Pengembangan Berorientasi Objek Metode Fusion

Sri Eniyati
Fakultas Teknologi Informasi, Universitas Stikubank Semarang
email : eni@unisbank.ac.id

ABSTRAK : Metode adalah "prosedur untuk melakukan sesuatu", sedang metodologi adalah "studi mengenai metode". Sistem berbasis komputer harus merupakan hasil dari suatu analisis dan dirancang sebaik mungkin. Orientasi objek merupakan salah satu pendekatan pemrograman yang memandang persoalan menggunakan model-model yang diorganisasikan seputar konsep objek yang mengkombinasikan struktur data dan perilaku suatu entitas. Ada banyak metode pengembangan sistem yang berorientasi objek. Metode Fusion merupakan salah satu metode pengembangan perangkat lunak berorientasi objek "generasi kedua". Fusion dikembangkan untuk menyediakan satu pendekatan secara sistematik bagi pengembangan perangkat lunak berorientasi objek yang mengintegrasikan dan mengembangkan beberapa pendekatan yang ada.

Kata kunci : metode fusion, pengembangan berorientasi objek

PENDAHULUAN

Sistem berbasis komputer harus merupakan hasil dari suatu analisis dan rancangan sebaik mungkin. Rancangan merupakan suatu bentuk aktivitas yang dilakukan berdasarkan suatu pendekatan yang beralasan. Pendekatan konvensional (aliran data atau terstruktur) yang tidak berdasarkan pada entitas-entitas di dunia ekstenal dan hal ini akan mempersulit di dalam mengelola dan mengadaptasikan ketika tejadi perubahan kebutuhan. Pendekatan berorientasi objek adalah efektif karena objek-objek dapat merepresentasikan bagian-bagian dari dunia eksternal, mempersempit kesenjangan (gap) konseptual antara dunia eksternal dan komponen-komponen perangkat lunak.

Metode pengembangan sistem berorientasi objek berbeda dari pengembangan konvensional yang memandang perangkat lunak sebagai fungsi dan data yang saling tidak berhubungan. Pada pendekatan konvensional, kebanyakan berfokus pada fungsi. Namun juga terdapat pendekatan yang berfokus pada data terutama pada pemakaian basisdata dan pemodelan informasi. Sedangkan pendekatan berorientasi objek berpusat pada objek yang mengkombinasikan data dan fungsionalitas. Berorientasi objek adalah konsep objek yang didalamnya terkandung data sekaligus fungsi-fungsinya.

Pendekatan berorientasi objek adalah cara memandang persoalan menggunakan model-model yang diorganisasikan seputar konsep objek yang mengkombinasikan struktur data dan perilaku suatu entitas. Pada pendekatan ini organisasi perangkat lunak adalah sebagai kumpulan objek diskrit yang saling bekerja sama, berkomunikasi dan berinteraksi menuju sasaran tertentu.

Fusion merupakan metode pengembangan perangkat lunak untuk pemakaian perangkat lunak berorientasi objek, yang secara penuh melingkupi metode yang menyediakan kemampuan analisis, design dan implementasi. Notasi Fusion memungkinkan secara sistematik menemukan dan memelihara struktur objek sistem dengan megintegrasikan dan mengembangkan pendekatan yag ada, Fusion menyediakan secara langsung lintasan dari mendefinisikan kebutuhan sampai implementasi ke bahasa pemrograman.

Fusion berdasarkan pada satu ringkasan tetapi meliputi banyak hal tentang sekumpulan notasi-notasi yang dikenali dengan baik untuk mengambil keputusan-keputusan yang berkenaan dengan analisis dan perancangan.

Apa yang ditawarkan metode Fusion ?

Fusion mendukung aspek-aspek teknis dan manajerial pengembangan perangkat lunak yaitu :

1. Menyediakan satu proses untuk pengembangan perangkat lunak. Membagi-bagi proses ke dalam beberapa phase dan menunjukkan apa yang seharusnya dilakukan di setiap phase. Memberikan petunjuk untuk tujuan apa yang akan dilakukan dalam setiap phasenya, dimana pengembang bisa mengetahui bagaimana membuat kemajuannya. Juga menyediakan kriteria yang memberitahukan ke pengembangan kapan untuk berpindah ke phase berikutnya.
2. Menyediakan yang meliputi banyak hal, ringkas, menggunakan notasi yang dikenal dengan baik untuk semua modelnya, karena notasi-notasi ini berdasarkan pada praktek yang sudah ada, juga mudah untuk dipelajari.
3. Menyediakan tool manajemen untuk pengembangan perangkat lunak. Keluaran dari setiap phase yang berbeda mudah diidentifikasi. Setiap phase memiliki teknik tersendiri dan ditujukan untuk aspek-aspek penterjemahan satu dokumentasi kebutuhan yang berbeda kedalam bentuk kode yang bisa dijalankan.

Proses Fusion

Fusion mengadopsikan pembagian proses pengembangan system menjadi tahapan analisis, design dan implementasi. Metode Fusion tidak memiliki phase kebutuhan.

TAHAPAN-TAHAPAN METODE FUSION

Berdasarkan gambar 1, menyediakan satu pemetaan lintasan metode Fusion dan menunjukkan ketergantungan diantara model-model yang dihasilkan.

Tahapan Analysis

Tahap analysis disini mengambarkan 'Apa' yang sistem lakukan daripada menjelaskan tentang 'Bagaimana' sistem itu melakukannya. Perilaku sistem dari cara diimplementasikan memerlukan gambaran sistem dari perspektif pemakai daripada cara pandang dari mesin. Jadi proses analisa difokuskan pada domain masalah dan dihubungkan dengan perilaku yang mungkin.

Tahap 1 : Mengembangkan model Objek

Tujuan dari pemodelan objek adalah mengambil konsep yang ada dalam domain masalah dan menghubungkan diantaranya. Bisa digambarkan dengan class, attribut, dan hubungan antar class.

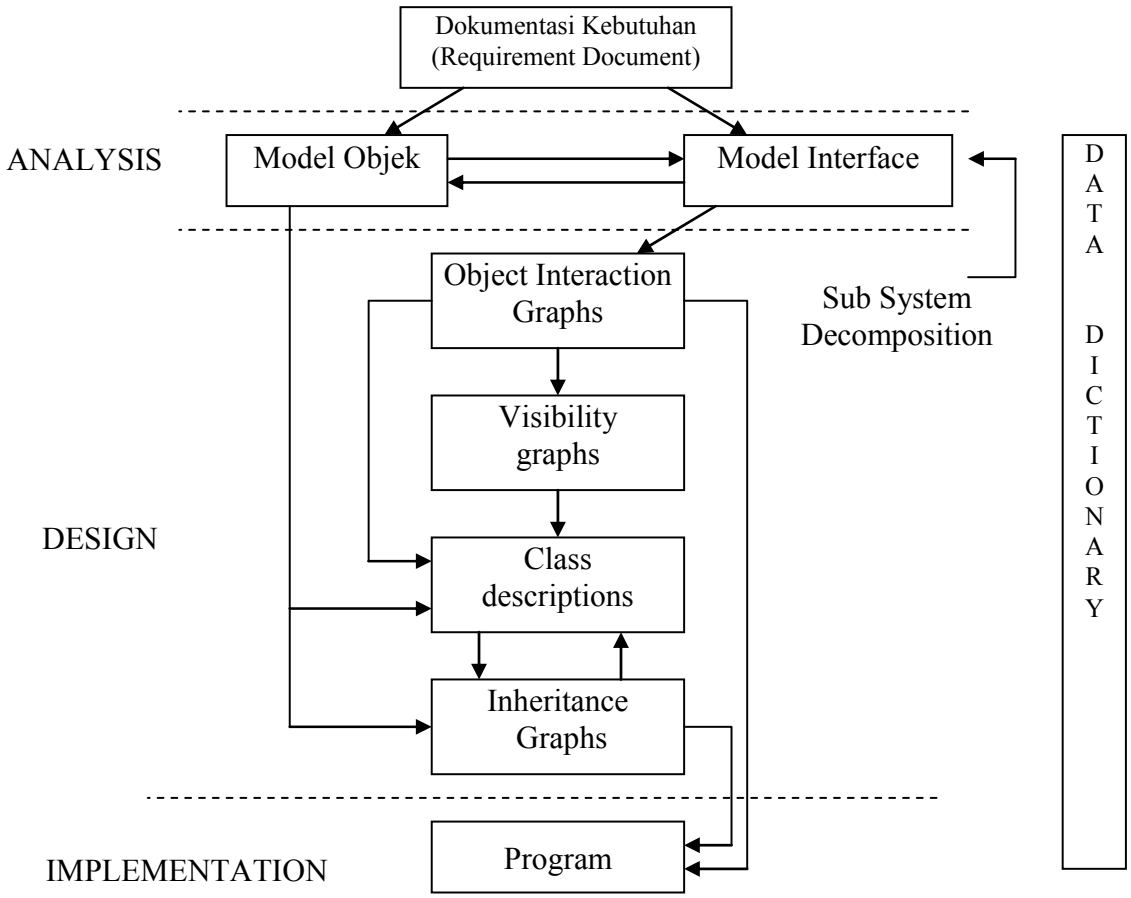
1. Mendaftarkan semua Class yang mungkin dan relasionsipnya.
2. Memasukkan Class-Class dan relasinya ke dalam Kamus data
3. Ditambah menghasilkan model objek untuk mencari
   - pemodelan "macam dari" /generalisasi" atau relasionsipnya
   - pemodelan "bagian dari"/agregasi atau relasionsipnya,
   - attribut-attribut dari Class
   - cardinalitas relasi
   - Batasan umum yang direkam di kamus data
   - Relasi yang diminta dan direkam di kamus data, tetapi tidak tampak pada model objek

Tahap 2 : Menentukan *Interface System*

Satu sistem bekerja sama dengan agen-agen (alat) aktif dalam lingkungannya. Agen meminta sistem mengoperasikan apa yang bisa diubah dari keadaan sistem dan bisa menyebabkan terjadinya keluaran.

Interface sistem merupakan sekumpulan operasi-operasi sistem yang bisa merespon dan keadaan-keadaan yang bisa dikeluarkan. Satu masukan diperlukan untuk kamus data bagi setiap agen, operasi sistem dan keadaan keluaran.

1. Mengenai agen, operasi-operasi sistem, dan keadaan



Gambar 1. Metode Fusion

2. Menghasilkan model objek sistem. Model objek sistem merupakan penghalusan dari model objek yang dikembangkan di tahap awal analisa

   - menggunakan informasi dari interface sistem, mengenali class dan relasinya pada model objek yang menyinggung keadaan sistem.
   - batasan dokumen sistem untuk menghasilkan model objek sistem.

   Teknik manfaatnya adalah untuk menentukan batasan interface yang difokuskan pada skenario-skenario pemakaian. Untuk setiap skenario mempertimbangkan

   a. Agen-agen yang dilibatkan,

   b. Apa yang ingin sistem lakukan.

   Diagram *timeline* bisa digunakan untuk menggambarkan skenario. Dengan catatan, diagram *timeline* merupakan tool yang membantu untuk mengenali batasan sistem, tetapi hanya menyediakan satu potret perilaku sistem.

Tahap 3 : Mengembangkan Model Interface
Model interface dibuat dari model siklus hidup dan model operasi

Model Siklus Hidup

Model siklus hidup memberikan definisi / menetapkan serangkaian interaksi yang mungkin dimana satu sistem bisa terlibat. Model siklus hidup ditetapkan dalam istilah ekspresi reguler yang berarti menggambarkan serangkaian pola.

1. mengeneralisasi skenario dan bentuk yang dinamakan ekspresi siklus hidup,
2. mengkombinasikan ekspresi siklus hidup ke bentuk model siklus hidup.

Model operasi

Model operasi menetapkan semantik setiap operasi sistem di sistem *interface*.

Untuk setiap operasi sistem :

1. Mengembangkan ketentuan asumsi dan hasil
2. Mengekstraksi ketentauan *Send*, *Read*, dan *Change* dari asumsi dan hasil

Tahap 4 : Cek Model Analisis

Ada dua aspek untuk mengecek model analisa; yang saling melengkapi dan konsisten. Satu model dikatakan lengkap pada saat bisa menangkap semua abstraksi yang berarti dalam domain. Satu model dikatakan konsisten pada saat tidak terjadi kontradiksi antara satu dengan yang lain. Satu model bisa dicek untuk konsistensi intenal dan juga untuk area dimana bisa terjadi overlap dengan model-model yang lain.

Dengan catatan, pengecekan berikut ini tidak lengkap tetapi menyediakan petunjuk untuk mengecek analisa :

1. *Completeness against the requirement*. Membaca kembali kebutuhan-kebutuhan dokumen secara hati-hati . Cek apakah :
   a. Semua skenario yang mungkin sudah dicover oleh siklus hidup
   b. Semua operasi sudah ditetapkan dengan skema
   c. Semua informasi statik ditangkap oleh model objek sistem
   d. Beberapa informasi lain (seperti definisi teknis, dan batasan yang berbeda) di kamus data.
2. *Simple concistency*. Pengecekan ini berkaitan dengan daerah yang overlap antar model-model analisa. Cek apakah :
   a. Semua class, relasionship, dan attribut disebutkan dalam model objek yang nampak dalam model objek sistem. Semua konsep lain (seperti predikat) harus ditetapkan di kamus data atau beberapa sumber yang ditunjuk.
   b. Batasan model objek sistem konsisten dengan model interface
   c. Semua operasi sistem dalam model siklus hidup memiliki satu skema.
   d. Semua identifier di semua model memiliki masukan di kamus data.
3. *Semantic Consistency*. Pengecekan ini bertujuan untuk meyakinkan bahwa implikasi dari model-model konsisten.
   a. Output keadaan di dalam model siklus hidup dan model operasi harus konsisten. Skema untuk operasi sistem harus menghasilkan keluaran kejadian yang berikutnya dalam model siklus hidup.
   b. Model operasi harus memelihara model objek sistem dengan batasan yang berbeda. Jika ada perbedaan berkenaan dengan relasionship atau class, maka beberapa operasi yang bisa mengubahnya harus mematuhi perbedaan dalam skemanya.
   c. Bagian cek skenario menggunakan skema. Pilih contoh-contoh skenario dan tetapkan status perubahan dimana masing-masing bisa terjadi. Kemudian ”*execute*” skenario, menggunakan skema untuk menetapkan perilaku setiap operasi sistem. Cek apakah hasilnya sesuai dengan yang diharapkan.

Tahapan Desain

Selama perancangan struktur perangkat lunak diperkenalkan untuk memenuhi definisi abstraksi yang dihasilkan dari proses analisis.

Tahapannya adalah sebagai berikut :

Tahap 1 : *Object Interaction Graphs*

Mengembangkan graph interaksi objek untuk setiap operasi sistem dalam model operasi. Graph interaksi objek menunjukkan bagaimana secara fungsional antara objek-objek di satu sistem didistribusikan.

1. Mengenali secara relevan objek yang dilibatkan dalam perhitungan
2. Menentukan aturan setiap objek dalam perhitungan
3. Memutuskan pesan-pesan antar objek
4. Merekam bagaimana objek-objek yang dikenali berinteraksi pada graph interaksi objek dengan mengecek berikut ini :
   a. Konsistensi dengan model analisis. Cek setiap class dalam model objek sistem yang digambarkan di salah satu graph interaksi objek yang paling sedikit.

b. Verifikasi efek fungsional. Cek apakah pengaruh fungsional setiap graph interaksi objek memenuhi spesifikasi operasi sistem yang diberikan di model operasi.

Tahap 2 : *Visibility Graphs*

Untuk setiap class ditetapkan dengan kelayakan graph. Hal ini menunjukkan bagaimana sistem berorientasi objek dibentuk untuk mengenablekan komunikasi antar objek. Kelayakan graph dibangun sebagai berikut :

1. Semua graph interaksi objek diperiksa. Setiap pesan pada satu graph interaksi objek yang diberikan apakah menunjukkan kelayakan yang diperlukan dari class client ke objek server.
2. Memutuskan ragam petunjuk kelayakan yang diperlukan dengan menyimpan :
   a. Petunjuk *lifetime*
   b. Lelayakan target objek
   c. *Lilfetime* target objek
   d. Perubahan target objek
3. Menggambar graph kelayakan untuk setiap rancangan objek class, dengan mengecek berikut ini :
   a. Konsistensi dengan model analisis. Untuk setiap hubungan pada model objek sistem yang ada merupakan satu lintasan kelayakan untuk sekumpulan class pada kelayakan graph.
   b. *Mutual concistency*

      Cek apakah target objek eksklusif tidak ditunjukkan oleh lebih dari satu class dan apakah target yang dipakai bersama ditunjuk oleh lebih dari satu class.

Tahap 3 : *Class Description*

Deskripsi class merupakan spesifikasi dari dimana coding dimulai. Dengan menentukan status intenal dan interface ekstenal yang diperlukan oleh setiap class.

Mengekstraksi informasi dari model objek sistem, graph interaksi objek, dan kelayakan graph untuk membangun deskripsi class.
Setiap deskripsi class merekam berikut ini :

1. Metode dan parameter dari graph interaksi objek
2. Attribut data dari model objek sistem dan kamus data.
3. Attribut-attribut objek dari kelayakan graph untuk class
4. Informasi pewarisan (diisi setelah tahap berikutnya) dari graph pewarisan (*inheritance*).

   Cek berikut ini :

   a. *Methods and parameters*. Cek semua metode dari graph interaksi objek yang direkam.
   b. *Data attributes*. Cek semua metode dari model objek sistem yag direkam
   c. *Object Attributes*. Cek dimana semua petunjuk kelayakan direkam,
   d. *Inheritance*. Cek dimana semua superclass yang diwariskan direkam.

Tahap 4 : *Inheritance Graphs*

Tahapan ini untuk membangun struktur pewarisan (*inheritance*), yang berada di class-class yang dikenali keumumannya dan abstraksinya.

Mengenali *superclass* dan *subclass*. Membangun graph pewarisan. Dengan mencari berikut ini :

1. Generalisasi dan spesialiasi dalam model objek
2. Metode yang umum dalam graph interaksi objek dan deskripsi class
3. Kelayakan yang umum dalam graph kelayakan.

Tahap 5 : *UpDate Class Descriptions*

Mengperbaharui deskripsi class dengan informasi pewarisan yang baru. Cek berikut ini :

1. *System object model*. Cek dimana subtipe relasi di dipelihara/dilindungi.
2. *Object Interaction Graphs*. Cek semua class yang digambarkan dalam graph pewarisan. Dengan asumsi yang naif bahwa setiap setiap class dalam graph interaksi objek berada dalam graph pewarisan. Hal ini biasanya perkasus, tetapi juga diperlukan untuk bahan pertimbangan dalam struktur

class yang diorganisasikan kembali karena memperkenalkan abstraksi clas-class yang baru.

3. *Visibility graphs*. Cek semua class yang digambarkan dalam graph pewarisan. Abstraksi class bisa dikenali untuk struktur yang biasa diantara class-class dalam kelayakan graph.
4. *Class description*. Cek dimana deskripsi class yang di *Update* mengimplementasikan semua fungsionalitas dari salah satu persiapan dan mematuhi graph pewarisan.

Tahapan Implementasi

Tahapan metode *Fusion* yang terakhir adalah pemetaan perancangan ke dalam satu bentuk implementasi yang efektif.

Siklus hidup pada tahap implementasi terjadi dalam dua tahap. Pertama, ekspresi siklus hidup yang diterjemahkan ke dalam satu *state mechine* (*non deterministic*). Kemudian baru *state machine* itu sendiri diimplementasikan karena keluaran kejadian (*event*) dibangkitkan oleh operasi sistem.

State machine merupakan sekumpulan *state* (keadaan) dan diberi label transisi diantara *state*. *State machine* bisa ditampilkan sebagai sebagai tabel atau diagram gelembung. Untuk setiap tabel keadaan memberikan hasil keadaan baru setiap masukan keadaan yang mungkin. Diagram gelembung menunjukkan informasi yang sama, dengan *state* (keadaan) dilambangkan dengan gelembung dan transisi diberi label panah. Salah satu state ditandai sebagai keadaan awal mesin (*start state machine*).

Tahap 1 : *Coding*

Siklus Hidup Sistem

Untuk siklus hidup dengan tanpa *interleaving* .

1. Menterjemahkan ekspresi tetap siklus hidup ke dalam satu (*non deterministic state machine*)
2. Mengimplementasikan *state machine*.

Untuk siklus hidup dengan *interleaving*

1. Mengimplementasikan *interleave*- bebas subekspresi
2. menghubungkan hasil *state machine*.

Deskripsi Class

1. Menentukan spesifikasi dan *interface class-class*
   a. Deklarasi attribut. Attribut-attribut dari deskripsi class biasanya dibatasi dengan nama-nama class. Attribut-attribut yang memenuhi syarat ditambahkan untuk menentukan *mutability* dan pemamkaian bersama (*sharing*) jika sesuai.
   b. Deklarasi metode. Metode dari deskripsi class diimplementasikan dengan kode dalam class (yaitu anggota fungsi, rutin, prosedur)
   c. Pewarisan.
2. Mengimplementasikan metode isi.

Tahap 2 : Performasi

Tahap ini sebenarnya kurang tepat untuk tahapan proses implementasi. Mengingat berikut ini :

1. Performansi tidak bisa dihasilkan sebagai satu pemikiran di kemudian hari. Harus dipertimbangkan keseluruhan proses analisa, design dan implementasi.
2. Optimasi jarang mengeksekusi kode yang tidak berguna, maka profil sistem yang dibuat bisa dibuat dalam berbagai cara.

Tahap 3 : Review

1. Inspeksi

Teknik biaya efektif untuk mendeteksi cacat dalam perangkat lunak. Mengingat bahwa dalam perangkat lunak berorientasi objek merupakan analisa statik dari aliran pengendalian yang rumit dengan mengikuti :

- Bahasa orientasi objek menekankan hubungan pewarisan daripada hubungan pengendalian. Tidak ada pemetaan langsung antara kebutuhan fungsional dan fungsi-fungsi dengan level yang lebih tinggi.
- Ukuran struktur data lebih kecil, fungsi pengaksesan digunakan untuk mengimplementasikan class-class, dan class-class sebagai satu kesatuan yang utuh.

- Pembubaran metode spesifikasi dan implementasi diantara class.
- Ikatan secara dinamik membuat kesulitan untuk menentukan kode yang mana yang sebenarnya dieksekusi untuk satu permintaan metode.

Inspeksi diperlukan untuk tujuan penelusuran aliran pengendalian, yang difokuskan pada mendeteksi cacat dalam sistem berorientasi objek. Mengkonfirmasikan dimana *subclass* mengimplementasikan satu metode khusus memenuhi untuk spesifikasi metode.

2. Testing

Teknik yang melengkapi untuk memeriksa cacat yang nampak dalam perangkat lunak. Uji kasus untuk class seharusnya meliputi berikut ini :

- Pengecekan status observasi dan manipulasi
- Menggunakan properti yang bersifat aljabar seperti asosiasi dan mengenali pemeliharaan untuk keanggotaan permintaan fungsi.
- Mengecek apakah pembongkar dalam C++ sudah konsisten dengan kumpulan pembangun.
- Mengecek pemakaian inisialisasi.
- Mengecek apakah pemilihan C++ telah digunakan dengan cara yang aman.
- Mencoba untuk memicu pengecualian penanganan kemampuan melalui masukan batasan nilai ekstrem.

KESIMPULAN

a. Proses pengembangan perangkat lunak diperlukan untuk mengurangi sejumlah kesalahan (error) yang tampak di akhir produk perangkat lunak

b. Metode Fusion mengintegrasikan beberapa aspek terbaik dari beberapa metode pengembangan berorientasi objek yang ada seperti Metode FORMAL, OMT, CRC dan BOOCH.

c. Fusion merupakan metode generasi kedua yang dibangun berdasarkan kesuksesan bagian-bagian metode berorientasi objek awal dan ditujukan untuk mengurangi kelemahan-kelemahan yang ada. Dengan tiga tahapan yaitu :

- Analisa, yang melingkupi objek dan class dalam sistem, menggambarkan hubungan diantaranya, dan menetapkan operasi-operasi dimana sistem bisa melakukannya.
- Desain, yang memutuskan bagaimana untuk menggambarkan operasi sistem dengan interaksi antar objek dan bagaimana objek-objek tersebut bisa mengakses satu dengan yang lainnya.
- Implementasi, yang menterjemahkan perancangan kedalam bentuk bahasa pemrograman.

DAFTAR PUSTAKA

1. Anderung, Letzte, 1996, "*CASE Methodology*", http://www.unix-ag.uni-kl.de/~lippold/case-methods.html#RUMBAUGH
2. Cetus Team, 2001, "*Architecture and Design : Unified Modeling Language (UML)*", http://www.cetus-links.org/oo_uml.html
3. Coleman, D., et all., 1994, *Object Oriented Development : The Fusion Method* , Prentice-Hall, Englewood Cliffs, New Jersey 07632
4. Hariyanto, H., 2004, *Rekayasa Sistem Berorientasi Objek* , Informatika Bandung
5. R.J.Abbott, "Program Design by Informal English Descriptions," CACM, Vol. 26, #11, Nov. 1983, pp. 882-894.
6. Rumbaugh**,** 1991, **"***Object-Oriented Modeling and Design for Database Applications*", Prentice-hall.
7. Yordon, Edward, 1994, "*Object Oriented System Design : An Integreted Approach*", Prentice-Hall, Inc