ANALISIS DAN IMPLEMENTASI STATE PATTERN

Fahrul Muttaqin[1], Kusuma Ayu Laksitowening[2], Toto Suharto[3]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

Design pattern muncul akibat adanya permasalahan yang sama yang sering muncul pada desain pembuatan perangkat lunak. Pada perkembangannya sudah banyak design pattern yang sudah ditemukan oleh pala programmer. Saat ini, design pattern dikelompokkan ke dalam tiga tujuan berbeda, yaitu creational, structural, dan behavioral.

State pattern merupakan salah satu design pattern yang tergolong ke dalam behavioral pattern. State pattern muncul akibat adanya kondisi-kondisi (states) yang muncul pada saat pembuatan perangkat lunak. Pattern ini mengijinkan state transition logic untuk disatukan dengan sebuah state object daripada berada dalam kondisional atau switch statement.

Pada tugas akhir kali ini, dibuat sebuah perangkat lunak yang mengimplementasikan state pattern untuk menyelesaikan sebuah kasus yang memiliki beberapa kondisi. Untuk mengevaluasi state pattern, dilakukan pengujian serta perhitungan terhadap object-oriented metrics. Sedangkan untuk mengetahui kelebihan dan kekurangan yang ada pada state pattern, akan dibandingkan hasil perhitungan object-oriented metrics perangkat lunak yang menerapkan state pattern dengan perangkat lunak yang tidak menerapkan state pattern untuk sebuah kasus yang serupa.

Berdasarkan hasil analisis dan pengujian, jika dilihat dari perhitungan object-oriented metrics, perangkat lunak dengan state pattern memiliki kompleksitas yang lebih tinggi dan membutuhkan usaha yang lebih besar ketika dilakukan maintenance dibandingkan perangkat lunak tanpa state pattern. Namun, jika dilihat dari penanggulangan state yang ada, perangkat lunak dengan state pattern memiliki cara penanggulangan state yang lebih optimal dibandingkan perangkat lunak tanpa state pattern.

Kata Kunci : design pattern, state pattern, dan object-oriented metrics.

Abstract

Design pattern was invented because there are similar problems that often appeared in the software engineering. There are a lot of design patterns was invented by programmers in its development. Today, design pattern is clustered by three different purposes, there are creational, structural, and behavioral.

State pattern is one of behavioral pattern. State pattern was invented because programmers found states when they made software. This Pattern allows state transition logic to be incorporated into a state object rather than in a monolithic if or switch statement.

This final assignment will implement state pattern to finish a case that has several states. For evaluating state pattern, this final assignment will test and calculate object-oriented metrics from software of state pattern. Whereas to know the strength and weaknesses of state pattern, the result of object-oriented metrics state pattern's software will compare with non state pattern's software in same case study.

Based on object-oriented metrics, state pattern's software has higher complexity and higher effort in maintenance rather than non state pattern's software. However, state pattern's software has best way to keep states in software rather than non state pattern's software.

Keywords : design pattern, state pattern, and object-oriented metrics.

1. PENDAHULUAN

1.1 Latar belakang masalah

Pemrograman berorientasi objek (PBO) adalah salah satu cara membangun aplikasi, dengan didasarkan pada objek-objek dunia nyata, yang mengandung data dan fungsi/prosedur yang bekerja atas objek tersebut. Para *programmer* cenderung menyelesaikan masalah yang ada, dengan membuat desain perangkat lunak yang ada[13].

Dalam perkembangannya, banyak masalah-masalah yang kerap terjadi dalam pembangunan aplikasi berorientasi objek. Kebanyakan masalah tersebut memiliki inti permasalahan yang sama. Solusi umum dari pemasalahan yang kerap muncul tersebut telah diselesaikan oleh *developer* sebelumnya, yang disebut *design pattern*.

Salah satu permasalahan yang kerap muncul adalah terdapatnya beberapa kondisi (*state*) pada perangkat lunak yang akan kita buat. Perpindahan tiap *state* akibat adanya suatu *event* harus benar-benar dapat ditanggulangi dengan baik oleh sebuah desain perangkat lunak. Permasalahan seperti itu dapat diselesaikan dengan *state pattern*, yang akan menjamin perpindahan setiap *state* yang ada sesuai dengan *state diagram* yang ada.

Dalam sebuah jasa travel, memerlukan adanya system pencatatan penumpang, untuk memudahkan proses bisnisnya. Untuk membuat aplikasi kasus tersebut, mempunyai permasalahan yang sama yang dapat diselesaikan menggunakan state pattern, karena aplikasi tersebut memiliki beberapa kondisi (status) seperti kosong, terisi dan penuh.

Object-oriented metrics (OO metrics) dapat digunakan untuk menganalisis perangkat lunak yang menerapkan konsep pemrograman berorientasi objek. Dengan parameter yang ada, *OO metrics* dapat menilai *reusability*, *complexity*, *coupling* dan *cohesion* sebuah perangkat lunak.

1.2 Perumusan masalah

Permasalahan yang akan dibahas pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana menyelesaikan sebuah kasus dengan menggunakan state pattern?
2. Bagaimana menganalisis sebuah perangkat lunak yang menerapkan state pattern?
3. Apakah kelebihan dan kekurangan perangkat lunak yang menerapkan state pattern?

Batasan masalah pada tugas akhir ini adalah sebagai berikut:

1. Requirement dari perangkat lunak ini adalah meliputi :
   - Lihat Penumpang
   - Tambah Penumpang
   - Kurangi Penumpang
   - Hapus Semua Penumpang

2. *Object-oriented metrics* yang digunakan untuk mengukur perangkat lunak adalah CBO, LCOM, WMC, dan DIT.
3. Pada perangkat lunak ini akan digunakan basisdata dengan menggunakan MySQL sebagai databasenya. Tetapi basisdata tersebut bukan merupakan fokus dari tugas akhir ini.
4. Analisis dan desain perangkat lunak dimodelkan dengan menggunakan Eclipse SDK 3.2.0.
5. Menggunakan bahasa pemrograman Java dan NetBeans 6.7 sebagai aplikasi GUI nya serta Microsoft Windows XP sebagai system operasinya.

1.3 Tujuan

Tujuan yang ingin dicapai dalam pembuatan Tugas Akhir ini antara lain :

a) Terciptanya sebuah perangkat lunak yang menerapkan state pattern.
b) Terperolehnya hasil analisis, berupa nilai-nilai pada setiap OO metrics yg digunakan untuk menganalisis state pattern.
c) Terperolehnya kelebihan dan kekurangan state pattern berdasarkan OO metrics yang dibandingkan dengan perangkat lunak yang tidak menerapkan state pattern.

1.4 Metodologi penyelesaian masalah

1. Studi Literatur :
   Hal ini meliputi pencarian referensi yang berkaitan dengan *state pattern*, implementasi serta karateristik dari *state pattern, object-oriented metrics*, dan penerapan *state pattern* pada kode perangkat lunak, serta cara perhitungan *object-oriented metrics* pada sebuah perangkat lunak.
2. Analisis dan Desain :
   Menganalisa dan merancang desain untuk membangun perangkat lunak baik yang menggunakan *state pattern* maupun yang tidak menerapkan *state pattern*.
3. Implementasi sistem:
   Membangun prangkat lunak yang menerapkan state pattern dan yang tidak menerapkan state pattern.
4. Pengujian dan pengukuran:
   Pengujian hasil implementasi perangkat lunak secara keseluruhan yaitu pengujian seluruh fungsionalitas perangkat lunak yang dibuat serta pengukuran *object-oriented metrics* terhadap perangkat lunak yang dibangun.
5. Analisis hasil pengukuran:
   Menganalisis hasil pengukuran *object-oriented metrics*

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan rangkaian analisis, desain, implementasi, pengujian dan pengukuran yang telah dilakukan, maka dapat disimpulkan bahwa:

1. Sebuah kasus dapat menerapkan *state pattern*, jika memiliki beberapa kondisi (*state*), dimana setiap *state* yang ada dapat berpindah ke *state* yang lain jika ada suatu *event* yang terjadi.
2. *State pattern* dapat dianalisis dengan *object-oriented metrics* namun, untuk mengetahui kelebihan dan kekurangannya, hasil perhitungan *object-oriented metrics* harus dibandingkan dengan perangkat lunak yang tidak menerapkan *state pattern.*
3. Bertambahnya jumlah kelas pada desain perangkat lunak yang menerapkan *state pattern* membuat perangkat lunak ini lebih kompleks dan mempunyai nilai *coupling* yang tinggi dibandingkan perangkat lunak tanpa *state pattern.*
4. Jika dilihat dari segi kohesi, kelas-kelas yang muncul akibat penerapan konsep *state pattern* memiliki nilai LCOM yang rendah, yang berarti memiliki kohesi yang tinggi, sehingga *methods* dari kelas-kelas tersebut sudah tepat, tidak perlu dipecah ke dalam dua kelas atau lebih.
5. Tingginya nilai CBO perangkat lunak yang menerapkan *state pattern* selain berdampak besarnya usaha dalam *maintenance* perangkat lunak tersebut, mengindikasikan bahwa perangkat lunak yang menerapkan *state pattern* memiliki nilai *reusability* yang lebih tinggi terhadap penggunaan *methods* antar kelas.
6. *State pattern* mengharuskan *state trantition logic* untuk disatukan dengan sebuah *state object* daripada berada dalam kondisional atau *switch statement.*
7. Dengan menerapkan *state pattern*, kita diharuskan membuat semua kemuingkinan perpindahan *state* yang terjadi terhadap *state transition* yang terjadi. Sehingga, akan terciptanya *error handling* terhadap semua kemungkinan yang ada yang mungkin tidak terpikir pada awal perancangan perangkat lunak.

5.2 Saran

1. *State pattern* dapat dikembangkan sebagai sistem *monitoring* terhadap *states* yang ada pada perangkat lunak, seperti memantau status barang pada proses pengiriman barang.
2. Agar menghasilkan nilai *object-oriented* yang lebih akurat, dapat digunakan tools untuk menghitung parameter-parameter yang ada secara otomatis, sehingga dapat membantu mempercepat dalam proses analisis.

DAFTAR PUSTAKA

[1]. Anonymous. 2008. *Design Pattern (Computer Science).* Dikutip : 23 Juli 2009, [online]. Available : http://en.wikipedia.org/wiki/Design_ pattern_%28computer_ science%29.

[2]. *Anonymous. 2009. Object Oriented Analysis (OOA).* Dikutip : 23 Juli 2009, [online]. Available : http://en.wikipedia.org/wiki/Object-oriented_analysis_and_design.

[3]. *Anonymous. 2009. Object Oriented Design (OOD).* Dikutip : 23 Juli 2009, [online]. Available : http://en.wikipedia.org/wiki/Object-oriented_design.

[4]. Anonymous. 2008. *State Pattern.* Dikutip : 18 November 2008, [online]. Available : http://en.wikipedia.org/wiki/State_pattern.

[5]. Chidamber, Shyam R. and Kemerer, Chris F. 1994. A Metrics Suite for Object Oriented Design. Transaction on Software Engineering.

[6]. Freeman, Eric & Freeman, Elisabeth. 2004. *Head First Design Pattern*, First Edition, O'Reilly Media, Inc.

[7]. Jasnowski, Mike. 2004. *Java, XML, and Web Service.* Third Edition, Hungry Minds.

[8]. Kusumaningrum, Desty. 2008. *Tugas Akhir: Implementasi Refactoring Untuk Mendapatkan Abstract Factory Design Pattern Pada Kode Berbasis Object Oriented (Studi Kasus : Sistem Kos-kosan).* Fakultas Teknik Informatika Institut Teknologi Telkom.

[9]. Larman, Craig. 1999. *Applying UML and Patterns.* Third Edition, Prentice Hall PTR.

[10]. Liem, Inggriani. 2003. *DIKTAT KULIAH Pemrograman Berorientasi Objek.* Fakultas Teknik Informatika Institut Teknologi Bandung.

[11]. Martin, Robert. 1994. *OO Design Quality Metrics.* Cranbrook Road.

[12]. Rosenberg, Linda H. 1998. *Applying and Interperting Object oriented Metrics.* Sowftware Assurance Technology Center.

[13]. Salea, Trisya Kansya. 2008. *Tugas Akhir: Analisis dan Implementasi Refactoring Untuk Mendapatkan Decorator Design Pattern pada Kode Pemrograman Berbasis Objek.* Fakultas Teknik Informatika Institut Teknologi Telkom.

[14]. Schach, S. R. 1997. Software Engineering with Java. Times Mirror Higher Education Group.

[15]. Schroeder, Mark. 1999. *A Practical Guide to Object-Oriented Metrics.* Dikutip : 18 November 2008, [online]. Available : http://www.cin.ufpe.br/~inspector/relacionados/ metricsbymark.pdf.

[16]. Syst A, Tarja dan Yu, Ping. 1999. *Using OO Metrics and Rigi To Evaluate Java Software*. Dikutip : 18 November 2008, [online]. Available : http://www.cs.uta.fi/reports/A-1999-9.ps.Z.

[17]. Tar, Bob. *The State and Strategy Pattern.* Dikutip : 23 Juli 2009, [online]. Available : *userpages.umbc.edu/~**tarr**/dp/lectures/**StateStrategy**-2pp.pdf.*

[18]. Tim Asisten. 2009. *MODUL PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK 2008-2009*. Fakultas Teknik Informatika Institut Teknologi Telkom.