REFACTORING OBJEK MENJADI ASPEK PADA PERANGKAT LUNAK E-COMMERCE MENGGUNAKAN METODE EKSTRAKSI

Muhammad Aris Ganiardi[1], Sony Oktapriandi[2]
[1,2]Jurusan Manajemen Informatika, Politeknik Negeri Sriwijaya
e-mail : [1]marisg2010@gmail.com, [2]sony.oktapriandi@gmail.com

ABSTRAK

Perangkat lunak berorientasi objek memiliki kelemahan yaitu terdapat concern-concern yang tersebar di berbagai objek. Hal ini menyebabkan sulitnya perangkat lunak beradaptasi terhadap perubahan-perubahan kebutuhan yang berasal dari pengguna dan stakeholder. Agar perangkat lunak berorientasi objek dapat beradaptasi terhadap perubahan tersebut maka struktur programnya harus di-refactoring menjadi program aspek. Concern-concern yang tersebar di berbagai objek diidentifikasi, dipisahkan dan dimodularisasi sebagai aspek perangkat lunak. Refactoring perangkat lunak berorientasi objek menjadi perangkat lunak berorientasi aspek dilakukan dengan menggunakan metode ekstraksi. Dengan menggunakan metode ini concern-concern yang ditemukan akan diletakkan berdasarkan penggunaan dari concern-concern tersebut ketika perangkat lunak digunakan. Studi kasus yang digunakan dalam penelitian ini adalah aplikasi web e-commerce pada mahasiswa dan alumni wirausaha Inkubator Bisnis Polsri. Hasil yang dihasilkan pada penelitian ini diharapkan dapat membantu mahasiswa dan alumni wirausaha Inkubator Bisnis Polsri untuk memasarkan produk mereka dan memudahkan pembeli untuk memesan sebuah produk di Inkubator Bisnis Polsri.

Kata Kunci: *refactoring, pemograman berorientasi aspek, metode ekstraksi*

1. PENDAHULUAN

Paradigma pengembangan perangkat lunak yang paling popular saat ini adalah pengembangan perangkat lunak berorientasi objek. Perangkat lunak yang dikembangkan dengan menggunakan paradigma berorientasi objek dikonstruksi dari objek-objek yang didapat dari fase analisis siklus pengembangan perangkat lunak. Sebuah objek didapatkan berdasarkan modularisasi atribut-atribut dan operasi-operasi yang terjadi di sebuah perangkat lunak. Keuntungan penggunaan paradigm ini adalah kemudahan penggunaan ulang objek-objek pada perangkat lunak lain dan mudah dilakukan perbaikan objek pada perangkat lunak.

Kepopuleran paradigma berorientasi objek tidak membuat paradigma ini mempunyai kelemahan. Kelemahan pertama adalah pada paradigma ini, objek dimodularisasi berdasarkan sekumpulan data. Efek dari modularisasi berdasarkan sekumpulan data adalah muncul satu fungsi yang digunakan di banyak objek (kode *tangling*) dan satu objek digunakan di banyak fungsi (kode *scattering*) [2]. Fungsi dan objek ini disebut dengan *concern*. Kelemahan yang kedua paradigma berorientasi objek adalah proses pembangunan perangkat lunak lebih menekankan pada faktor fungsionalitas dibandingkan dengan faktor non-fungsionalitasnya. Padahal dalam implementasinya kadang kala faktor non-fungsionalitas seperti keamanan lebih memiliki peranan dibandingkan dengan faktor fungsionalitas. Akibat kelemahan tersebut objek-objek akan saling bergantung satu dengan yang lain. Apabila ada satu kesalahan di sebuah objek maka kesalahan tesebut akan mempengaruhi objek yang lain. Secara global kesalahan-kesalahan di atas akan mempengaruhi kualitas perangkat lunak.

Kelemahan-kelemahan pengembangan perangkat lunak berorientasi objek diperbaiki oleh sebuah paradigma baru yang disebut pengembangan perangkat berorientasi aspek. Perangkat lunak berorientasi aspek merupakan perangkat lunak yang dibangun dengan cara memisahkan penyebaran *concern* (kode *tangling* dan *scattering*) yang terdapat di berbagai objek. Struktur kode program perangkat lunak mudah dipahami dan berisikan *method- method* inti bisnis utama yang dimiliki oleh objek-objek. *Concern-concern* yang tersebar dipisahkan dan dimodularisasi menjadi aspek yang bersifat mudah digunakan ulang. Perangkat lunak yang dibangun menggunakan paradigma berorientasi aspek lebih mudah untuk dirawat untuk ditambah atau dikurangi fiturnya.

Pengembangan perangkat lunak berorientasi aspek dapat dilakukan dengan berbagai cara salah satunya adalah dengan melakukan *refactoring* perangkat lunak berorientasi objek. Fase-fase yang dilakukan adalah analisis, perancangan, pengkodean, pengujian, dan pemeliharaan perangkat lunak. Pada fase pengkodean perangkat lunak berorientasi objek, objek-objek yang sudah dibuat diubah menjadi aspek berdasarkan identifikasi dan pemisahkan *concern* yang terdapat pada objek. Permasalahan yang muncul pada pengembangan perangkat lunak berorientasi aspek ini adalah "Bagaimana cara melakukan *refactoring* program *concern* menjadi aspek pada program perangkat lunak setelah perangkat lunak selesai dibuat ? Untuk menyelesaikan permasalahan tersebut, peneliti menggunakan metode ekstraksi untuk mengubah objek-objek menjadi aspek ketika perangkat lunak selesai dibuat.

2. TINJAUAN PUSTAKA

Penelitian di bidang pengembangan perangkat lunak berorientasi aspek sudah banyak di mulai dari awal tahun 2000. Fokus dari penelitian dari penelitian ini adalah 1. Menotasikan sebuah aspek dalam bentuk rancangan 2. Pengembangan Perangkat lunak Berorientasi aspek ? 3. Bagaimana mengidentifikasi dan memisahkan *concern* menjadi aspek ?

a. Lidia Fuentes dalam jurnalnya berjudul "*Designing and Weaving Aspect-Oriented Executable UML models*" tahun 2007 menotasikan aspek-aspek dalam sebuah perangkat lunak menggunakan notasi UML [9].
b. Tudor B. Ionescu et al dalam jurnalnya berjudul "*An Aspect-Oriented Approach for the Development of Complex Simulation Software*" tahun 2010 mengembangkan perangkat lunak berorientasi aspek yang mensimulasikan pencegahan bencana radioaktif yang disebut dengan sistem ABR [12].
c. Elisa Baniassad and Siobh´an Clarke dalam jurnalnya berjudul "*Theme: An Approach for Aspect-Oriented Analysis and Design*" tahun 2005 membahas identifikasi dan pemisahan *concern* pada fase analisis dan perancangan pada pengembangan perangkat lunak menggunakan *template theme* [2].

2.1. Konsep Pembangunan Perangkat Lunak Berorientasi Aspek

Sistem yang besar memiliki hubungan yang kompleks antara kebutuhan-kebutuhan dan komponen-komponen program. Sebuah kebutuhan tunggal dapat diimplementasikan dalam sejumlah komponen dan setiap komponen terdapat di elemen-elemen beberapa kebutuhan. Dalam keadaan seperti ini untuk mengimplementasikan sebuah perubahan pada kebutuhan-kebutuhan harus melibatkan perubahan di banyak komponen dan pemahaman tentang komponen itu sendiri. Penggunaan ulang komponen-komponen menjadi sangat sulit karena komponen-komponen tidak dapat diimplementasikan pada sebuah sistem abstraksi tunggal tetapi juga termasuk fragmen-fragmen kode program yang diimplementasikan di kebutuhan-kebutuhan lain.

Pembangunan perangkat lunak berorientasi aspek (AOSD, *Aspect Oriented Software Development*) adalah sebuah pendekatan yang digunakan dalam pembangunan perangkat lunak yang bertujuan untuk mengatasi permasalahan diatas dan membuat program menjadi mudah untuk dirawat dan digunakan ulang. AOSD dibangun berdasarkan tipe abstraksi baru yang disebut dengan sebuah aspek. Aspek terbentuk dari bentuk abstraksi lain seperti objek-objek dan *method-method*. Aspek mengenkapsulasi fungsionalitas yang tersebar dan berdampingan dengan fungsionalitas lain yang termasuk di dalam sistem. Sebuah program berorientasi aspek dibuat secara otomatis dengan memkombinasikan (*weaving*) objek-objek, *method-method*, dan aspek-aspek lain, berdasarkan spesifikasi yang dibutuhkan.

Keunggulan pembangunan perangkat lunak berorientasi aspek adalah aspek mendukung pemisahan *concern-concern*. Pemisahan *concern-concern* ke dalam bentuk elemen-elemen independen merupakan teknik pemrograman yang sangat baik dalam pembangunan perangkat lunak. Dengan merepresentasikan penyebaran *concern-concern* sebagai aspek, *concern* dapat dipahami, mudah digunakan ulang dan digunakan secara tunggal. Sebagai contoh otentikasi pengguna direpresentasikan sebagai sebuah aspek yang dapat diminta sebagai login dengan masukkan sebuah nama dan kata kunci. Aspek ini dapat secara otomatis dikombinasikan ke dalam program dimana otentikasi dibutuhkan.

2.2. Refactoring

Refactoring adalah proses memperbaiki struktur internal sebuah sistem perangkat lunak tanpa mengubah sedikitpun fungsionalitas dari sistem. Artinya, dalam proses *refactoring* dilakukan modifikasi program untuk memperbaiki struktur, mengurangi kompleksitas, atau untuk membuatnya lebih mudah dimengerti. *Refactoring* dilakukan pada program perangkat disebabkan karena adanya *Bad Smell*. Bentuk *Bad Smell* yang sering terjadi di pada program seperti : duplikasi program yang sangat banyak, method yang ditulis panjang, dan ukuran sebuah *class* yang besar [4].

Perangkat lunak berorientasi objek masih memiliki kelemahan yaitu terdapatnya duplikasi dan penyebaran concern di berbagai class. [1] memperkenalkan enam metode ektraksi refactoring perangkat lunak berorientasi aspek yang berasal dari perangkat lunak berorientasi objek. Hasil dari perubahan perangkat lunak tesebut diharapkan struktur program lebih baik karena tidak ada Bed Smell lagi.

Metode Ektraksi *refactoring* perangkat lunak berorientasi objek menjadi perangkat lunak berorientasi aspek :

a. Ekstraksi awal/akhir dari *method/handler*
b. Ekstraksi sebelum atau sesudah pemanggilan
c. Ekstraksi bersyarat
d. Ekstraksi sebelum return
e. Ekstraksi *wrapper*
f. Ekstraksi penanganan *exception*

3. METODE PENELITIAN

Tahapan-tahapan yang dilakukan pada penelitian ini ada empat tahapan yaitu :

3.1. Metode Pemngumpulan Data dan Kebutuhan

a. Studi Literatur

Metode ini dilakukan dengan mengkaji beberapa literatur yang berkaitan dengan penelitian pengembangan perangkat lunak berorientasi aspek, pemrograman berorientasi aspek, dan konsep *refactoring* perangkat lunak. Literatur-literatur tersebut diperoleh dari :

1) Buku-buku dan jurnal-jurnal penelitian baik oleh penulis dari dalam negeri maupun dari luar negeri.
2) Data Perguruan Tinggi Negeri dan Swasta yang ada di kota Palembang.
3) Informasi dari media masa, seperti surat kabar dan internet.

b. Survei

Pelaksanaan metode ini dengan melakukan observasi, kuesioner dan wawancara langsung pada para mahasiswa dan alumni wirausaha yang tergabung dalam Inkubator Bisnis Politeknik Negeri Sriwijaya Palembang. Peneliti mendatangi langsung ke tempat usaha para mahasiswa dan alumni wirausaha tersebut yang tersebar di berbagai wilayah kota palembang, melakukan pengamatan pada proses bisnis mereka dan wawancara berkaitan dengan kebutuhan mereka terhadap perangkat lunak *e-commerce* dalam transaksi *on-line* dengan para pembeli.

3.2. Pembangunan Perangkat Lunak e-commerce Berorientasi Objek

Metode pengembangan perangkat lunak *e-commerce* yang akan digunakan pada penelitian ini adalah metode spiral. Secara umum metode spiral memiliki lima fase yaitu : (1) Analisis, (2) perancangan, (3) Pengkodean, (4) Pengujian, (5) Pemeriharaan. Metode spiral merupakan pengembangan dari metode spiral. Metode ini dipilih dalam pengembangan perangkat lunak karena metode ini mengizinkan pengembang kembali ke fase sebelumnya ketika ditemukan sebuah kesalahan. Sebagai contoh jika saat ini pengembang berada di fase pengkodean dan pengembang menemukan kesalahan terjadi di fase analisis maka pengembang dapat kembali ke fase analisis. Pada fase analisis tersebut pengembang dapat memperbaiki kesalahan tersebut sampai benar. Perangkat lunak ini nantinya akan dibuat dengan menggunakan bahasa pemrograman Java menggunakan IDE (*Integrated Development Environment*) Eclipse dan DBMS (*Database Management System*) MySQL di atas Sistem Operasi Windows 7 (*seven*). Pemilihan bahasa pemrograman Java, Eclipse dan MySQL dilakukan karena bersifat *freeware* kecuali untuk sistem operasi Windows 7 (*seven*).

Berikut ini penjelasan tahapan pengembangan perangkat lunak menggunakan metode spiral

a. Analisis
   1) *Pengumpulan kebutuhan*: data yang dikumpulkan adalah kebutuhan primer dan sekunder. Kebutuhan primer berasal dari data yang diambil langsung dari para *user* dan *stakeholder* pengguna perangkat lunak *e-commerce* seperti pembeli, penjual, dan pakar di bidang ilmu pengetahuan dan kebutuhan sekunder berasal dari data buku dan internet. Tahap ini sering disebut juga dengan tahapan akuisisi kebutuhan perangkat lunak. Teknik yang digunakan untuk mengumpulkan kebutuhan adalah wawancara, observasi, kuesioner, dan dokumentasi.
   2) *Analisis kebutuhan*: Kebutuhan yang telah terkumpul diinterpretasi maksud dari kebutuhan yang didapat. Kebutuhan-kebutuhan yang berasal dari *user* dan *stakeholder* dianalisis untuk menentukan prioritas implementasinya. Interpretasi sebuah kebutuhan dapat berupa data, batasan, aturan, skenario, dan fungsionalitas perangkat lunak.
   3) *Analisis basis data*: kasus yang telah terkumpul kemudian dibentuk sesuai dengan format basis data supaya bisa dimasukan ke dalam basis data seperti pembuatan tabel kasus, pembuatan kunci, relasi antar tabel dan pembentukan *query*.
   4) *Analisis model aplikasi*: model aplikasi yang akan digunakan berbasis *web* yang nantinya dapat didistribusikan secara melalui internet.
   5) *Analisis perangkat lunak*: perangkat lunak yang digunakan adalah bahasa pemrograman java, IDE Eclipse, DBMS MySQL serta Sistem Operasi Windows 7 (*seven*).
   6) *Analisis perangkat keras*: Analisa kebutuhan perangkat keras terhadap sistem seperti kecepatan *processor*, kapasitas memori utama dan memori sekunder.
b. Perancangan
   1) *Rancangan basis data*: Rancangan basis data merupakan lanjutan dari analisa basis data. Perancangan basis data dilakukan dengan menggunakan DBMS MySQL serta melakukan pembuatan *query-query* yang nanti akan digunakan oleh sistem.
   2) *Rancangan antarmuka*: Merancang tampilan masukan dan keluaran yang berbasis GUI (*Graphical User Interface*) menggunakan IDE Eclipse.
   3) *Rancangan Objek* : Merancang modul-modul program dalam bentuk objek yang nantinya akan digunakan pada saat pengkodean sistem. Rancangan modul dapat berbentuk algoritma, notasi UML dan *pseudo-code*.
c. Pengkodean

1) *Pembuatan kode modul basis data*: kode modul basis data dibuat terpisah dengan kode sistem sehingga lebih bersifat *reusable*. Kode modul basis data berisi operasi basis data seperti membuat koneksi ke basis data, *insert*, *update*, *delete* dan *query*.
2) *Pembuatan kode komponen program*: kode program dibuat dan dimodularisasi menjadi objek berdasarkan spesifikasi yang telah ditetapkan dari hasil analisis.

d. Pengujian
1) *Pengujian basis data*: pengujian koneksi basis data dan akurasi *query* basis data.
2) *Pengujian sistem*: pengujian secara keseluruhan dari sistem baik dari masukan, proses dan keluaran sistem.
3) *Pengujian program*: pengujian ini dilakukan untuk mengetahui kualitas objek perangkat lunak.

e. Pemeliharaan

Dilakukan dengan dua cara yaitu ketika proses pengembangan berlangsung melakukan *backup* kode-kode program yang dibuat jika melakukan revisi program dan setelah proses pengembangan dengan melihat kinerja sistem apakah masih menghasilkan akurasi yang baik selama sistem berjalan.

3.3. Refactoring Perangkat Lunak Objek Menjadi Perangkat Lunak Berorientasi Aspek

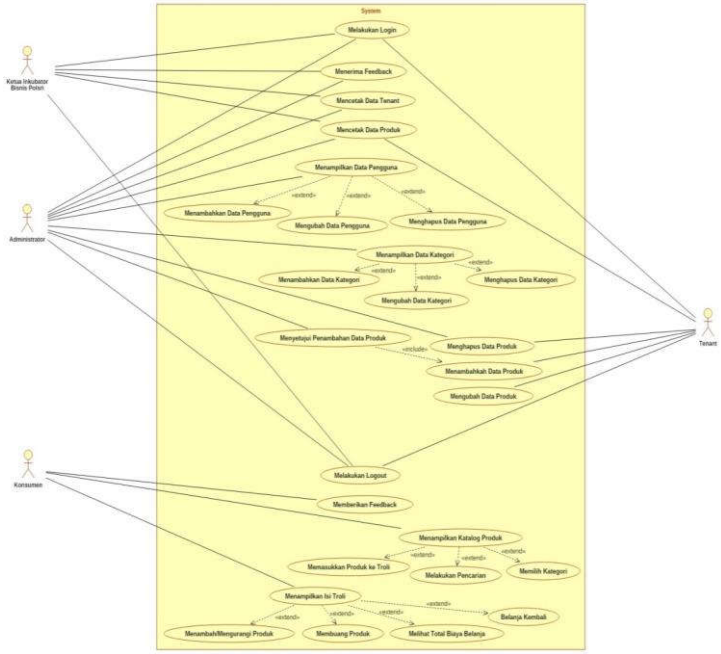
Tahapan *refactoring* perangkat lunak berorientasi objek menjadi perangkat lunak berorientasi aspek merupakan tahpan utama pada penelitian ini. Pada tahapan ini perangkat lunak berorientasi objek diubah struktur program menjadi perangkat lunak berorientasi aspek. Proses yang dilakukan pengubahan ini dengan cara : (1) Identifikasi *cross cutting concern* yang tersebar di berbagai objek perangkat lunak. *Cross cutting concern* yang ditemukan merupakan kandidat aspek perangkat lunak. (2) Pisahkan *cross cutting concern* yang telah ditemukan. (3) Membuat aspek berdasarkan *cross cutting concern* yang telah ditemukan dan dipisah. (4) Melakukan refactoring struktur program perangkat lunak berdasarkan aspek yang telah dibuat menggunakan metode ektraksi. Ada enam bentuk metode ektraksi yang digunakan untuk me-*refactoring* struktur program yaitu : Ekstraksi awal/akhir dari *method/handler*, Ekstraksi sebelum atau sesudah pemanggilan, Ekstraksi bersyarat, Ekstraksi sebelum return, Ekstraksi *wrapper*, Ekstraksi penanganan *exception*.

3.4. Pengujian Perangkat Lunak Berorientasi Aspek

Perangkat lunak berorientasi aspek yang telah selesai di-*refactoring* diuji fungsionalitasnya. Tujuan dari pengujian ini untuk mengetahui apakah ada perubahan fungsionalitas setelah selesai *refactoring*. Pengujian dilakukan dengan cara membuat skenario berbagai kondisi data pada setiap form pada perangkat lunak.Kondisi yang dibuat dalam keadaan normal dan tidak normal penggunaan perangkat lunak.

4. HASIL DAN PEMBAHASAN

4.1. Diagram Use Case



Gambar 1. Diagram Use Case *E-Commerce*

Diagram *use case* menunjukkan interaksi antara aktor dengan dengan aplikasi web e-*commerce*. Ada empat aktor yang berinteraksi dengan aplikasi web e-*commerce* yaitu Administrator yang bertanggung jawab

dengan keseluruhan sistem, kepala inkubator bisnis, tenant dan konsumen sebagai pengguna utama aplikasi web e-*commerce* dan pengunjung yang mempunyai hak hanya melihat-lihat halaman utama aplikasi web e-*commerce*. Bentuk interaksi antara aktor dan aplikasi web e-*commerce* merupakan implementasi dari kebutuhan-kebutuhan fungsionalitas dari semua aktor.

4.2. Aspek Perangkat Lunak E-Commerce

Berdasarkan penggunaan *use cases* dalam identifikasi aspek pada analisis perangkat lunak *e-commerce*, ada sembilan aspek yang berhasil diidentifikasi dan dimodularisasi. Dari kesembilan aspek tersebut ada dua aspek yang tidak diintegrasikan ke perangkat lunak *e-commerce*, sedangkan tujuh aspek lainnya diintegrasikan. Aspek-aspek yang dimodularisasi dibagi menjadi empat kategori yaitu aspek *pooling* yang berfungsi untuk mengatur penggunaan sumber daya basisdata, aspek *validator* yang berfungsi untuk mevalidasi data, aspek *security* berfungsi untuk keamanan perangkat lunak *e-commerce*, dan aspek *testing* berfungsi untuk menguji secara internal kode program perangkat lunak *e-commerce*. Daftar aspek perangkat lunak *e-commerce* secara lengkap ditunjukkan pada tabel 1.

Tabel 1. Daftar Aspek perangkat lunak *e-commerce*

No.	Kategori	Aspek E-Commerce	Keterangan
1	Aspek *Pooling*	Database	Diimplementasikan
		ConnectionnChecking	Diimplementasikan
2	Aspek *Validator*	ValidateString	Diimplementasikan
		ValidateNumeric	Diimplementasikan
		ValidateNull	Diimplementasikan
		Errors	Tidak diimplementasikan
		Status	Tidak diimplementasikan
3	Aspek *Security*	GeneratePassword	Diimplementasikan
4	Aspek *Testing*	Tracer	Diimplementasikan

Penjelasan aspek-aspek perangkat lunak *e-commerce* adalah sebagai berikut :

a. Aspek Database adalah aspek perangkat lunak *e-commerce* yang berfungsi untuk koneksi antara perangkat lunak *e-commerce* dengan basisdata commerce. Aspek ini digunakan untuk menyimpan data perangkat lunak *e-commerce* dan menampilkan data tersebut di halaman *web* perangkat lunak *e-commerce*.
b. Aspek ConnectionChecking adalah aspek perangkat lunak *e-commerce* yang berfungsi sebagai pendeteksi hubungan antara perangkat lunak *e-commerce* dengan basisdata commerce.
c. Aspek ValidateString adalah aspek yang berfungsi untuk mevalidasi *input*-an dalam bentuk huruf yang dimasukkan oleh pengguna. perangkat lunak *e-commerce* hanya mengijinkan data yang dimasukkan dalam bentuk huruf pada *field-field* tertentu pada sebuah *form*. Jika pengguna memasukan data *input*-an dalam bentuk angka maka perangkat lunak *e-commerce* tidak memproses dan menyimpan data *input*-an tersebut ke dalam basisdata.
d. Aspek ValidateNumeric adalah aspek yang berfungsi untuk mevalidasi *input*-an dalam bentuk angka yang dimasukkan oleh pengguna. perangkat lunak *e-commerce* hanya mengijinkan data yang dimasukkan dalam bentuk angka pada *field-field* tertentu pada sebuah *form*. Jika pengguna memasukan *input*-an dalam bentuk huruf maka perangkat lunak *e-commerce* tidak memproses dan menyimpan *input*-an tersebut ke dalam basisdata.
e. Aspek ValidateNull adalah aspek yang berfungsi untuk mevalidasi *input*-an dalam bentuk "**kosong**" yang dimasukkan oleh pengguna. Pada *field-field* form tertentu, perangkat lunak *e-commerce* mewajibkan pengguna memasukkan data. Data *field-field* yang harus dimasukkan adalah data-data berbentuk kunci primer seperti kode tenant, kode barang atau data karakteristik unik entitas perangkat lunak *e-commerce*. Jika pengguna memasukan *input*-an dalam bentuk kosong maka perangkat lunak *e-commerce* akan menolak memproses dan menyimpan *input*-an tersebut ke dalam basisdata.
f. Aspek Errors adalah aspek yang berfungsi untuk menampilkan pesan kesalahan jika adalah kesalahan *input*-an yang dimasukkan oleh pengguna. Aspek ini tidak diimplementasikan dan diintegrasikan ke perangkat lunak *e-commerce* karena keterbatasan teknik implementasi pemrograman berorientasi aspek. Pada aspek Errors harus mendeklarasikan objek *errors* yang bertipe data Hashtable sebagai parameternya. Ketika diintegrasikan, aspek Errors tidak mendeteksi *join point* yang berasal dari *class*. Secara teknis harus ada kesesuaian antara parameter *join point* aspek dan *method class*.
g. Aspek Status adalah aspek yang berfungsi untuk menampilkan pesan aksi-aksi yang dilakukan oleh pengguna. Contoh pesan seperti : "pesan data sudah tersimpan di basisdata", "pesan data sudah diubah", dan "pesan data sudah dihapus di basisdata". Aspek ini tidak diimplementasikan dan diintegrasikan di perangkat lunak *e-commerce* karena bentuk *method* status yang akan diubah menjadi aspek Status sudah dalam bentuk primitif. Jika diimplementasikan maka bentuk aspek dan *method* di *class* sama.

h. Aspek GeneratePassword adalah aspek yang berfungsi untuk mengenkripsi *password* pengguna perangkat lunak *e-commerce*. Untuk mencegah pengguna lain yang tidak memiliki hak akses mengetahui *password* pengguna maka setiap *password* pengguna yang masuk ke perangkat lunak *e-commerce* akan dienkripsi.
i. Aspek Tracer adalah aspek yang berfungsi untuk menelusuri aliran kerja *method-method* program perangkat lunak *e-commerce* saat sebuah *class* dieksekusi. Aspek ini digunakan untuk menguji struktur internal perangkat lunak *e-commerce*. Hasil pengujian dengan menggunakan aspek Tracer akan didapatkan urutan *method-method* yang dieksekusi saat sebuah fungsionalitas mengeksekusi sebuah *class*.

4.3. Refactoring Perangkat Lunak E-Commerce

Perangkat lunak *e-commerce* berorientasi objek yang selesai dibuat di-*refactoring* menjadi perangkat lunak berorientasi aspek dengan menggunakan metode ekstraksi. *Refactoring* dilakukan dengan cara memisahkan method yang sama pada setiap objek menjadi aspek yang telah diidentifikasi pada fase sebelumnya. Metode ekstraksi akan menentukan posisi aspek ketika objek tersebut dieksekusi.

```
package com.aspects;

import java.sql.*;

public aspect ConnectionChecking
{
    public static final String cvs="$Id: ConnectionChecking.java,v 1.2 2002/02/07 05:57:18 ikk Exp

    Connection around() throws SQLException: call(static Connection DriverManager.getConnection(..
    {
        Connection conn;
        do
        {
            conn = proceed();
        }
        while(bad(conn));

        return conn;
    }

    private boolean bad(Connection conn)
    {
        try
        {
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT 2+2");
```

Gambar 2. Aspect ConnetionChecking hasil metode ekstraksi

4.4. Pengujian Perangkat Lunak

Secara umum pengujian perangkat lunak *e-commerce* sulit dilakukan karena ada beberapa karakteriktik aspek dan objek yang unik. Sebuah aspek tidak dapat diuji secara terpisah karena hanya berisi *concern-concern* yang tersebar dan bergantung pada objek asalnya. Untuk menguji sebuah aspek maka harus melibatkan semua objek yang menggunakan aspek. Objek sulit berkomunikasi dengan aspek karena objek tidak mengenali aspek, sedangkan aspek mengenali objek. Pengujian perangkat lunak berorientasi aspek sering dilakukan dengan cara pemeriksaan kode progam.

Pengujian perangkat lunak *e-commerce* berorientasi aspek akan dilakukan dengan tiga bentuk pengujian yaitu pengujian eksternal yang dilakukan pada fungsionalitas perangkat lunak *e-commerce*, pengujian internal yang difokuskan pada proses aliran kerja *method-method class-class* perangkat lunak *e-commerce* ketika dieksekusi, dan penambahan fungsionalitas perangkat lunak *e-commerce*. Hasil pengujian perangkat lunak *e-commerce* berorientasi aspek akan dibandingkan dengan perangkat lunak *e-commerce* berorientasi objek.

Sebagai contoh pengujian perangkat lunak *e-commerce* berorientasi aspek adalah pengujian dilakukan pada fungsionalitas mengelola data tenant dan penjualan. Fungsionalitas ini dipilih karena melibatkan banyak aspek dan memiliki jumlah variasi data yang lebih banyak dibandingkan dengan fungsionalitas lain. Prinsip kerja pengujian fungsionalitas pengelolaan data tenant dan transaksi penjualan dapat dilakukan fungsionalitas-fungsionalitas yang lain.

5. KESIMPULAN

Kesimpulan yang didapatkan selama melakukan penelitian ini adalah

a. Perangkat lunak *e-commerce* ini memberikan banyak keuntungan bagi segala pihak yang terlibat di dalam proses bisnis Inkubator Bisnis Politeknik Negeri Sriwijaya. Salah satunya adalah semakin mudahnya para tenant dalam menjangkau pangsa pasar yang secara otomatis akan meningkatkan omzet penjualan.
b. Perangkat lunak *e-commerce* berorientasi aspek yang dikembangkan memiliki karakteristik independen yang sangat tinggi. Tidak ada lagi *concern-concern* yang sama tersebar diberbagai *class-class* perangkat lunak. Karakteristik ini memudahkan adaptasi perangkat lunak *e-commerce* terhadap perubahan-perubahan yang berasal dari pengguna.
c. *Use cases* digunakan untuk mengidentifikasi penyebaran *concern-concern* perangkat lunak berdasarkan spesifikasi perangkat lunak. Spesifikasi perangkat lunak merupakan hasil konversi kebutuhan-kebutuhan pengguna. Keunggulan menggunakan u*se case*, karena use cases menggambarkan langsung interaksi antara pengguna dengan sistem.
d. Metode ekstraksi memudahkan proses *refactoring* objek menjadi aspek perangkat lunak.

DAFTAR PUSTAKA

[1] Ceccato, Mariano, (2006), *Migrating Object Oriented Code to Aspect Oriented Programming*, Ph.D. thesis. University of Trento.
[2] Clarke .2004. Theme: An Approach for Aspect-Oriented Analysis and Design. ICSE 2004. Proceedings 26th International Conference on . IEEE.
[3] Filman. 2004. *Aspect-Oriented Software Development.* Addison Wesley. USA.
[4] Fowler, M.; Beck, K.; Brant, J.; Opdyke, W. F.; Roberts, D. 2002. Refactoring*: Improving the Design of Existing Code*. Addison-Wesley.
[5] Hidehiko Masuhara1 and Gregor Kiczales. 2003. *Modeling Crosscutting in Aspect-Oriented Mechanisms*. Proceeding ECOOP. Pp 2-28.
[6] Jacobson dan Pan Wei Ng. 2004. *Aspect-Oriented Software Development with Use Cases*. Addison Wesley. USA.
[7] Jing Zhang. 2007. *Aspect Composition in The Motorola Aspect-Oriented Modeling Weaver*. Journal Of Object Technology. Vol. 6. No. 7. Special Issue: Aspect-Oriented Modeling.
[8] Joseph D. Gradecki. 2003. *Mastering AspectJ Aspect-Oriented Programming in Java.* Wiley Publishing .Inc. Canada.
[9] Lidia Fuentes. 2007. *Designing And Weaving Aspect-Oriented Executable UML Models*. Journal Of Object Technology. Vol. 6. No. 7. Special Issue: Aspect-Oriented Modeling.
[10] Ramnivas Laddad. 2006. *AspectJ in Action Practical Aspect-Oriented Programming*. Manning Publication. USA.
[11] Russell Miles. 2005. *AspectJ Cookbook*. O'Reilly. USA.
[12] Tudor B. Ionescu et al. 2010."*An Aspect-Oriented Approach for the Development of Complex Simulation Software*". Journal Of Object Technology. Vol. 9. No. 1.