IMPLEMENTASI MESIN TURING UNTUK IDENTIFIKASI KEBUTUHAN STATE PATTERN BERDASARKAN SOURCE CODE

Dziky Dwi Rahmanto[1], Eko Darwiyanto[2], Sri Widowati[3]

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

Abstrak

Identifikasi design pattern merupakan suatu proses yang dilakukan pada saat mendesain perangkat lunak, khususnya state pattern identifikasi kebutuhannya dilihat pada source code. Hal ini dikarenakan pola code smell pada state pattern yang hanya dapat terlihat pada source code-nya. Mencari suatu pola pengkodean dalam suatu source code bukanlah hal yang mudah dan cepat, selain diharuskan memahami isi source code, terkadang efek human error seperti terlewatnya suatu baris kode saat membaca source code kerap kali terjadi. Oleh karena itu dibutuhkan suatu perangkat lunak yang dapat mencari pola tersebut secara otomatis.

Dalam Tugas Akhir ini telah diimplementasikan mesin turing untuk menyelesaikan permasalahan pencarian pola – pola code smell state pattern.. Mesin turing dipilih karena kemampuannya sangat mendukung proses pencarian pola tersebut dalam suatu source code.

Hasil analisis nilai akurasi hasil identifikasi perangkat lunak yang dilihat berdasarkan hasil identifikasi para pakar OO membuktikan bahwa mesin turing hanya dapat membaca pola secara sintaktik saja, namun tidak semantiknya.

Kata Kunci : state pattern, identifikasi, code smell, mesin turing, source code, akurasi

Abstract

Identify design pattern is a process performed when designing software, especially state pattern, its identification of needs seen in the source code. It is because the pattern of state pattern‟s code smell can be detected only from its source code. Finding a code pattern in a source code is not easy and fast, beside we need to understand the source code, sometime the human error effect like skipping a line of code when we read the source code, often happen. That‟s why a software to find its pattern automatically is needed.

In this Final Project, a turing machine has been implemented to solve the problem of finding pattern of state pattern‟s code smell‟s. Turing machine was chosen for its ability to strongly support the process of finding those patterns in a source code.

The results of the identification accuracy value analysis software that is seen on the results of the identification of OO experts proved that a Turing machine can only read a syntactic patterns, but not semantic.

Keywords : state pattern, identification, code smell, turing machine, source code, accuration

1. PENDAHULUAN

1.1 Latar Belakang

Mengetahui kebutuhan akan suatu *design pattern* dalam suatu perangkat lunak dapat meningkatkan pemahaman pengembang dalam memahami desain perangkat lunak tersebut [7]. Terutama apabila pengembang ingin mengembangkan perangkat lunak tersebut lebih lanjut. Kebutuhan akan suatu design pattern ditunjukkan dengan kemunculan *code smell* pada suatu perangkat lunak. Setiap *design pattern* memiliki *code smell* yang berbeda dari *design pattern* lainnya.

Pada *State Pattern*, salah satu pola *code smell* yang muncul adalah adanya percabangan kompleks. Berbeda dari *design patter*n lainnya yang dapat diidentifikasi kebutuhannya dari diagram kelas, *state pattern* mengharuskan pengembangnya untuk mengidentifikasi dari *source code*. Hal ini dikarenakan pola percabangan pada *code smell state pattern*, hanya dapat dilihat dari *source code*.

Seperti yang telah diketahui, mencari suatu pola pengkodean pada *source code* bukanlah hal yang mudah dan cepat, terutama apabila *source code* tersebut memiliki jumlah baris yang sangat panjang dan atau memiliki struktur yang kompleks. Selain itu ada kemungkinan terlewatnya suatu baris pengkodean saat membaca manual kerap kali terjadi. Mungkin mencari percabangan dalam suatu *source code* yang panjang bukanlah suatu hal yang sulit, namun untuk mencari percabangan yg sesuai dengan pola *code smell state pattern* hal ini akan sedikit memakan waktu. Salah satu cara menemukan pola – pola tersebut dalam suatu *source code* tanpa harus membacanya secara manual adalah dengan memanfaatkan otomata, salah satunya adalah mesin Turing.

Mesin Turing memiliki kemampuan untuk menggerakkan pointernya (*head*) ke arah kanan dan kiri pita. Mesin Turing juga memiliki kemampuan untuk membaca dan menulis di lebih dari satu pita simbol (mesin Turing *Multiple-Tape*). Jika dilihat dari karakteristik pencarian pola *code smell state pattern* yang mengharuskan kita untuk mencari suatu pola pada bagian tertentu secara berulang – ulang , yaitu memastikan suatu percabangan adalah *code smell state pattern*, kemampuan mesin turing tersebut sangat mendukung. Karena kita dapat menyalin bagian tertentu dari *source code* ke dalam pita lain (kemampuan *Multiple-Tape*) untuk dicek secara berulang – ulang (kemampuan menggerakkan pointer ke kiri dan kanan) tanpa harus mengganggu pita utamanya.

Seperti halnya pada mesin otomata lainnya, *state – state* pada mesin Turing nantinya akan dijadikan pedoman oleh perangkat lunak dalam mencari pola – pola tersebut.

1.2 Perumusan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini adalah sebagai berikut :

1. Bagaimana merepresentasikan pola *code smell state pattern* ke dalam model mesin Turing?
2. Berapa besar akurasi yang dihasilkan oleh perangkat lunak ditinjau dari hasil identifikasi para pakar *OO* ?

Batasan masalah pada Tugas Akhir ini adalh sebagai berikut :

1. Diasumsikan *source code* tidak terdapat error.
2. Data uji yang digunakan adalah *source code* berbasis Java.
3. Sintaks ekspresi percabangan harus di awali dengan '{' dan diakhiri dengan '}'.
4. Sintaks Java yang dapat dikenali oleh perangkat lunak **hanya** sintaks yang terdapat pada subbab 2.3 Sintaks Java.
5. Hasil yang dikeluarkan perangkat lunak adalah sebuah identifikasi kebutuhan akan suatu *design pattern*, **bukan** *source code* yang sudah dimodifikasi dengan *design pattern.*

1.3 Tujuan

Tujuan yang ingin dicapai oleh Tugas Akhir ini adalah :

1. Mengimplementasikan pola *code smell state pattern* ke dalam model mesin Turing.
2. Membuat suatu aplikasi yang dapat menyelesaikan permasalahan dalam pengidentifikasian *State Pattern*.
3. Menganalisis nilai akurasi luaran perangkat lunak ditinjau dari hasil identifikasi para pakar *OO*.

1.4 Metodologi Penyelesaian Masalah

Metodologi yang digunakan dalam memecahkan masalah di atas adalah dengan menggunakan langkah-langkah berikut:

1. Studi literature

   Pencarian referensi dan sumber – sumber yang berhubungan dengan *state pattern*, mulai dari pola *code smell state pattern* hingga contoh – contohnya, mesin Turing, dan *platform* atau bahasa pemrograman untuk membangun aplikasi tersebut.
2. Pengumpulan Data

   Disini penulis mengumpulkan data pola *code smell state pattern* untuk membangun mesin turing perangkat lunak dan mengumpulkan data uji berupa *file source code* Java.
3. Pengembangan Perangkat lunak

   Menganalisa pola *code smell state pattern* sehingga dapat direpresentasikan ke dalam mesin turing. Melakukan perancangan terhadap perangkat lunak yang dibangun. Penentuan platform, arsitektur, fungsionalitas dan antarmuka perangkat lunak dilakukan juga pada tahap ini.
4. Implementasi dan Pembangunan system

Membangun suatu perangkat lunak yang telah dirancang pada tahap sebelumnya dengan berdasar pada mesin turing yang telah dibuat.

5. Pengujian dan Analisis

Pengujian dilakukan dengan cara menguji perangkat lunak yang telah dibangun dengan beberapa data uji dari pada pakar *OO* sekaligus menghitung nilai akurasi hasil identifikasi perangkat lunak dilihat berdasarkn hasil identifikasi pada pakar *OO*. Kemudian dilakukan analisis terhadap nilai akurasi tersebut.

5. PENUTUP

5.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian, diperoleh kesimpulan sebagai berikut:

1. Mesin turing dapat digunakan untuk merepresentasikan pola – pola *code smell state pattern*.
2. Mesin turing hanya dapat membaca pola pengkodean secara sintaktik, namun tidak secara semantik.
3. Pola yang diidentifikasi oleh mesin turing belum tentu merupakan *code smell state pattern*.
4. Identifikasi suatu *design pattern* tidak dapat dilihat hanya dari sisi sintaktik pengkodean saja, tetapi juga harus dilihat sisi semantiknya.
5. Untuk menentukan kebutuhan suatu *source code* terhadap *state pattern*, mesin turing saja tidak cukup, dibutuhkan bantuan manusia untuk menganalisis pola *code smell* yang ditemukan oleh mesin turing.

5.2 Saran

Berdasarkan hasil analisis dan kesimpulan, peneliti memiliki beberapa saran untuk penelitian berikutnya :

1. Dilakukan penelitian lebih lanjut tentang pengimplementasian mesin turing untuk mengenali pola – pola code smell design pattern yang lain.
2. Dilakukan penelitian lebih lanjut agar mesin turing tidak hanya mendeteksi pola berdasarkan secara sintaktik, tetapi juga semantik.

Daftar pustaka

[1] Anonymous. 2008. *Turing machine simulator (Java)*. Diakses : 13 Juni 2011, [online]. Available : http://en.literateprograms.org/Turing_machine_simulator_%28Java%29#chunk%20def:TuringSimulator%20constructor.

[2] Anonymous. 2011. *Automata Theory*. Dikutip : 15 September 2011, [online]. Available : http://en.wikipedia.org/wiki/Automata_theory.

[3] Anonymous. 2011. *Java Syntax*. Dikutip : 14 September 2011, [online]. Available : http://en.wikipedia.org/wiki/Java_syntax.

[4] Brookshear, J. Glenn. 1989. *THEORY OF COMPUTATION FORMAL LANGUAGES, AUTOMATA, AND COMPLEXITY*, The Benjamin/Cummings Publishing Company, Inc.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

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

[7] Fukaya, Kazuhiro, Kubo, Atsuto, Washizaki, Hironori, dan Fukazawa, Yoshiaki. *Design pattern Detection Using Source code of Before Applying Design patterns. Eighth IEEE/ACIS International Conference on Computer and Information Science*, 2009.

[8] Hariyanto, Bambang , Ir., MT. 2007. *Esensi – Esensi Bahasa Pemrograman JAVA.* Informatika Bandung.

[9] Hendro, Steven. 2008. *Aturan Identifier*. Dikutip : 14 September 2011, [online]. Available : http://sinau-java.blogspot.com/2008/05/aturan-identifier.html.

[10] Jacobs, Aaron. *GoF State Pattern.* Diakses : 14 September 2011. [online]. Available : h*ttp://www.cse.wustl.edu/~cdgill/courses/cse432/Klein.State.pptx.*

[11] Muttaqin, Fahrul. *Tugas Akhir : Analisis dan Implementasi State Pattern.* Fakultas Teknik Informatika Institut Teknologi Telkom.

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