ANALISIS IMPLEMENTASI ASPECT ORIENTED PROGRAMMING (AOP) DAN DEPENDENCY INJECTION (DI) PADA KUALITAS DESAIN APLIKASI JEE, STUDI KASUS IMPLEMENTASI SPRING FRAMEWORK PADA APLIKASI JEE

Adhika Subhaga Trahkasno[1], Dana Sulistyo Kusumo[2], Kusuma Ayu Laksitowening[3]

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

Abstrak

Proses bisnis dari suatu organisasi akan selalu mengalami perubahan, sehingga aplikasi yang digunakan pada organisasi tersebut juga akan ikut berubah. Untuk itu bukan hanya platform pengembangan saja yang membuat aplikasi dapat lebih menyesuaikan dengan perkembangan kebutuhan proses bisnis, tetapi juga kualitas desain aplikasi.

JEE adalah suatu platform pengembangan yang terstandarisasi dan memiliki API yang lengkap. Pendekatan AOP dan DI memiliki kelebihankelebihan melalui prinsip Inversion of Control. Oleh karena itulah penulis berusaha untuk mengetahui hubungan AOP dan DI terhadap kualitas desain aplikasi JEE. Spring framework digunakan sebagai studi kasus karena memberikan dukungan terhadap AOP dan DI.

Dengan AOP dan DI dari spring framework akan dianalisis tingkat ketergantungan (coupling) fungsional concern terhadap nonfungsional concern dari suatu aplikasi JEE. Sebagai perkakas untuk analisis digunakan package dependencies metrics, CK metrics suite untuk AOP, dan LOCC metrics.

Dari hasil perhitungan seluruh metrik tersebut dapat disimpulkan bahwa AOP dan DI dari spring framework dapat menurunkan tingkat ketergantungan (coupling) fungsional concern terhadap nonfungsional concern dan juga mengurangi duplikasi kode dari aplikasi JEE. Yang pada akhirnya meningkatkan kualitas desain aplikasi JEE.

Kata Kunci : AOP, DI, JEE, kualitas desain aplikasi, spring framework

Abstract

The organization business process will always change. However the application will be changed to. Due to that change not only environment development that make application more adaptable with business process requirements growth but also application design quality.

JEE is standardized environment development and have rich API. AOP and DI approach have advantages by Inversion of Control principle. However the writer wants to know the relation between AOP and DI toward JEE application design quality. Spring framework is used as cases study because have support to AOP and DI.

With AOP and DI of spring framework will be analyzed coupling between functionality concern and non functionality of some JEE application. As tool for analyzing, package dependencies metrics, CK metrics suite for AOP, and LOCC metrics are used.

From calculation of all those metrics can be summarized that AOP and DI of spring framework can decrease coupling between functionality concern and non functionality concern and also reduce code duplication of JEE application. Finally increase the design quality of JEE application.

Keywords : AOP, DI, JEE, application design quality, spring framework

1. Pendahuluan

1.1 Latar Belakang

Dewasa ini, dengan adanya teknologi-informasi proses-proses yang terjadi di kehidupan sehari-hari manusia dapat dilakukan dengan lebih mudah. Semisal adalah proses pengiriman uang, jika dahulu uang dikirim dengan menggunakan kurir maka sekarang uang dapat dikirim dengan hanya mengakses layanan e-banking yang disediakan oleh bank. Atau ketika hendak membeli suatu barang maka transaksi jual beli tinggal dilakukan melalui situs yang disediakan oleh si penjual, yang dapat dilakukan dimana saja dan kapan saja. Tetapi pernahkah terpikirkan proses yang berjalan di belakang itu semua? Sekarang beralih pada sudut pandang si pembuat aplikasi. Ketika proses bisnis organisasi masih sederhana (mungkin hanya pembelian online) maka tidak akan terlalu terpikirkan bagaimanakah manajemen transaksi aplikasi atau bagaimanakah arsitektur web server aplikasi. Clustering web server masih belum perlu untuk dilakukan dan protokol transaksi dua fase juga belum perlu untuk diimplementasikan. Pembagian aplikasi menjadi beberapa layer aplikasi pun belum terlalu dipikirkan.

Ketika proses bisnis organisasi pun menjadi semakin kompleks maka hal-hal yang dijelaskan oleh penulis di atas (manajemen transaksi, arsitektur aplikasi red) adalah suatu kebutuhan. Maka jika pengembang aplikasi tidak mengantisipasi hal-hal tersebut yang terjadi adalah perombakan seluruh aplikasi dan pergantian pada lingkungan atau platform development yang menyediakan layanan-layanan di atas atau yang lebih ekstrim membangun sendiri API yang dibutuhkan.

Pada saat seperti itulah JEE mengambil peran, dengan sifatnya yang terbuka (para vendor bebas untuk mengimplementasikan spesifikasi dari JEE), terstandarisasi (sun sebagai pembuat standar), dan API yang lengkap menjadikan JEE sebagai salah satu solusi pada development aplikasi enterprise. Salah satu keunggulan lain dari JEE adalah layanan yang tersedia di setiap layer aplikasi. Mulai dari manajemen data source (JNDI) , manajemen persistence (JPA), EJB, manajemen transaksi (JTA), Servlet, JSP, dan Web Service. Istilah yang digunakan jika layanan JEE di setiap layer digunakan pada aplikasi sesuai dengan tuntunan dari sun adalah menggunakan sun java blueprints.

Namun, lingkungan atau platform development yang mendukung saja belumlah cukup untuk membuat aplikasi yang dapat berevolusi sesuai dengan kebutuhan proses bisnis organisasi. Desain aplikasi juga harus berkualitas sehingga dapat mengantisipasi perkembangan-perkembangan proses bisnis organisasi. Hendaknya desain aplikasi dapat mengantisipasi perubahan kebutuhan proses bisnis organisasi (fungsional *concern*) dan kebutuhan teknis (nonfungsional *concern*) di masa mendatang.

Pada sisi desain inilah AOP dan DI diharapkan dapat mengambil peran. AOP dan DI pada literatur digolongkan menjadi Inversion Of Control karena mempersilahkan suatu objek menjadi pihak yang pasif, yaitu dengan mendelegasikan pengaturan yang dimiliki oleh objek tersebut terhadap fitur atau aspek yang dimilikinya kepada framework [5]. AOP adalah suatu paradigma pemrogaman untuk pemisahan concern-concern dan untuk pemodulan *crosscutting concern* pada suatu entitas software yang disebut *aspect* [12]. Sedangkan contoh penggunaan dari AOP adalah pembersihan code dari *tangling* dan *scattering*. Kode *tangling* disebabkan ketika suatu modul yang

diimplementasikan mengemban beberapa concern secara bersamaan sedangkan kode *scattering* disebabkan ketika suatu concern diimplementasikan pada banyak modul[6]. DI sendiri adalah suatu mekanisme untuk memberikan atau menginjeksi dependensi-dependensi yang dimiliki oleh suatu objek [2]. Salah satu contoh penggunaan DI adalah penyederhanaan mekanisme pemenuhan dependensi terhadap data source yang berhubungan erat dengan layanan JNDI.

Dengan kelebihan-kelebihan yang dimiliki oleh AOP dan DI inilah, penulis berusaha untuk mengimplementasikan keduanya pada aplikasi JEE. Diharapkan dengan pengimplementasian AOP dan DI pada aplikasi JEE, dapat meningkatkan kualitas desain pada aplikasi tersebut. Spring framework digunakan pada implementasi aplikasi JEE karena menyediakan dukungan terhadap AOP dan DI.

1.2 Perumusan Masalah

Dari latar belakang yang telah penulis jabarkan diatas, bukan hanya platform pengembangan saja yang membuat aplikasi dapat lebih menyesuaikan dengan perkembangan kebutuhan proses bisnis organisasi, tetapi juga kualitas desain aplikasi. Oleh karena itulah penulis merumuskan rumusan masalah sebagai berikut.

1. Bagaimana pengaruh AOP dan DI dari spring framework terhadap kualitas desain suatu aplikasi JEE?
2. Dan untuk menjawab pertanyaan pertama, penulis mengajukan sebuah pertanyaan yang lebih khusus, yaitu bagaimana perbandingan tingkat ketergantungan fungsional *concern* terhadap nonfungsional *concern* suatu aplikasi JEE yang mengimplementasikan AOP dan DI dari spring framework dengan aplikasi JEE yang hanya mengimplementasikan sun java blueprints?

1.3 Tujuan

1. Untuk mengetahui perbandingan kualitas desain suatu aplikasi JEE yang mengimplementasikan AOP dan DI dari spring framework dengan aplikasi JEE yang hanya mengimplementasikan sun java blueprints.
2. Untuk mengetahui perbandingan tingkat ketergantungan fungsional *concern* terhadap nonfungsional *concern* suatu aplikasi JEE yang mengimplementasikan AOP dan DI dari spring framework dengan aplikasi JEE yang hanya mengimplementasikan sun java blueprints.

1.4 Batasan Masalah

1. Kualitas desain aplikasi pada tugas akhir ini dibatasi hanya pada aspek *coupling* dari suatu aplikasi. Dan tidak membahas aspek yang lain seperti *cyclomatic complexity* dan sebagainya.
2. *Crosscutting concern* yang akan dimodelkan, dirancang, dan diimplementasikan dengan AOP dan DI dari spring framework adalah concern yang merupakan nonfungsional concern saja.
3. Pendekatan atau metodologi yang dipakai untuk mengimplementasikan fungsional *concern* (*core concern*) pada tugas akhir ini adalah pendekatan OOP.
4. Aplikasi yang akan dianalisis pada tugas akhir ini adalah aplikasi client web Duke's Bank yang disertakan pada paket source dari JEE 5 tutorial.
5. Tugas akhir ini tidak memperhitungkan segala sesuatu yang berhubungan dengan performansi aplikasi, performansi database, tingkat kualitas keamanan, atau protokol jaringan yang digunakan.

1.5 Metodologi Penyelesaian Masalah

1. Studi literatur terhadap :
   a) Konsep AOP dan DI.
   b) API AOP dan DI yang disediakan oleh spring framework.
   c) Konsep sun java blueprints.
   d) Pola-pola yang memungkinkan diterapkannya AOP dan DI.
   e) Konsep pengukuran kualitas desain perangkat lunak.
2. Mempelajari proses bisnis dan kebutuhan dari aplikasi client web Duke's Bank dengan mempelajari dokumentasi. Kemudian melakukan reverse engineering pada aplikasi client web Duke's Bank.
3. Mengidentifikasi *concern-concern*, terutama *crosscutting concern*, yang yang ada pada aplikasi client web Duke's Bank melalui use case hasil reverse engineering untuk penerapan AOP. Dan mengidentifikasi class-class yang membutuhkan DI.
4. Mendesain dan mengimplementasikan AOP dan DI pada aplikasi client web Duke's Bank dengan menggunakan spring framework.
5. Melakukan pengukuran dan analisa terhadap kualitas desain dari aplikasi client web Duke's Bank yang mengimplementasikan AOP dan DI dari spring framework dan aplikasi client web Duke's Bank yang hanya mengimplementasikan sun java blueprints dengan menggunakan *Package Dependencies Metrics Suite, CK Metrics Suite* untuk AOP, dan LOCC *Metrics*.
6. Penyusunan laporan.

5. Kesimpulan

5.1 Kesimpulan

1. Dari pengukuran dengan *Package Dependencies Metrics Suite*, penggunaan AOP dan DI dari spring framework dapat meningkatkan *independence* dan *responsibility* dari aplikasi JEE.
2. Penggunaan AOP dan DI dari spring framework dapat menurunkan tingkat ketergantungan *(coupling)* fungsional *concern* terhadap nonfungsional *concern (security, logging, persistence, validation, dan transaction management)* dari aplikasi JEE.
3. Dari pengukuran dengan *CK metrics suite* untuk AOP, penggunaan AOP dan DI dari spring framework dapat mengalihkan ketergantungan terhadap suatu class menjadi ketergantungan terhadap *aspect* dan class. Hal ini menjadikan pendekatan AOP dan DI dari spring framework membutuhkan pemanggilan *advice-advice* dari suatu *aspect* untuk melakukan fungsionalitas tertentu.
4. Dari pengukuran dengan *LOCC metrics*, penggunaan AOP dan DI dari spring framework dapat mengurangi duplikasi kode dari aplikasi JEE.
5. Penggunaan AOP dan DI dari spring framework dapat menaikkan kualitas desain aplikasi JEE, karena dengan penerapan keduanya dapat menurunkan *coupling* dan menurunkan duplikasi kode dari aplikasi JEE.

5.2 Saran

1. Pembuatan atau pengembangan suatu aplikasi JEE hendaknya menggunakan AOP dan DI.
2. Teknik pengukuran kualitas desain aplikasi dari analisis implementasi AOP dan DI pada aplikasi JEE ini hendaknya dapat memakai pendekatan lain seperti *Maintainability Index.*
3. Penelitian terhadap implementasi AOP dan DI hendaknya dapat terus dilakukan, terutama di IT Telkom yang masih belum memiliki riset khusus tentang kedua metode tersebut.

Referensi

[1] Ceccato, Mariano & Tonella, Paolo. *Measuring the Effects of Software Aspectization.* Italy: Centro per la Ricerca Scientifica e Tecnologica

[2] Interface 21 et. al.. 2007. *Building Spring 2 Enterprise Applications.* United States of America: Apress

[3] Johnson, Rod et. al.. 2008. *The Spring Framework - Reference Documentation*

[4] Jacobson, Ivar & Ng Pan-Wei. 2004. *Aspect-Oriented Software Development with Use Cases.* Massachusetts: Pearson Education, Inc.

[5] Ladd, Seth et. al.. 2006. *Expert Spring MVC and Web Flow.* United States of America: Apress

[6] Laddad, Ramnivas. 2003. *AspectJ in Action Practical Aspect-Oriented Programming*. United States of America: Manning Publications Co.

[7] Larman, Craig. 1998. *Applying UML And Patterns an Introduction to Object-Oriented Analysis and Design*. NJ: Prentice-Hall

[8] Mak, Gary. 2008. *Spring Recipes: A Problem-Solution Approach*. United States of America: Apress

[9] Martin, Robert. 1994. *OO Design Quality Metrics An Analysis of Dependencies*. Cranbrook Road Green Oaks

[10] Minter, Dave. 2008. *Beginning Spring 2: From Novice to Professional*. United States of America: Apress

[11] Mukhar, Kevin et. al.. 2006. *Beginning Java EE 5: From Novice to Professional.* United States of America: Apress

[12] Pawlak, Renaud et. al.. 2005. *Foundations of AOP for J2EE Development.* United States of America: Apress

[13] Rashid, Awais & Aksit, Mehmet (Eds.). 2007. *Transactions on Aspect-Oriented Software Development III.* Berlin: Springer-Verlag

[14] Walls, Craig & Breidenbach, Ryan .2008. *Spring In Action Second Edition*. United States of America: Manning Publications Co.

[15] ,. 2007. *The Java EE 5 Tutorial For Sun Java System Application Server 9.1.* Santa Clara: Sun Microsystems, Inc.

[16] http://www.eclipse.org/aspectj/doc/released/devguide/ltw.html. Diakses pada tanggal 10 Maret 2009

[17] https://blueprints.dev.java.net/petstore. Diakses pada tanggal 10 Maret 2009