang merealisasikan kelas *interface IModulPlugin* (jika modul *enhancement* maka merealisasikan kelas *interface IEnhancementPlugin*) dan kelas *IPlugin*.
2. fotokita.*widget.edit.enhancement*
   Efek yang dibuat wajib memiliki satu kelas yang mengimplementasikan kelas abstrak efek yang berada pada komponen ini.
3. Fotokita.*widget.tool.toolbox*
   *Tool* yang dibuat wajib memiliki satu kelas yang mengimplementasikan kelas abstrak *tool* yang berada pada komponen ini.

### D. *Perancangan Data*

Dengan menggunakan *Java Properties*, setiap berkas arsip *java* yang berhasil dipasang pada aplikasi akan memiliki satu berkas *properties* yang berisi beberapa data yang dijelaskan pada Tabel 1.

Tabel 1.
Perancangan Berkas *Properties*

| Nama *Property* | Keterangan |
|---|---|
| *pluginName* | Nama berkas arsip *java* |
| *pluginPath* | Direktori berkas arsip *java* |
| *enabled* | Modul terpasang atau tidak terpasang |
| *active* | Status modul |

## IV. IMPLEMENTASI

### A. *Lingkungan Implementasi*

Dalam merancang perangkat lunak ini digunakan beberapa perangkat pendukung yang terdiri dari perangkat keras dan perangkat lunak.

Spesifikasi perangkat keras yang digunakan dalam membangun *web* layanan transaksi *photobook* adalah sebagai berikut:

Tipe : HP *Pavilion* 7
Prosesor : *Int el ® Core* ™ i7-4510U CPU @ 2.00 GHz
Memori : 8.00 GB RAM
Sistem Operasi : *Windows* 8.1 *Professional* 64 bit

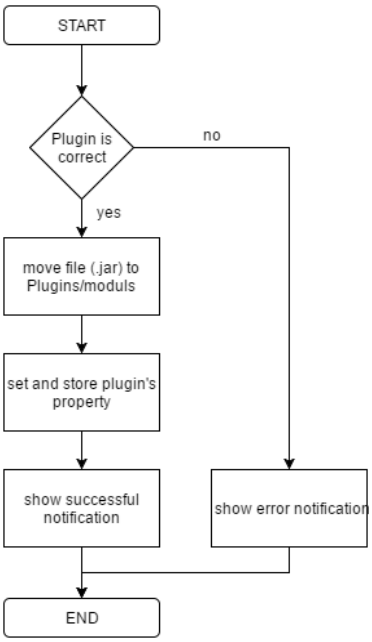
Perangkat lunak yang digunakan dalam Fotokita modular adalah sebagai berikut:

- *Eclipse Mars* digunakan sebagai IDE
- *Star*UML 2.7.0 digunakan untuk membuat digram kasus penggunaan, diagram aktivitas dan diagram kelas.

### B. *Implementasi Proses Bisnis*

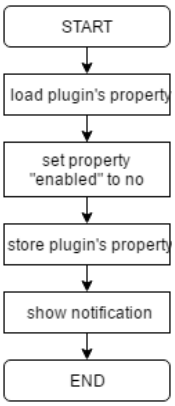
Gambar 4 menunjukkan diagram alir dari kasus penggunaan menambah modul. Ketika tombol *add* pada jendela manajemen modul ditekan, *file dialog* akan muncul dan menampilkan berkas yang mempunyai ekstensi .jar. Ketika berkas arsip *Java* dipilih dan tombol *open* ditekan, berkas arsip *Java* akan diperiksa terlebih dahulu apakah sudah memenuhi aturan modularitas dan/atau berkas tersebut belum ada, jika iya maka

berkas tersebut akan disalin ke dalam *folder Plugins*/Modul pada proyek, dan jika tidak maka pesan error akan dimunculkan. Aplikasi juga akan membuat satu berkas *properties* yang berisi informasi berkas arsip *Java* tersebut, yaitu nama *plugin*, lokasi *plugin*, status *plugin* dan aktif/tidaknya *plugin*. Berkas *properties* ini berfungsi untuk manajemen modul.



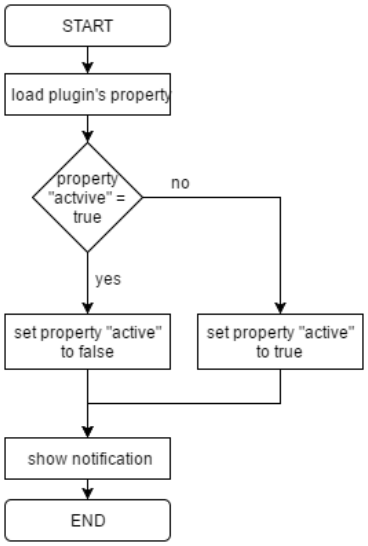

Gambar 4. Diagram Alir Kasus Penggunaan Menambah Modul

Gambar 5 menunjukkan diagram alir dari kasus penggunaan menghapus instalasi modul. Ketika salah satu baris dari dari tabel ditekan dan tombol *uninstall* pada jendela manajemen modul ditekan, sistem akan membuka *berkas properties* dari modul yang ingin dihapus instalasi, lalu akan merubah properti *enabled* menjadi *no*. Untuk melihat perubahan, aplikasi harus di*restart* ulang terlebih dahulu. Berkas arsip *Java* pada *folder Plugins*/Modul dapat dihapus oleh pengguna secara *manual*.




Gambar 5. Diagram Alir Kasus Penggunaan Menghapus Instalasi Modul

Gambar 6 menunjukkan implementasi dari kasus penggunaan mengubah status modul. Ketika salah satu baris dari tabel dipilih dan tombol *change status* ditekan, sistem memeriksa properti *active* pada *berkas properties* modul yang dipilih. Jika properti *active* bernilai *true*, nilai properti diubah menjadi *false* dan sebaliknya jika properti *active* bernilai *false*, nilai properti diubah menajdi *true*.




Gambar 6. Diagram Alir Kasus Penggunaan Mengubah Status Modul

### *C. Implementasi Modularitas*

Penjelasan cara pembuatan modul meliputi aturan apa saja yang diperlukan sebuah modul agar dapat digabungkan dengan perangkat lunak utama dan langkah-langkah pembuatan modul yang mengikuti aturan-aturan yang telah dijelaskan.

Berikut aturan-aturan yang perlu diimplementasikan pada modul:

1. Modul wajib di-*build* sebagai berkas arsip *Java* (*.jar*).
2. Modul wajib memiliki satu kelas yang mengimplementasikan *IEnhancementPlugin* atau *IToolPlugin*.
3. Modul wajib memiliki satu berkas pada folder META-INF/*Services* yang berjudul *plugins.IEnhancementPlugin*, atau *IToolPlugin* yang berisi satu baris dengan nama *package* lengkap dengan kelas berkas yang akan diimplementasi.
4. Pada *project* modul *Fotokita.jar* wajib untuk dimasukkan ke *build path*. Tetapi pada saat di-*build*, *FotokitaPlugin.jar* tidak wajib untuk ikut di arsipkan.
5. Penamaan berkas arsip *java* harus sesuai dengan nama pada fungsi *getName()*.
6. Tidak boleh ada *error* pada *project*.

Adapun langkah-langkah untuk membuat modul adalah:

1. Membuat *project* baru pada *Eclipse*
2. Klik kanan pada *project*, pilih *Build Path -> Configure Build Path*
3. Pada *tab Libraries* menekan tombol *Add External JARs* dan memilih berkas Fotokita.jar
4. Membuat kode
5. Membuat *folder* META-INF/Services dengan nama berkas sesuai dengan modul yang ingin dibuat, *plugins.IEnhancementPlugin* untuk modul efek dan *plugins.IToolPlugin* untuk modul *tool*. Isi berkas tersebut berisi satu baris berisi nama kelas lengkap dengan *package*. Sebagai contoh untuk membuat *text tool* maka isi dari *plugins.IToolPlugin* adalah *text.TextToolPlugin*.

6. Klik kanan pada *project*, pilih *Export*, lalu pilih *JAR File* sebagai *export destination*, lalu tekan tombol *Next*, dan centang *package* maupun berkas yang akan diekspor.
7. Pilih destinasi berkas, lalu klik tombol *Finish*.

Pembacaan berkas modul dilakukan pada kelas *PluginManager* dalam fungsi getEnhancement*Plugin*s, get*ToolPlugin*s dan get*FramePlugin*s yang masing-masing akan mengembalikan list yang berisi modul yang sesuai dengan aturan. Untuk modul *frame*, daftar *frame* tidak perlu dimuat karena berupa gambar (.png).

## V. PENGUJIAN DAN EVALUASI

Pengujian yang dilakukan berupa pengujian fungsionalitas. Pengujian fungsionalitas dilakukan dengan model *blackbox* untuk menguji proses penambahan modul, proses penghapusan instalasi modul dan proses pengubahan status modul.

Uji coba fungsionalitas administratif adalah uji coba yang dilakukan terhadap fungsionalitas yang dapat dilakukan oleh pengguna. Uji coba dilakukan dengan metode *black box* yang artinya fungsionalitas diperiksa apakah terpenuhi atau tidak tanpa melihat struktur internal ataupun metode yang digunakan dalam pengerjaan fungsionalitas tersebut. Skenario pengujian ditunjukkan pada Tabel 2.

Tabel 2.
Skenario Pengujian

| Kasus Pengujian | Skenario | Hasil yang Diharapkan |
|---|---|---|
| Menambah Modul | Semua aturan modul sudah terpenuhi | Modul berhasil didaftarkan |
| | Ada aturan modul yang tidak terpenuhi | Modul gagal ditambahkan |
| | Modul sudah terpasang | Modul berhasil didaftarkan |
| Menghapus Instalasi Modul | Menghapus modul yang sudah terpasang | Modul terhapus |
| Mengubah Status Modul | Mengubah status menjadi aktif | Modul aktif |
| | Mengubah status menjadi tidak aktif | Modul tidak aktif |

Berdasarkan data pada Tabel 2, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik kesimpulan bahwa fungsionalitas dari program telah bisa bekerja sesuai dengan yang diharapkan dan menunjukkan hasil yang benar.

## VI. KESIMPULAN

Dari hasil pengamatan selama proses perancangan, implementasi dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Album foto digital Fotokita dibangun dengan menggunakan pemrograman berorientasi objek sehingga dapat dipecah menjadi elemen-elemen kecil sehingga bisa dibangun secara modular.
2. Modul yang dibangun harus memenuhi beberapa aturan tertentu agar bisa berfungsi pada sistem.
3. Daur modul pada Fotokita adalah *installed*, *enabled*, *disabled* dan *uninstalled*.

## DAFTAR PUSTAKA

[1] K. Knoernschil, Java Application Architechture, Modularity Patterns with Examples Using OSGi, Indiana: Prentice Hall, 2013.

[2] "IBM Developer Works," IBM, [Online]. Available: http://www.ibm.com/developerworks/library/j-5things12/. [Accessed 31 May 2016].

[3] "Class ServiceLoader<S>," Oracle, [Online]. Available: http://docs.oracle.com/javase/6/docs/api/java/util/ServiceLoader.html. [Accessed 31 May 2016].

[4] "Wikipedia - Standard Widget Toolkit," [Online]. Available: https://en.wikipedia.org/wiki/Standard_Widget_Toolkit. [Accessed 8 June 2016].

[5] "Properties (Java Platform)," Oracle, [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/util/Properties.html. [Accessed 8 June 2016].

[6] "Java - The Properties Class," [Online]. Available: http://www.tutorialspoint.com/java/java_properties_class.htm. [Accessed 8 June 2016].