DESAIN DAN IMPLEMENTASI PEMROGRAMAN BERORIENTASI-ASPEK STUDI KASUS : PERANGKAT LUNAK BANTU ADMINISTRASI KLINIK KEBIDANAN DAN PENYAKIT KANDUNGAN UMMI BENGKULU

Ridho Nur Imansyah[1], -[2]

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

Abstrak

Salah satu teknik pemrograman yang kini populer adalah pemrograman Berorientasi-Objek yang membungkus atribut dan operasi menjadi unit modularitas yang disebut objek. Pada teknik pemrograman ini terkadang ditemui crosscutting concern yang merupakan fungsionalitas yang tidak dapat dienkapsulasi ke dalam satu objek. Crosscutting concern dapat menyebabkan masalah dalam evolusi dan pemeliharaan perangkat lunak. Sayangnya pemrograman Berorientasi-Objek tidak menyediakan mekanisme khusus untuk menangani masalah ini.

Tugas akhir ini menerapkan pemrograman Berorientasi-Aspek sebagai suatu cara membungkus crosscutting concern sehingga lebih mudah ditangani. Pembangunan perangkat lunak dimulai dengan analisis dan desain Berorientasi-Objek. Diagram kelas kemudian dikembangkan lebih lanjut untuk memodelkan aspek-aspek yang ditambahkan pada perangkat lunak. Implementasi pemrograman Berorientasi-Aspek dilakukan berdasarkan desain yang dilengkapi dengan pemodelan aspek. Sebagai studi kasus, tugas akhir ini menggunakan perangkat lunak bantu administrasi pada Klinik Kebidanan dan Penyakit Kandungan UMMI Bengkulu. Dari hasil analisis, didapatkan kesimpulan bahwa pemrograman Berorientasi-Aspek dapat meningkatkan reusabilitas, efisiensi (dalam hal ukuran kode program), serta mempermudah pengembangan dan pemeliharaan perangkat lunak.

Kode program ditulis dalam Java dengan AspectJ sebagai pustaka tambahan untuk pemrograman Berorientasi-Aspek pada Java. Pemodelan dibuat dalam UML dengan menggunakan Rational Rose.

Kata Kunci : objek, aspek, aspectj, java, crosscutting concern

Abstract

One programming technique that popular nowadays is Object-Oriented programming which encapsulates attributes and operations into single unit called object. In this programming technique, we might find crosscutting concern, which is a functionality that can not be encapsulated into an object. Crosscutting concern can cause problems in evolution and maintenance of the software. Unfortunately, Object-Oriented programming does not provide any special mechanism to handle these problems.

This final project applied Aspect-Oriented programming as a way to wrap crosscutting concerns so they become easier to handle. Software development began with Object-Oriented analysis and design. Then the class diagram was expanded to model the aspects which was added in software. Implementation with Aspect-Oriented Programming was done based on the design which had been extended with aspects modeling. As case study, this final project used administration software at Clinic of Obstetric and Gynecology UMMI Bengkulu. From the analysis, it had been concluded that Aspect-Oriented Programming can improve reusability, eficiency (in term of the size of the code), and also make it easier for program to evolve and maintain.

The program was written in Java with AspectJ as extended library for Aspect-Oriented programming in Java. Modeling was made in UML by using Rational Rose.

Keywords : object, aspect, aspectj, java, crosscutting concern

BAB I
PENDAHULUAN

I.1 Latar Belakang

Dunia rekayasa perangkat lunak berkembang sangat pesat, termasuk dalam teknik pemrograman. Dari pemrograman yang tiap barisnya diinterpretasikan secara sekuens dan hampir tidak mengenal modularitas, hingga pemrograman prosedural dan kemudian pemrograman yang berorientasi pada objek. Peningkatan teknologi memungkinkan para pengembang perangkat lunak untuk memandang suatu domain masalah secara lebih sederhana, yaitu dengan adanya pemisahan bagian-bagian tertentu dari perangkat lunak sehingga lebih mudah diidentifikasi dan dimanipulasi, atau yang biasa disebut pemisahan *concern*. Kebutuhan akan modularitas semacam ini kemudian menghantarkan pada pemrograman yang berorientasi pada Objek (*Object-Oriented programming*).

Pemrograman Berorientasi-Objek memandang domain masalah sebagai sekumpulan objek yang mempunyai atribut dan kelakuan yang spesifik. Objek disini merupakan suatu bentuk pemisahan *concern* sehingga lebih mudah diidentifikasi dan dienkapsulasi. Objek-objek yang telah diidentifikasi mempunyai data atau atribut dan dapat dimanipulasi dengan sekumpulan fungsi (metode, layanan, operasi). Objek juga dapat berinteraksi antara satu sama lain dengan mekanisme pertukaran pesan. Cara pandang ini membuat pemrograman Berorientasi-Objek telah menjadi metode yang cukup populer dalam pengembangan perangkat lunak. Objek menjadi kunci utama keberhasilan metode ini karena mengurangi tingkat kompleksitas dalam pemrograman dan memungkinkan pemeliharaan yang relatif lebih mudah terhadap perangkat lunak itu sendiri dibandingkan dengan metode konvensional.

Namun masalah kemudian muncul saat ditemui *crosscutting concern*, yaitu *concern* yang merupakan fungsionalitas yang tidak dapat dienkapsulasi ke dalam satu objek. *Crosscutting concern* dapat menyebabkan masalah dalam evolusi dan pemeliharaan perangkat lunak. Sayangnya pemrograman Berorientasi-Objek tidak menyediakan mekanisme khusus untuk menangani masalah ini. Hal inilah yang kemudian mendorong lahirnya pemrograman Berorientasi-Aspek (*Aspect-Oriented programming*).

I.2 Perumusan Masalah

Berdasarkan uraian pada latar belakang, tampak bahwa permasalahan timbul saat ditemukannya *crosscutting concern*, karena pemrograman Berorientasi Objek tidak mempunyai mekanisme khusus dalam menangani hal tersebut. Hal ini membutuhkan suatu teknik pemrograman lain yang dapat menangani *crosscutting concern*, yaitu pemrograman Berorientasi-Aspek.

Untuk mendapatkan gambaran yang cukup mengenai permasalahan yang dapat timbul, maka dalam tugas akhir ini akan digunakan studi kasus yaitu perangkat lunak bantu administrasi Klinik Kebidanan dan Penyakit Kandungan UMMI Bengkulu. Perangkat lunak semacam ini cukup umum dan jamak ditemui, dengan fungsi-fungsi standar seperti penyimpanan data, pengolahan data dan penampilan informasi. Dengan bentuk yang telah dikenal, diharapkan pembahasan dapat lebih terfokus pada penerapan pemrograman Berorientasi-Aspek.

I.3 Tujuan Pembahasan

Tujuan tugas akhir ini adalah menerapkan pemrograman Berorientasi-Aspek dalam level desain dan implementasi untuk menanggulangi masalah *crosscutting concern*. Melalui tugas akhir ini diharapkan dapat lebih mengenal karakteristik pemrograman Berorientasi-Aspek dan juga mengetahui sejauh mana teknik pemrograman ini dapat memberikan nilai tambah pada perangkat lunak, khususnya dalam hal efisiensi program (dilihat dari ukuran program yang dibuat), reusabilitas, dan kemudahan dalam pengembangan perangkat lunak.

Pembangunan perangkat lunak akan dimulai dengan analisis dan desain Berorientasi-Objek. Diagram kelas yang didapat kemudian akan dikembangkan lebih lanjut untuk memodelkan aspek-aspek yang telah diidentifikasi. Implementasi pemrograman Berorientasi-Aspek akan dilakukan berdasarkan desain yang dilengkapi dengan pemodelan aspek.

I.4 Batasan Masalah

Untuk menghindari terlalu meluasnya pembahasan tugas akhir ini, maka terdapat batasan masalah sebagai berikut :

- Tugas akhir ini hanya akan menerapkan Pemrograman Berorientasi-Aspek sejak level desain hingga implementasi.
- Tugas akhir ini tidak menganalisis sisi teknis dari pemrograman (kecepatan proses, pemakaian memori, dll).
- Perangkat lunak yang dibangun berbasis stand-alone; menangani perincian layanan untuk tiap pasien, penghitungan tagihan, pemberian laporan kepada pihak pengelola klinik, serta pemberian informasi seputar data pasien. Perangkat lunak ini tidak menangani pengelolaan obat, kepegawaian, dan keuangan klinik.
- Untuk mendukung perangkat lunak, akan digunakan basisdata yang menyimpan data-data yang diperlukan. Namun, basisdata sendiri bukan merupakan pembahasan utama dalam tugas akhir ini.
- Analisis dan desain perangkat lunak akan dimodelkan dalam UML dengan menggunakan Rational Rose.
- Bahasa yang dipakai dalam pemrograman adalah Java dengan menggunakan Borland JBuilder sebagai IDE dan AspectJ sebagai pustaka tambahan untuk implementasi pemrograman Berorientasi-Aspek.
- Database yang digunakan adalah Oracle.
- Sistem Operasi yang digunakan adalah Microsoft Windows XP.

I.5 Metodologi Pemecahan Masalah

Metode yang akan digunakan untuk menyelesaikan tugas akhir ini adalah :

- Studi literatur, yaitu dengan mempelajari literatur-literatur yang relevan dengan permasalahan yang meliputi studi pustaka dan pencarian referensi tentang pemrograman Berorientasi-Aspek, serta Java dan AspectJ sebagai bahasa yang digunakan dalam pemrograman.

- Pengumpulan data dan informasi yang dibutukan dalam pembangunan perangkat lunak, termasuk melalui proses wawancara dengan user mengenai fungsi yang diinginkan dari aplikasi, dan juga pengumpulan data-data contoh.
- Analisis dan desain perangkat lunak tahap I dengan menggunakan metode analisis dan desain berorientasi obyek. Dalam hal ini, bahasa pemodelan yang akan digunakan adalah UML (Unified Modeling Language).
- Desain perangkat lunak tahap II dengan melakukan pengembangan dari model desain yang telah dibuat pada tahap I ke dalam bentuk desain Berorientasi-Aspek.
- Implementasi perangkat lunak yaitu implementasi pemrograman berdasarkan desain tahap II yang telah dibuat. Bahasa pemrograman yang akan dipakai adalah Java dengan AspectJ sebagai modul tambahan untuk implementasi pemrograman Berorientasi-Aspek.
- Uji coba aplikasi.
- Penyusunan laporan tugas akhir dan kesimpulan akhir.

I.6 Sistematika Penulisan

BAB I PENDAHULUAN

Berisi latar belakang, perumusan masalah, tujuan pembahasan, batasan masalah, metodologi penyelesaian masalah dan sistematika penulisan.

BAB II LANDASAN TEORI

Membahas teori dan konsep dasar mengenai pemrograman Berorientasi-Aspek.

BAB III ANALISIS DAN DESAIN PERANGKAT LUNAK

Membahas tentang analisis dan desain perangkat lunak dengan pendekatan Berorientasi-Objek sebagai dasar penerapan pemrograman Berorientasi-Aspek, dilengkapi dengan desain basisdata yang digunakan.

BAB IV DESAIN, IMPLEMENTASI DAN ANALISIS PEMROGRAMAN BERORIENTASI-ASPEK

Membahas tentang implementasi pemrograman Berorientasi-Aspek dalam pembangunan perangkat lunak serta pembahasan lebih lanjut mengenai karakteristik pemrograman Berorientasi-Aspek secara umum.

BAB V KESIMPULAN DAN SARAN

Berisi kesimpulan akhir dan saran pengembangan lebih lanjut.

BAB V
PENUTUP

V.1 Kesimpulan

1. Untuk memodelkan aspek pada tahap desain, dapat digunakan mekanisme ekstensi pada UML seperti yang telah dilakukan pada tahap desain dalam pengembangan perangkat lunak di tugas akhir ini.
2. Pemrograman Berorientasi-Aspek dapat meningkatkan reusabilitas dan modularitas, ditunjukkan dengan mudahnya aspek *Tracing* diterapkan kembali pada dua perangkat lunak lain, yaitu Sistem Peminjaman Perpustakaan dan Aplikasi Administrasi Kursus Komputer. Hal ini membuktikan bahwa pemrograman Berorientasi-Aspek mempermudah pengembangan serta proses pemeliharaan perangkat lunak. Namun tetap merupakan hal yang penting untuk memahami konsep Berorientasi-Objek sebelum mengimplementasikan pemrograman Berorientasi-Aspek pada program.
3. Pemrograman Berorientasi-Aspek mempermudah pengembangan perangkat lunak dalam hal penambahan *concern* teknis baru pada program karena program yang telah ada tidak perlu mengalami perubahan, sebaliknya aspek yang akan mengenali tempat disisipkannya baris-baris fungsionalitas yang akan ditambahkan, contohnya pada penambahan fitur autentikasi, *tracing*, *logging*, dan pengecekan *query* yang telah diterapkan pada studi kasus.
4. Pemrograman Berorientasi-Aspek dapat meningkatkan efisiensi jika dilihat dari ukuran kode program yang dihasilkan. Pada studi kasus, ukuran kode program yang dibuat dengan pemrograman Berorientasi-Aspek berkurang sebesar 22.19% dalam metrik LoC dan berkurang sebesar 17.97% dalam metrik ABC.
5. Pemrograman Berorientasi-Aspek mempermudah dalam hal mengaktifkan dan menonaktifkan *concern* teknis yang hanya diperlukan pada saat tertentu. Misalnya saat fungsionalitas *tracing* tidak perlu dilakukan, maka cukup menghilangkan aspek atau tidak memasukkan aspek pada proses kompilasi.

6. Pointcut pada pemrograman Berorientasi-Aspek bekerja berdasarkan aturan penamaan kelas dan method, maka *naming convention* dan kehati-hatian dalam menggunakan *wildcard* menjadi yang sangat penting untuk diperhatikan.

V.2 Saran

1. Pemrograman Berorientasi-Aspek hanyalah bagian kecil dari keseluruhan kerangka pembangunan perangkat lunak Berorientasi-Aspek (AOSD). Bidang masih dalam tahap pengembangan dan penelitian, sehingga banyak sekali hal yang dapat dijadikan bahan penelitian lebih lanjut. Hal ini mencakup analisis dan desain Berorientasi-Aspek sejak awal, hingga pada uji coba khusus Berorientasi-Aspek, baik berupa *integration testing*, maupun *unit testing*.
2. Selain penerapan pemrograman Berorientasi-Aspek sejak desain yang berupa penambahan *concern* teknis seperti yang dilakukan pada tugas akhir ini, penerapan pemrograman Beorientasi-Aspek juga dapat dilakukan dengan *refactoring* dengan Berorientasi-Aspek.

DAFTAR PUSTAKA

[1] AspectJ Team. 2003. *The AspectJTM Programming Guide*. Palo Alto: Palo Alto Research Center.

[2] Booch, Grady. Jacobson, Ivar. Rumbaugh, James. 1999. *The Unified Modeling Language User Guide*. Massachusets: Addison-Wesley.

[3] Fitzpatrick, Jerry. 1997. *Applying ABC Metrics to C, C++, and Java*. C++ Report.

[4] Hariyanto, Bambang. 2004. *Rekayasa Sistem Berorientasi Objek*. Bandung: Penerbit Informatika.

[5] Java Team. 2002. *JavaTM 2 SDK, Standard Edition Documentation*. Sun Microsystem. Inc.

[6] Mahendra, Adhari C. 2004. *Pengantar Pemrograman Berbasis Aspek (AOP)*. www.ilmukomputer.com.

[7] Pressman, Roger S. 2001. *Software Engineering: A Practitioner's Approach, Edisi Kelima*. New York: McGraw-Hill.

[8] Suhendar, A. Gunandi, Hariman. 2002. *Visual Modeling Menggunakan UML dan Raitonal Rose*. Bandung: Informatika.

[9] Utdirartatmo, Firrar. 2003. *Aplikasi DataBase di Java dengan JBuilder*. Jakarta: Elex Media Computindo.

[10] Zakaria, Aida Atef. 2002. *A UML Extension for Modeling Aspect-Oriented Systems*. Cairo: The American University in Cairo.