The Solid Snake

Catatan Harian Seorang Developer

Merancang Sistem Dengan UML: Mulai Dari Mana?

19 FEBRUARI 2013 55 KOMENTAR (HTTPS://THESOLIDSNAKE.WORDPRESS.COM/2013/02/19/MERANCANG-SISTEM-DENGAN-UML-MULAI-DARI-MANA/#COMMENTS)

i
36 Votes

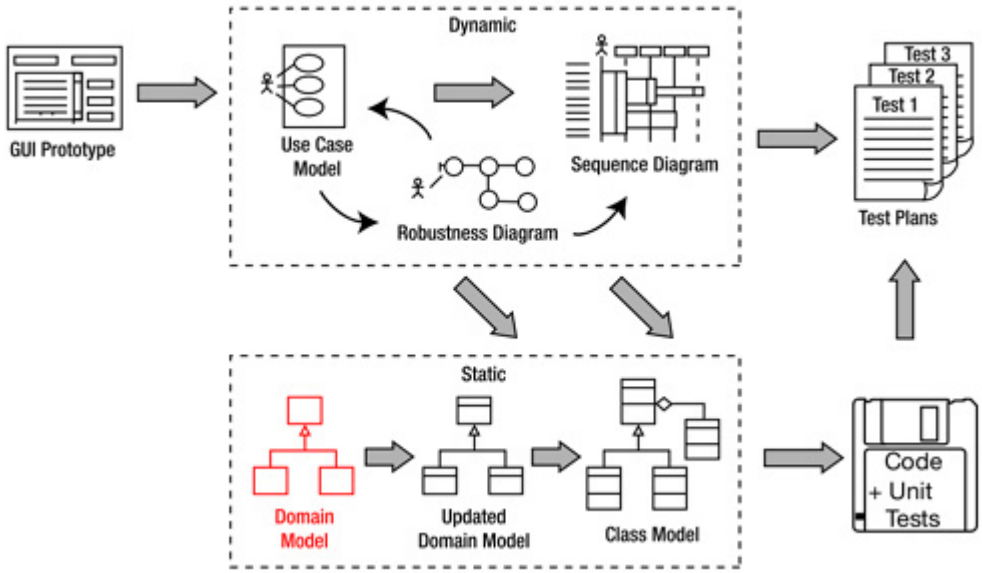
Salah satu pertanyaan yang paling sering saya temukan dari para mahasiswa adalah apa saja step-step yang harus ditempuh dalam merancang sistem berbasis UML? Saya biasanya akan memperbaiki pertanyaan ini karena UML pada dasarnya hanya kumpulan diagram atau teknik visualisasi; UML tidak mendikte langkah-langkah pengembangan sistem! Lalu apa yang harus dilakukan dalam merancang sebuah sistem?

Saya yakin para mahasiswa telah memperoleh mata kuliah pengembangan sistem informasi sebelumnya, tapi kenapa mereka bertanya demikian disaat mereka harus merancang sistem yang nyata (misalnya untuk skripsi)? Salah satu penyebabnya adalah analisa & perancangan terlalu teoritis sehingga tidak mendukung implementasi/pembuatan kode program.

Dari beberapa buku analisa & perancangan yang saya baca, buku **Use Case Driven Object Modeling with UML: Theory and Practice** adalah salah satu buku yang spesial dan sangat membantu. Bukan hanya dilengkapi dengan humor, tetapi buku ini memberikan metode analisa & perancangan yang sangant berguna saat diterapkan dalam pembuatan kode program! Buku yang memakai metode **ICONIX Process** ini ditulis oleh analyst yang memiliki latar belakang programmer.

Saya tahu bahwa sangat sulit merangkum sebuah buku 438 halaman dalam sebuah artikel blog, tapi saya ingin menunjukkan bahwa diagram UML hasil analisa & perancangan bukanlah sekedar basa basi yang dihasilkan analyst. Dengan metode yang salah, analyst kerap terlihat tidak berguna di mata developer. Metode yang salah juga menyebabkan mahasiswa lebih senang membuat kode program terlebih dahulu, baru melakukan reverse engineering untuk menghasilkan diagram UML. Dengan kata lain, sistem dibuat tanpa analisas & perancangan, sementara diagram UML hanya produk sampingan yang menambah ketebalan skripsi tanpa fungsi yang sangat berarti.

Gambar berikut ini memperlihat proses analisa & perancangan sistem informasi dengan **ICONIX process**:



(https://thesolidsnake.files.wordpress.com/2013/02/fig1.jpg)

ICONIX Process

Proses analisa & perancangan sistem yang saya rangkum dari buku **Use Case Driven Object Modeling With UML: Theory and Practice** adalah sebagai berikut ini:

1. Membuat Functional Requirement

Gunakan Microsoft Word untuk menuliskan **functional requirement** (apa yang dapat dilakukan oleh sistem?). Tahap ini melibatkan business analyst, pelanggan, end user, dan project stakeholders lainnya. **Functional requirement** bersifat tidak terstruktur dan tidak dapat dipakai dalam perancangan secara langsung.

Berikut ini adalah contoh potongan dari **function requirement** untuk sebuah sistem bengkel motor yang sederhana:

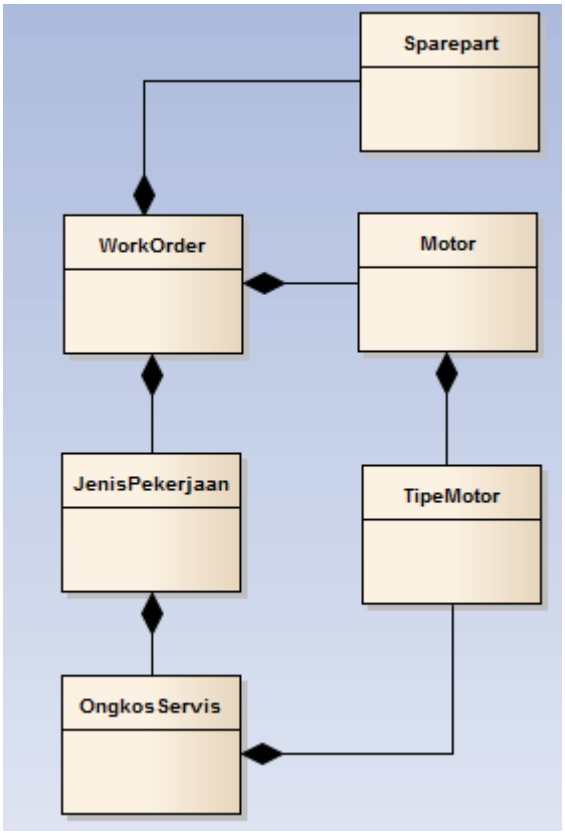
```
Sistem harus dapat memproses work order untuk motor mulai dari a
oleh seorang mekanik, selesai di-servis, dan proses pembayaran.
Sebuah work order memiliki informasi jenis pekerjaan dan sparepa
Harga ongkos servis berdasarkan jenis pekerjaan dan tipe motor.
```

2. Membuat Domain Model (sederhana)

Salah satu fungsi **domain model** adalah menyamakan istilah yang akan pakai diproses selanjutnya. Misalnya, apakah saya akan memakai istilah '*work order*' atau '*pekerjaan servis*'? Apa saya akan memakai sebutan '*sparepart*' atau '*suku cadang*'? Walau terlihat sepele, perbedaan istilah seringkali menimbulkan salah paham dalam komunikasi tim.

Pada tahap ini, **domain model** adalah *class diagram* yang hanya memakai relasi pewarisan (**is-a**/adalah sebuah) dan agregasi (**has-a**/memiliki sebuah). *Class diagram* ini belum memiliki atribut dan operasi. Nantinya, di proses selanjutnya, **domain model** akan diperbaiki dan dikembangkan menjadi lebih detail.

Gambar berikut ini memperlihatkan contoh **domain model** untuk **functional requirement** di langkah 1:



(https://thesolidsnake.files.wordpress.com/2013/02/gambar1.png)

Domain Model Versi Awal

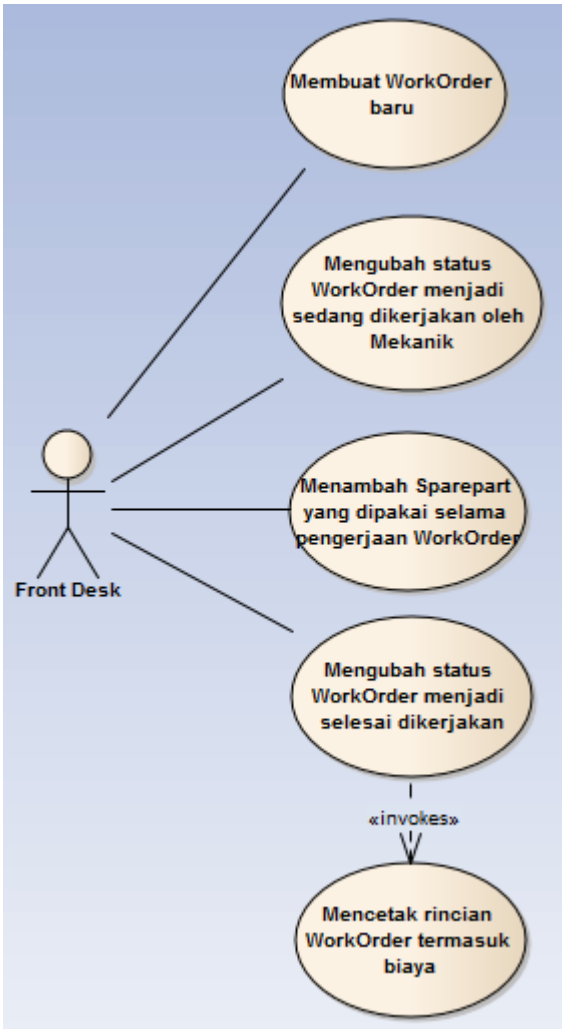
3. Membuat Use Case

Use case mendefinisikan **behavioral requirements** berdasarkan **functional requirement** (dan sumber lainnya). Berbeda dengan anjuran dari buku analisis sisfo lain, buku ini menyarankan untuk membuat use case dengan maksimal 2 paragraf! Tidak perlu mengikuti template yang detail! Sebuah use case yang panjang & detail malah akan memperlambat kita. Tim yang membuat use case bisa jadi akhirnya hanya mengisi form yang kosong tanpa banyak berpikir panjang (misalnya sekedar *copy paste)* sehingga proses membuat use case hanya sekedar ritual tanpa analisa mendalam.

Kalimat yang dipakai dalam use case harus berupa **kalimat aktif**, misalnya “pengguna men-klik tombol Login”. **Kalimat pasif** seperti “Sistem menyediakan tombol Login” adalah ciri-ciri **functional requirement** dan bukan bagian dari use case.

Use case harus mengandung nama di **domain model**. Dengan demikian, saya bisa menghubungkan class-class yang akan dirancang dengan use case. Setiap halaman/layar yang direferensikan di dalam use case sebaiknya diberi nama yang konsisten, misalnya *Halaman Tambah Sparepart* atau *Screen TambahSparepart*.

Sebuah use case hampir mirip seperti dokumentasi sistem. Kita perlu menspesifikasikan seperti apa cara pakai sistem (termasuk respon sistem) sebelum sebuah sistem dibuat. Berikut ini adalah contoh use case diagram berdasarkan **functional requirement** di langkah 1 dan memakai **domain model** dari langkah 2:



(https://thesolidsnake.files.wordpress.com/2013/02/gambar2.png)

Use Case Diagram

Sebuah use case selain memiliki **sunny-day scenario**, sebaiknya juga memiliki **rainy-day scenario** (apa yang akan terjadi bila sesuatu salah?) atau alternatif. Sebagai contoh, berikut ini contoh use case scenario untuk use case diagram di atas:

Membuat WorkOrder baru

Basic Scenario

Front Desk memilih Motor di Screen ListMotor dan men-klik tombo
Sistem akan membuat sebuah WorkOrder baru dengan status sedang

Alternate Scenario

Motor belum terdaftar - Front Desk terlebih dahulu menambah Mot
men-klik tombol untuk menambah Motor baru sebelum mengerjaka
yang ada di Basic Scenario.

Motor sudah memiliki WorkOrder yang sedang antri - Sistem akan

[Catatan: pada saat membuat use case ini, terlihat bahwa dibutuh
Agar ringkas, use case tersebut akan diabaikan.]

Mengubah Status WorkOrder menjadi sedang dikerjakan oleh Mekanik

Basic Scenario

Front Desk memilih sebuah WorkOrder di Screen ListWorkOrder dar
menandakan bahwa Workorder tersebut sedang dikerjakan.
Sistem akan menampilkan Screen PengerjaanWorkOrder. Front Desk
yang mengerjakan WorkOrder dan men-klik tombol untuk menyimpan
Sistem akan mengubah status WorkOrder menjadi sedang dikerjakar
serta mencatat tanggal & jam mulai dikerjakan.

Alternate Scenario

Status WorkOrder yang dipilih bukan sedang antri - Sistem akan

Menambah Sparepart yang dipakai selama pengerjaan WorkOrder

Basic Scenario

Front Desk memilih sebuah WorkOrder di Screen ListWorkOrder dar
menambah Sparepart di WorkOrder. Sistem akan menampilkan Screer
berisi daftar Sparepart untuk WorkOrder yang dipilih. Disini Fr
data ItemSparepart dengan memilih Sparepart, memasukkan jumlah
tombol Tambah ItemSparepart. Sistem akan memperbaharui daftar I
Front Desk men-klik tombol Simpan untuk selesai. Sistem akan me
pada WorkOrder terpilih.

Alternate Scenario

Terdapat lebih dari satu jenis Sparepart yang perlu ditambahkan
menambah data ItemSparepart. Setelah semua ItemSparepart sele
men-klik tombol Simpan di Screen TambahSparepart. Sistem akar

Status WorkOrder yang dipilih bukan sedang dikerjakan - Sistem
kesalahan.

[Catatan: pada saat membuat use case ini, terlihat bahwa ada yar
yaitu ItemSparepart. Segera update domain model! Terlihat juga
metode untuk menghapus Sparepart dan meng-edit jumlah Sparepar
Agar ringkas, use case tersebut akan diabaikan.]

Mengubah status WorkOrder menjadi selesai dikerjakan

Basic Scenario

Front Desk memilih sebuah WorkOrder di Screen ListWorkOrder, la
status WorkOrder menjadi selesai dikerjakan. Sistem akan menamp
Bila kasir menkonfirmasi, Sistem akan mengubah status WorkOrder
dikerjakan dan mencatat jam selesai dikerjakan. Front Desk kemu
use case "Mencetak rincian WorkOrder termasuk biaya".

Alternate Scenario
Status WorkOrder bukan sedang dikerjakan - Sistem akan menampil

Mencetak rincian WorkOrder termasuk biaya
Basic Scenario
Front Desk memilih tombol untuk mencetak. Sistem kemudian mence
ke printer.
Detail WorkOrder yang dicetak meliputi tanggal, plat nomor motc
jam selesai dikerjakan, nama mekanik yang mengerjakan, rincian
(jumlah & harga eceran tertinggi Sparepart), ongkos servis dan
Alternate Scenario
Status WorkOrder bukan selesai dikerjakan - Sistem akan menampi
[Catatan: use case ini dipisahkan dari use case "Mengubah status
selesai dikerjakan" karena dianggap nanti akan ada use case lai
mencetak rincian WorkOrder tetapi tidak ditampilkan disini.]

4. Requirements Review

Pada saat melakukan analisa dalam membuat use case, saya menemukan hal yang masih kurang. Misalnya, saya perlu menambahkan class `ItemSparepart` pada **domain model**. Selain itu, pada beberapa situasi, saya bahkan bisa menemukan ada use case yang masih kurang, misalnya use case "*Tambah Motor baru*".

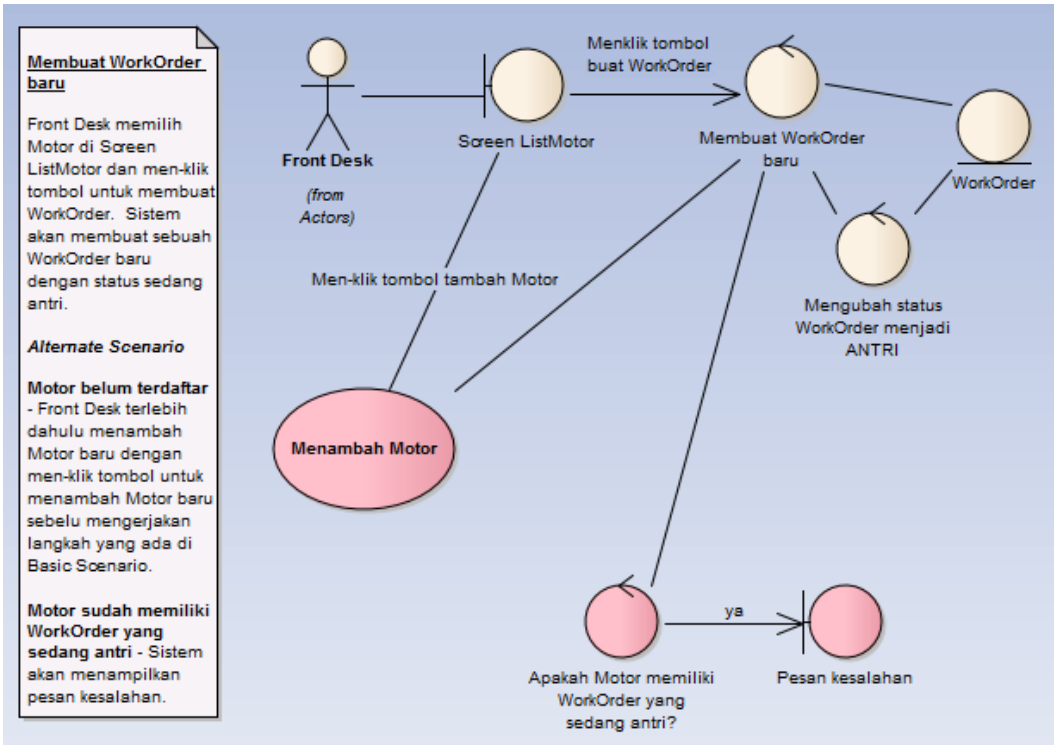
Pada langkah ini, saya kembali memastikan bahwa use case & **domain model** telah dibuat dengan baik. Pelanggan juga perlu dilibatkan untuk memastikan bahwa use case (**behavioral requirement**) & **functional requirement** sesuai dengan yang diharapkan. Ingatlah selalu bahwa bagian terpenting dari sebuah sistem bukanlah seberapa keren *design pattern* yang diterapkan di class diagram, tetapi sejauh mana sistem tersebut memberikan profit bagi penggunanya (memenuhi requirements).

5. Melakukan Robustness Analysis

Analisis adalah memikirkan "apa" (what), sementara perancangan adalah memikirkan "bagaimana" (how). Salah satu alasan mahasiswa sering mengabaikan UML dan langsung terjun ke coding adalah celah yang cukup jauh antara analisis dan perancangan sehingga mereka memilih merancang secara eksperimental dengan langsung coding. Umumnya mereka berakhir dalam jebakan siklus perubahan "coding" terus menerus (guna memperbaiki rancangan). Padahal, perubahan "coding" adalah sesuatu yang sangat memakan waktu dan upaya bila dibandingkan dengan mengubah diagram UML.

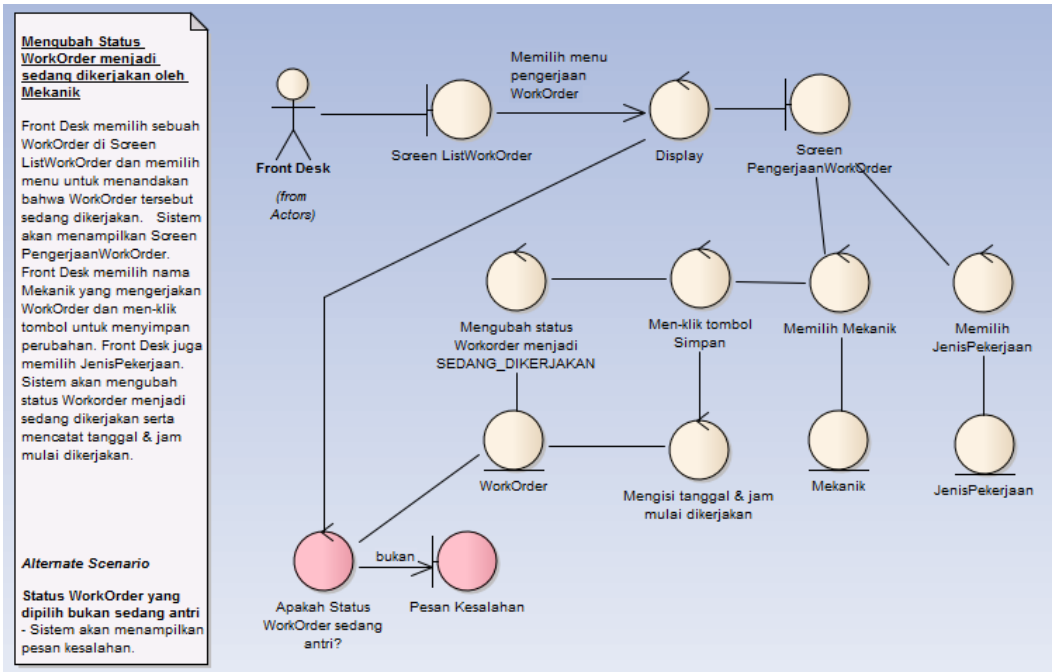
Robustness analysis dipakai untuk menjembatani analisis dan perancangan. **Robustness analysis** harus diterapkan pada setiap use case yang ada. Pada Enterprise Architect, **robustness analysis** dapat digambarkan dengan menggunakan **Analysis Diagrams (**terdapat di kategori **Extended**).

Berikut ini adalah contoh hasil **robustness analysis** untuk use case yang ada:



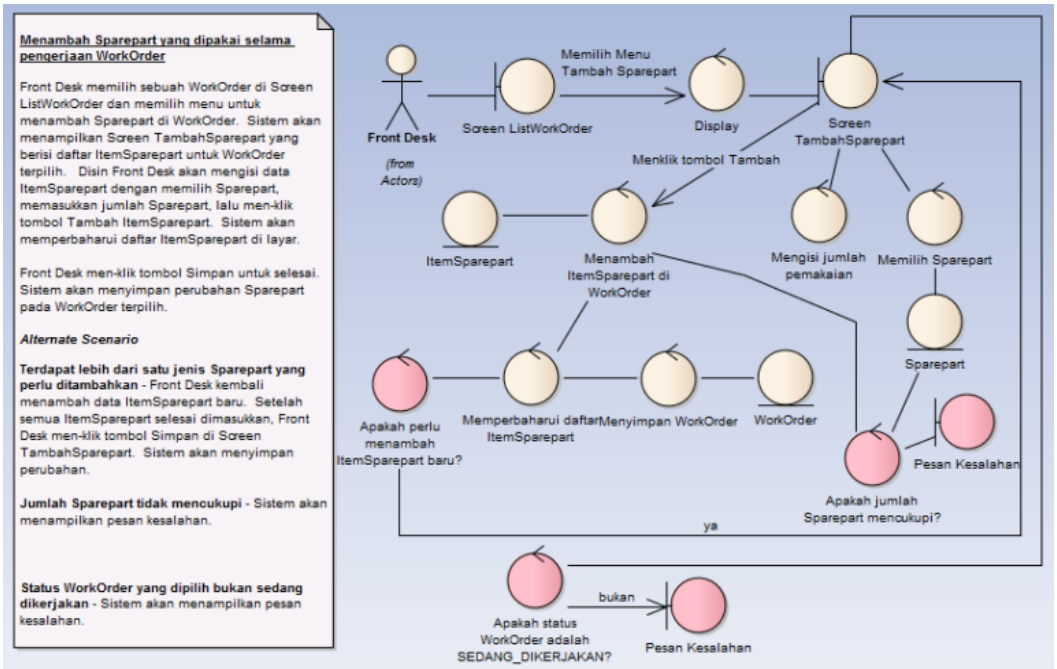
(https://thesolidsnake.files.wordpress.com/2013/02/gambar3.png)

Analysis Diagram Untuk Use Case Membuat WorkOrder Baru



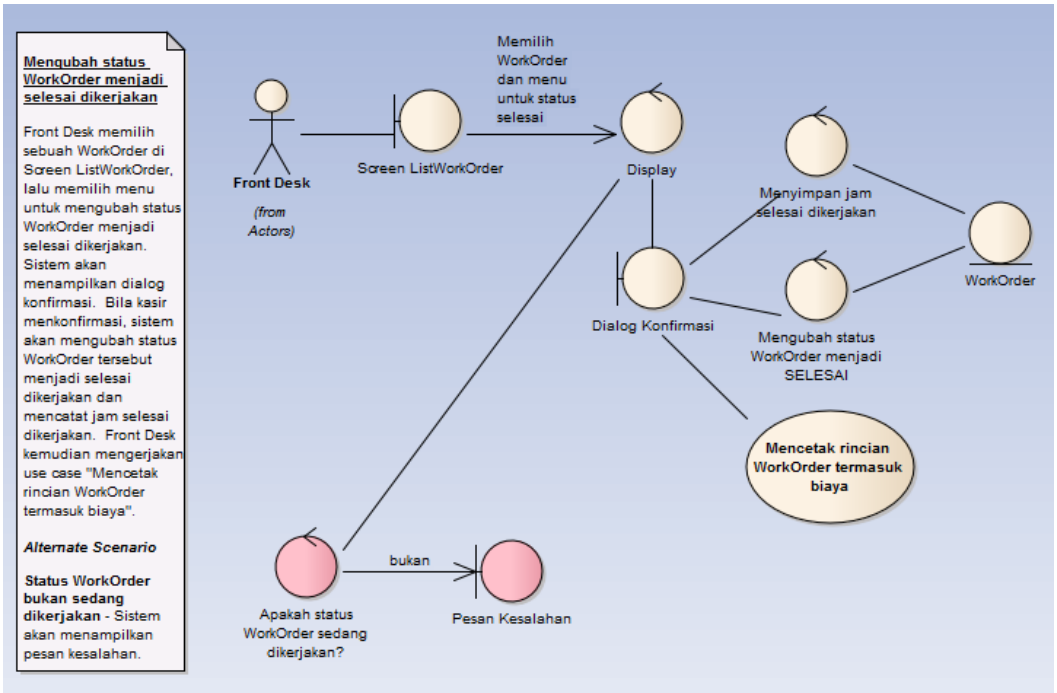
(https://thesolidsnake.files.wordpress.com/2013/02/gambar4.png)

Analysis Diagram Untuk Use Case Mengubah Status WorkOrder Menjadi Sedang DIkerjakan Oleh Mekanik



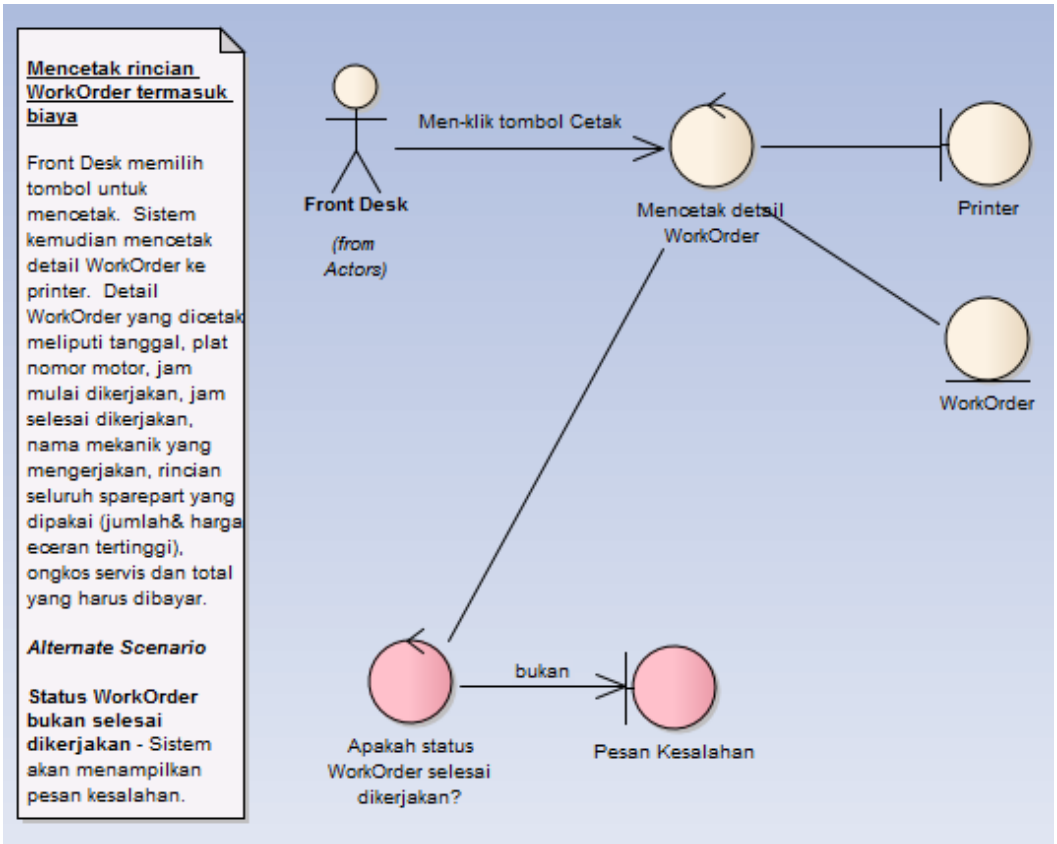
(https://thesolidsnake.files.wordpress.com/2013/02/gambar5.png)

Menambah Sparepart yang dipakai selama pengerjaan WorkOrder



(https://thesolidsnake.files.wordpress.com/2013/02/gambar6.png)

Analysis Diagram Untuk Use Case Mengubah Status WorkOrder Menjadi Selesai Dikerjakan



(https://thesolidsnake.files.wordpress.com/2013/02/gambar7.png)

Analysis Diagram Untuk Use Case Mencetak Rincian WorkOrder Termasuk Biaya

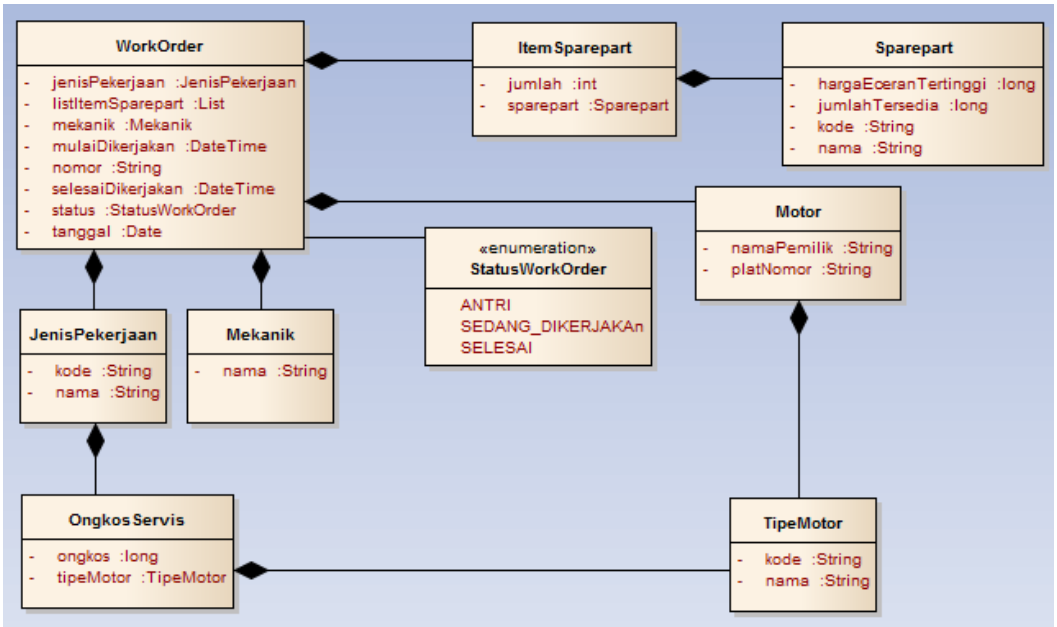
Semakin detail **robustness analysis**, maka semakin banyak hal yang kurang dari use case dan **domain model** yang akan ditemukan. Yup! Pada awalnya saya ragu kenapa saya harus menambah *control* "mengisi jumlah", "mengisi nama", dsb untuk setiap field yang ada. Tapi saya cukup terkejut saat menemukan dari hal sepele tersebut, saya menemukan beberapa alternate scenario yang kurang.

Sebagai contoh, saya menemukan bahwa saya lupa menambahkan alternate scenario "jumlah Sparepart tidak mencukupi" saat melakukan analisa robustness pada use case "Menambah Sparepart yang dipakai selama pengerjaan WorkOrder". Semakin cepat saya menyadari ada yang kurang, semakin baik! Idealnya adalah sebelum coding dilakukan. **Robustness analysis** adalah salah satu senjata yang ampuh untuk itu.

Saya juga menemukan bahwa pada use case "Mengubah Status WorkOrder menjadi sedang dikerjakan oleh Mekanik", saya lupa menuliskan bahwa Front Desk officer juga perlu memilih `JenisPekerjaan`. Saya perlu segera mengubah teks use case tersebut.

Selain itu, terkadang saya juga dapat menemukan ada class yang kurang pada domain model. Misalnya, dari hasil **robutsness analysis**, terlihat bahwa saya perlu menambahkan class `Mekanik` di domain model.

Pada tahap ini, saya juga perlu mengisi domain model dengan atribut, seperti yang terlihat pada gambar berikut ini:



(https://thesolidsnake.files.wordpress.com/2013/02/gambar8.png)

Domain Model Yang Telah Memiliki Atribut

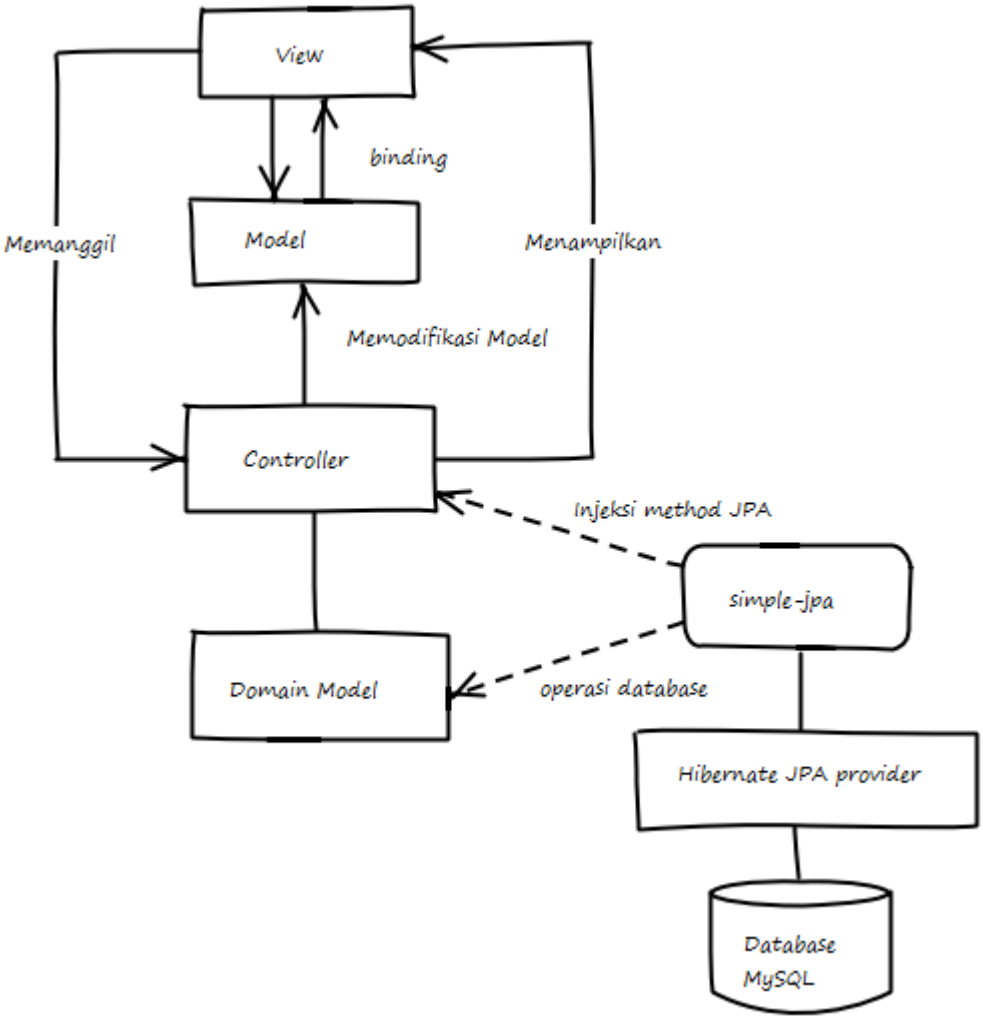
6. Preliminary Design Review

Kembali lagi seluruh tim melakukan review dan memastikan bahwa semua yang telah dibuat sesuai dengan requirement. Ini adalah langkah terakhir dimana pelanggan (stackholder) terlibat! Hal ini karena langkah berikutnya melibatkan proses techincal. Akan berbahaya bila membiarkan pelanggan yang non-technical atau *semi*-technical mengambil keputusan untuk hal-hal yang bersifat teknis (misalnya framework atau database yang dipakai). Walaupun demikian, pelanggan boleh memberikan komentar mengenai tampilan.

Setelah langkah ini, tidak ada lagi perubahan requirement. Lalu bagaimana bila pelanggan ingin menambah requirement? Buat sebuah rilis atau milestone baru dengan kembali lagi ke langkah pertama di atas.

7. Menentukan Technical Architecture

Tentukan framework apa yang akan dipakai. Sebagai contoh, saya akan membuat sebuah aplikasi desktop dengan Griffon. Pola arsitektur yang dipakai menyerupai Model View ViewModel (MVVM) dimana terdapat perbedaan antara **domain model** dan **view model**. Saya juga mengasumsikan penggunaan sebuah plugin fiktif yang dirujuk sebagai *simple-jpa*. Gambar berikut ini memperlihatkan contoh arsitektur yang dipakai:



(https://thesolidsnake.files.wordpress.com/2013/02/gambar9.png)

Gambaran Technical Architecture

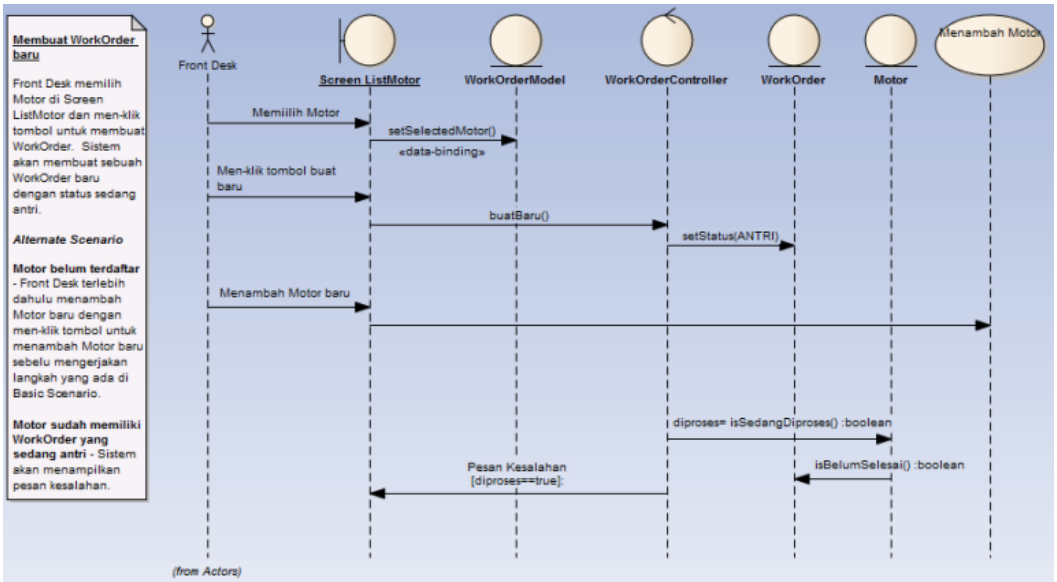
8. Membuat Sequence Diagram

Object oriented pada dasarnya adalah menggabungkan antara data dan operasi ke dalam sebuah entitas. Saat ini, **domain model** baru berisi data. Oleh sebab itu, dibutuhkan sebuah upaya untuk menemukan operasi untuk **domain model**. Caranya adalah dengan memakai **sequence diagram**.

Saat membuat **sequence diagram**, sertakan juga elemen dalam arsitektur teknis/framework. Misalnya penggunaan MVC akan menyebabkan ada class baru seperti *controller*. Yup! Penggunaan MVC akan membuat operasi tersebar ke controller. Hal ini sering dikritik karena bukan pendekatan OOP melainkan kembali ke zaman prosedural. Baca buku **Object Design: Roles, Responsibilities, and Collaborations** untuk pendekatan yang OOP, akan tetapi jangan lupa kalau kita dibatasi oleh framework yang dipakai (ehem, seharusnya bukan framework yang membatasi kita, melainkan kita yang tegas dalam memilih framework).

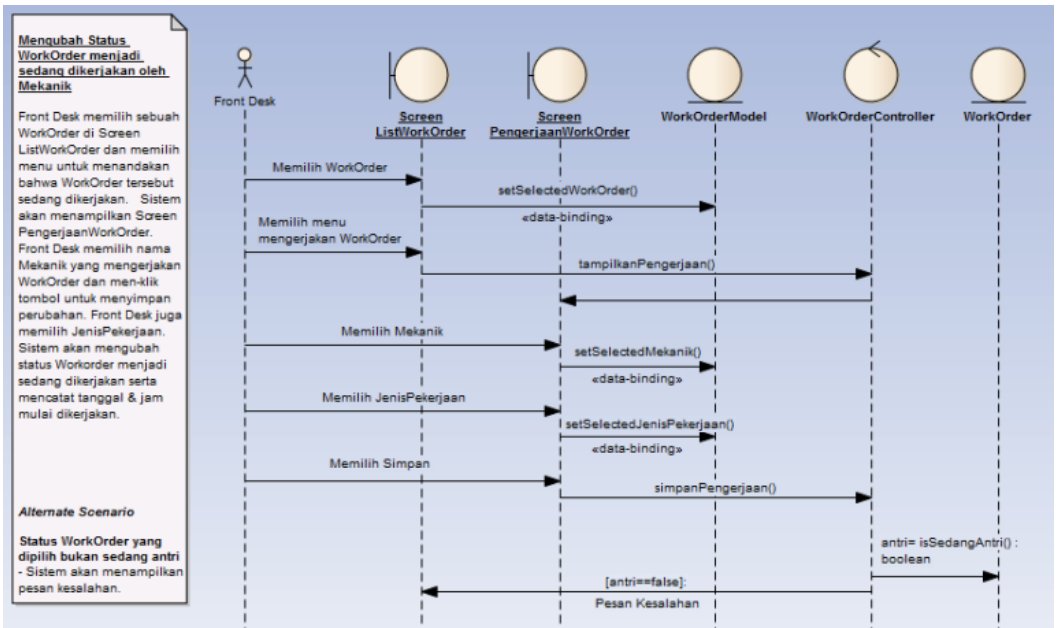
Selama membuat **sequence diagram**, ingatlah selalu bahwa tujuannya adalah menemukan operasi (**behavior**) untuk setiap class yang ada, bukan menunjukkan step-by-step operasi secara detail! Untuk menjaga agar tidak tersesat menjadi membuat *flow-chart* yang detail, jangan memikirkan **focus of control** (matikan saja fitur tersebut!).

Sequence diagram dibuat untuk setiap use case yang ada, berdasarkan hasil **robustness anaylsis**. Gambar berikut ini memperlihat contoh **sequence diagram** yang dihasilkan (agar sederhana, operasi penyimpanan data oleh *simple-jpa* tidak ditampilkan):



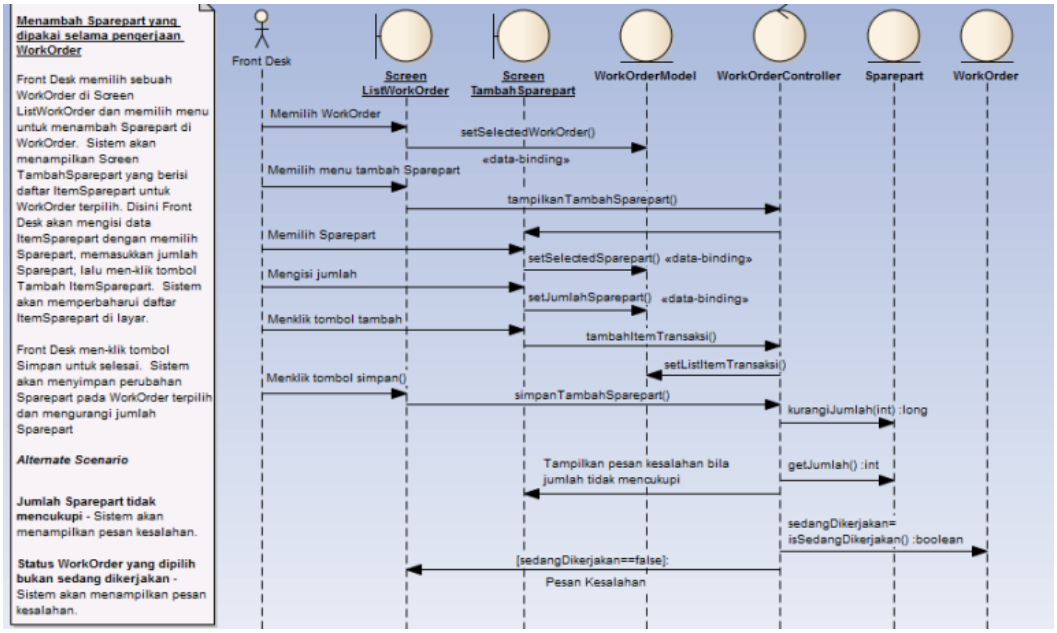
(https://thesolidsnake.files.wordpress.com/2013/02/gambar10.png)

Sequence Diagram Untuk Membuat WorkOrder Baru



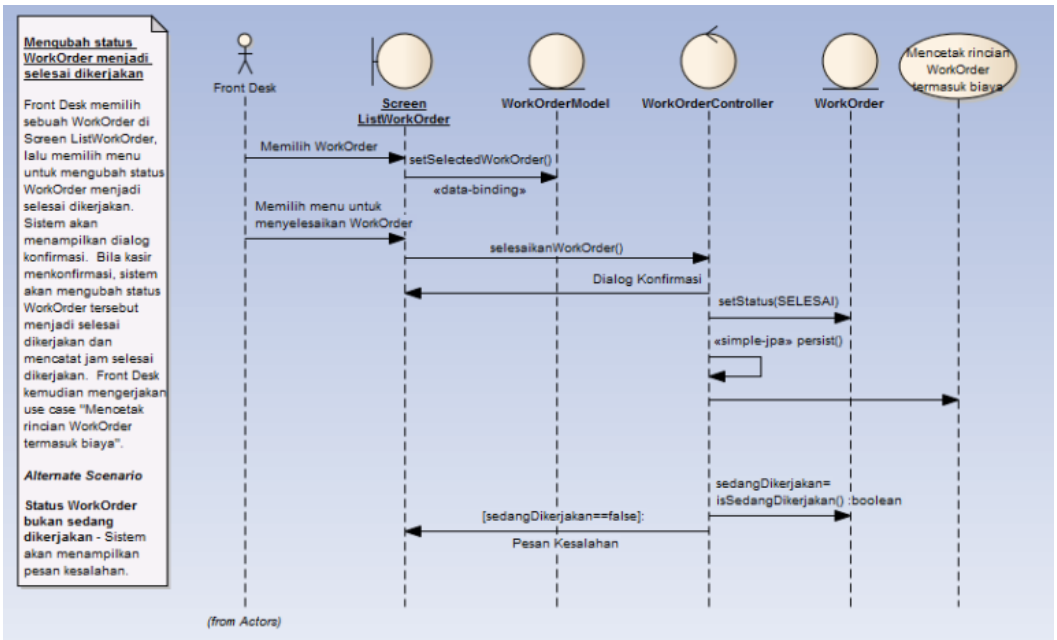
(https://thesolidsnake.files.wordpress.com/2013/02/gambar11.png)

Sequence Diagram Untuk Mengubah Status WorkOrder Menjadi Sedang Dikerjakan Oleh Mekanik



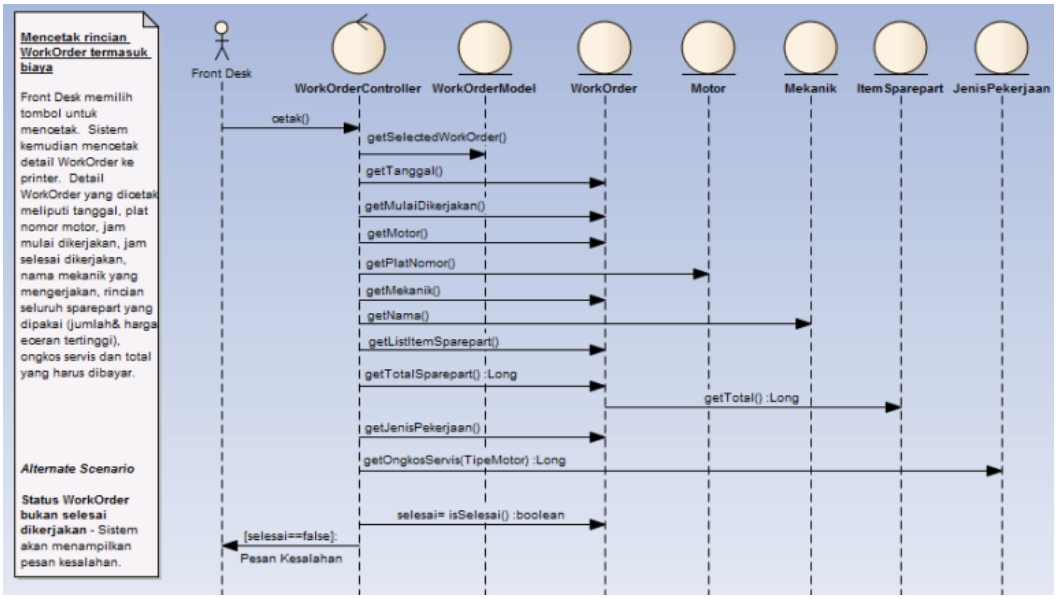
(https://thesolidsnake.files.wordpress.com/2013/02/gambar12.png)

Sequence Diagram Untuk Menambah Sparepart Yang Dipakai Selama Pengerjaan WorkOrder



(https://thesolidsnake.files.wordpress.com/2013/02/gambar13.png)

Sequence Diagram Untuk Mengubah Status WorkOrderMenjadi Selesai Dikerjakan

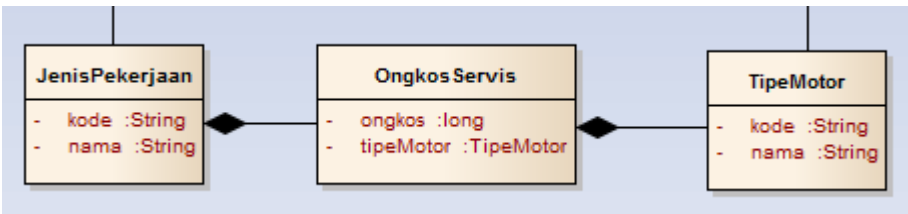


(https://thesolidsnake.files.wordpress.com/2013/02/gambar14.png)

Sequence Diagram Untuk Mencetak Rincian WorkOrder Termasuk Biaya

Proses analisa yang berulang kali lagi-lagi membantu saya menemukan kekurangan. Saya selama ini ternyata lupa bahwa pada use case "Menambah Sparepart yang dipakai selaman pengerjaan WorkOrder", sistem harus mengurangi jumlah stok `Sparepart` bila pengguna men-klik tombol simpan. Oleh sebab itu, saya segera memperbaharui teks use case.

Selain itu, saya juga menemukan sebuah kesalahan yang saya buat dari awal dan tidak terdeteksi hingga sekarang, yang berhubungan dengan class `OngkosServis`. Gambar berikut ini memperlihatkan perancangan awal class tersebut:



(https://thesolidsnake.files.wordpress.com/2013/02/gambarx0.png)

Kesalahan Rancangan Domain Model Akibat Berfokus Pada Penyimpanan Data

Bila membuat ERD atau design tabel, ini adalah sesuatu yang dapat diterima (ongkos disimpan dalam tabel yang mewakili hubungan one-to-many dari `JenisPekerjaan` ke `TipeMotor`). Tetapi, bila diterapkan ke dalam domain model, maka akan terjadi kejanggalan saat saya memakai domain model tersebut di sequence diagram. Untuk memperoleh ongkos servis, apa saya harus membuat instance objek `OngkosServis` baru? Bila diterapkan ke coding, maka ini berarti untuk memperoleh ongkos servis, saya harus selalu melakukan query JPQL yang kira-kira terlihat seperti berikut ini:

```
TipeMotor tipeMotor = model.selectedTipeMotor
JenisPekerjaan jenisPekerjaan = model.selectedJenisPekerjaan
Query query =
  em.createQuery("SELECT o.harga FROM OngkosServis o WHERE " +
      "o.tipeMotor = :tipeMotor AND o.jenisPekerjaan = :jenisPek
query.setParameter("tipeMotor", tipeMotor)
query.setParameter("jenisPekerjaan", jenisPekerjaan)
Long ongkos = query.getSingleResult()
```

Terlihat ada yang tidak beres! Bukankah OOP harus intuitive & bisa dipakai dengan mudah? Kenapa tiba2 harus melakukan query secara eksplisit untuk memperoleh sebuah ongkos servis? Selain itu, class `OngkosServis` tidak pernah dipakai secara langsung di kode program, hanya dipakai di query saja!

Oleh sebab itu, saya memperbaikinya dengan membuang class `OngkosServis`, dan menambahkan atribut `ongkosServis` di class `JenisPekerjaan` dengan tipe data berupa `Map<TipeMotor,Long>`. Dengan demikian, saya bisa memakainya seperti berikut ini:

```
TipeMotor tipeMotor = model.selectedTipeMotor
JenisPekerjaan jenisPekerjaan = model.selectedJenisPekerjaan
Long ongkos = jenisPekerjaan.getOngkosServis(tipeMotor)
```

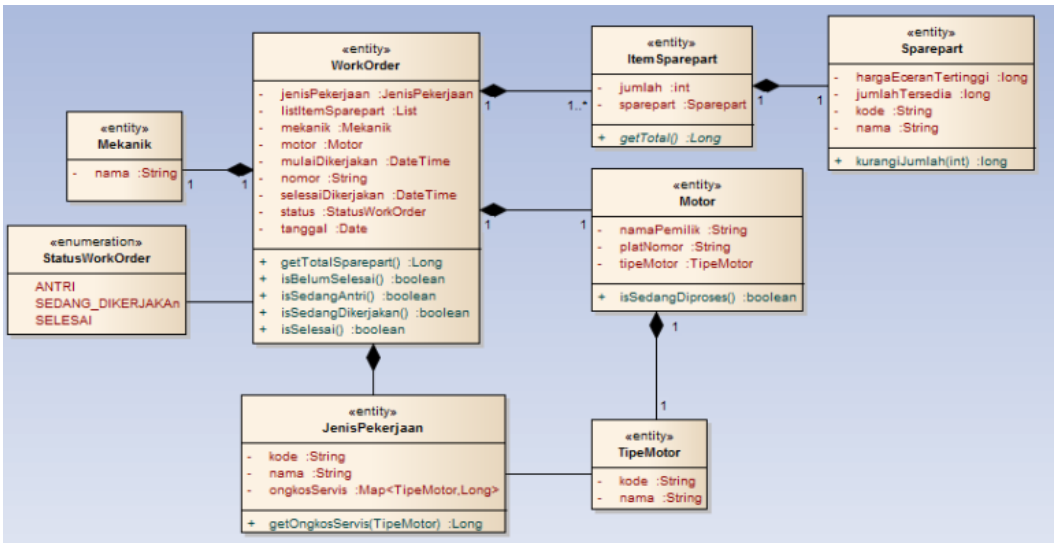
Cara di-atas jauh lebih intuitive dan mudah dipahami. Query database tetap terjadi, tetapi kali ini secara implisit (secara otomatis) oleh Hibernate tanpa campur tangan developer.

Ini adalah alasan kenapa saya selalu memberikan pesan pada mahasiswa agar tidak memikirkan proses penyimpanan data saat membuat **domain model**. Jangan menyamakan **domain model** dan ERD. Pada saat merancang **domain model**, pikirkan bagaimana class-class yang ada akan saling berinteraksi dan dipakai oleh developer. Bahkan bila tidak memakai ORM seperti Hibernate, tetap jangan memikirkan bagaimana penyimpanan data di **domain model**, melainkan buat ERD terpisah untuk dipakai oleh persistence layer (DAO).

9. Critical Design Review

Kembali melakukan review untuk memastikan bahwa tidak ada yang kurang pada **sequence diagram**. Pastikan bahwa setiap class yang ada telah memiliki atribut dan operasi yang didefinisikan secara lengkap (memiliki nama, tipe data, parameter, dsb).

Berikut ini adalah contoh hasil domain model yang telah dilengkapi dengan operasi dan multiplicity:

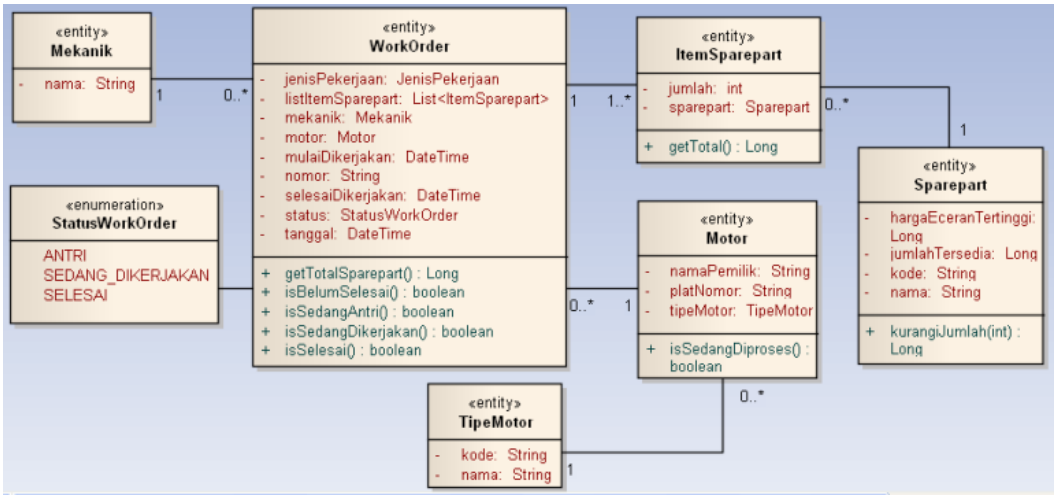


(https://thesolidsnake.files.wordpress.com/2013/02/gambar15.png)

Domain Model Yang Telah Lengkap

Getter dan setter tidak perlu ditampilkan karena hanya akan membuat class diagram terlihat ‘penuh’.

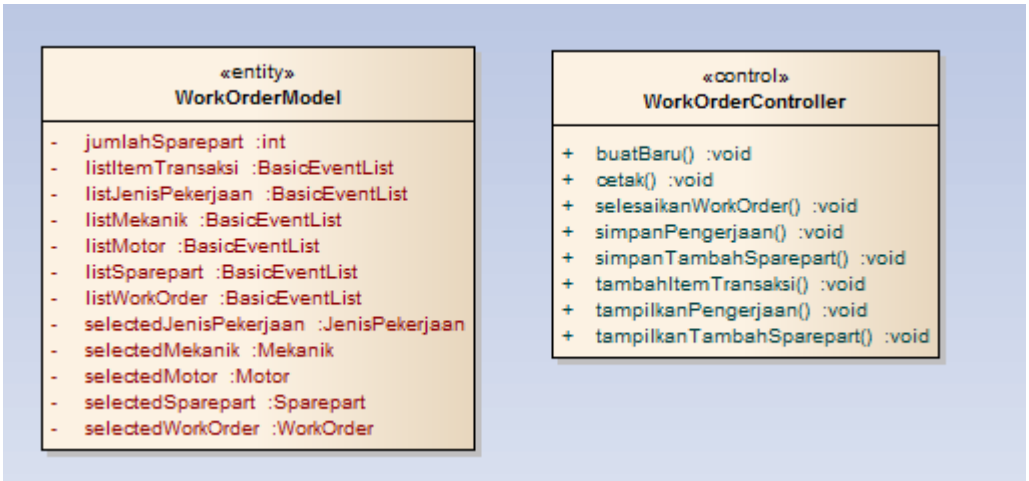
Versi *bidirectional*-nya akan terlihat seperti pada gambar berikut ini:



(https://thesolidsnake.files.wordpress.com/2013/02/gambar.png)

Domain Model Yang Telah Lengkap (Versi Bidirectional)

Selain **domain model**, saya juga menemukan terdapat class-class lain yang dihasilkan yang berkaitan dengan penggunaan framework, yang terlihat seperti pada gambar berikut ini:



(https://thesolidsnake.files.wordpress.com/2013/02/gambar16.png)

Class pembantu untuk framework

10. Coding

Disini developer berperan mengubah rancangan (*design*) menjadi kode program. Karena semua telah direncanakan dan dipikirkan sebelumnya, maka proses coding dapat dianggap sebagai sebuah pembuktian (test) bahwa rancangan yang dibuat sudah benar. Terkadang terdapat beberapa hal yang lolos dari perancangan dan baru terungkap saat coding; pada kasus tersebut, perubahan pada rancangan harus segera dilakukan sehingga kode program dan rancangan bisa tetap sinkron.

Bila pembuatan kode program tiba-tiba menjadi tidak terkendali (pada kasus skripsi, mahasiswa tiba-tiba merasa seolah otaknya hendak meledak dan jadi malas coding), maka ada beberapa kemungkinan:

- Hasil rancangan tidak bagus.
- Programmer tidak mengikuti hasil rancangan yang bagus dan mengerjakannya sesuka hatinya.
- Programmer tidak diikutsertakan dalam proses perancangan

- Mahasiswa tidak mau pikir panjang/hanya copy paste di bab analisa & perancangan, dalam pikirannya mau segera fokus ke bab implementasi dan kode program;
- Mahasiswa hanya memikirkan bab analisa & perancangan, tidak mau peduli dampaknya pada saat membuat kode program nanti.

FILED UNDER SOFTWARE ENGINEERING TAGGED WITH ICONIX PROCESS, UML

Perihal Solid Snake
I'm nothing...

55 Responses to *Merancang Sistem Dengan UML: Mulai Dari Mana?*

Ping-balik: Membuat Aplikasi CRUD MySQL Dalam 10 Menit Dengan Griffon | The Solid Snake

Komang Hendra Santosa says:
27 Juli 2013 pukul 6:04 PM
Saya masih kurang begitu mengerti mengenai UML ini, gimana ya caranya biar gampang mengertinya ya mas?

Balas

Solid Snake says:
27 Juli 2013 pukul 8:02 PM
Caranya adalah dengan mempraktekkannya. Dengan membuat implementasi dari rancangan, kita menjadi semakin mengerti ‘*kenapa begini*‘ dan ‘*kenapa begitu*‘. Membaca buku yang disebutkan di artikel ini juga akan membantu.

Balas

x1m4 says:
09 September 2013 pukul 1:27 PM
keren mas penjelasannya.. lebih mengena daripada teori yang d buku thx..

Balas

Afiz Ahmad Fauzi says:
13 Oktober 2013 pukul 8:27 AM
mas, cara buat robustness analysis nya gimana? saya buat use case nya pake starUML.

Balas

Steven Way says:
14 Oktober 2013 pukul 2:47 PM
Coba pakai Enterprise Architect, buat sebuah analysis diagram baru:
http://www.sparxsystems.com/enterprise_architect_user_guide/10/domain_based_models/analysisdiag

Enterprise Architect adalah salah satu tools yang mendukung ICONIX process:
http://www.sparxsystems.com/enterprise_architect_user_guide/10/software_engineering/iconix_proce

Balas

Syamsul Mustaqim says:
19 November 2013 pukul 2:54 PM

mantap sekali gan 😀
ane mahasiswa juga bingung kalo cuman dapet teori formal, ga dijelasin langkahnya, tujuannya, harus mulai darimana …

Balas

satya bayu Aji says:
15 Januari 2014 pukul 12:37 AM
sangat membantu sekali,,, perbedaan antara UML booch sama use case driven itu apa gan???

Balas

Steven Way says:
15 Januari 2014 pukul 1:09 PM
Graddy Booch adalah salah satu pencipta UML. UML adalah notasi grafis multi-fungsi untuk keperluan pengembangan software. UML hanya menyediakan visualisasi. *Use case driven development* adalah salah satu panduan dalam pengembangan software.

Analogi: Bila UML adalah *Adobe Photoshop*, maka use case driven development adalah *panduan mencari inspirasi gambar*. Seorang designer dapat memakai *Adobe Photoshop* dalam berbagai cara karena memang *Adobe Photoshop* tidak mengatur step-by-step dalam menggambar.

Balas

satya bayu Aji says:
16 Januari 2014 pukul 6:07 AM
kesimpulannya jadi use case driven tidak terlalu banyak membahas pada analisa, design maupun implementasi programnya tapi lebih melihat kepada kebutuhan pengguna serta menyederhanakan proses tersebut,
terima kasih gan atas jawabannya 😀

Balas

Olala Lala says:
04 Maret 2014 pukul 2:53 PM
maaf mas, kalau misalnya kita eksekusi ke coding, gimana ya cara implemtasinya. soalnya saya juga ada kerjaan seperti ini, cuman setelah ke coding jadi bingung lagi 😟

Balas

Solid Snake says:
05 Maret 2014 pukul 1:30 PM
Untuk implementasi ke kode program, serahkan pada programmer. Tunjukkan hasil rancangan dan beri tahu apa yang harus mereka buat! Mereka akan senang karena mereka sudah tahu apa yang harus mereka lakukan.

Balas

Steven Way says:
06 Maret 2014 pukul 11:27 AM
Sekedar fakta dan tips agar sukses…

Domain expert adalah orang paling alih di bidang yang hendak dikomputerisasikan. Analyst dan developer tidak akan pernah bisa menyaingi pengalaman domain expert.
Domain expert bisa jadi adalah client, atau bisa juga pihak ketiga yang berpengalaman di bidang tersebut.
Domain expert tidak bisa membuat sistem informasi dan melakukan pemograman karena bukan bidangnya.
Analyst bertugas menerjemahkan kebutuhan domain expert dan melihatnya dari sisi sistem informasi.

Analyst bukan menggambarkan diagram yang mewakili proses bisnis semata wayang karena domain expert bisa melakukan ini sendiri tanpa adanya analyst!
Diagram yang dihasilkan analyst adalah kebutuhan domain expert yang sudah disesuaikan dengan kelebihan dan kekurangan teknologi yang dipakai seperti framework, web, desktop.
Pada saat merancang sistem berbasis web, analyst memperhatikan skalabilitas, dampak penggunaan session dan cookie, dan bagaimana mudahnya implementasi nanti.
Pada saat memakai ORM seperti Hibernate atau Doctrine + Symfony, analyst memperhatikan keterbatasan class yang ada karena tidak semua jenis class dapat diimplementasikan. Misalnya, analyst menghindari relasi many-to-many dan one-to-many unidirectional di class diagram. Bukan karena tidak boleh, tapi karena frameworknya terbatas!
Programmer hanya perlu coding karena analyst sudah memperhatikan kebutuhan programmer pada saat merancang.
Analyst memperhatikan kebutuhan domain expert dan kebutuhan programmer dalam menghasilkan rancangan.

Balas

Yogie Kurniawan says:
07 Maret 2014 pukul 8:49 AM
Terimakasih gan artikelnya sangat membantu, tp saya ada sedikit pertanyaan untuk domain model yang telah lengkap versi Bidirectional kenapa <> JenisPekerjaan jadi hilang ?

Balas

Solid Snake says:
07 Maret 2014 pukul 2:24 PM
Itu adalah sebuah kesalahan di diagram tersebut. Harusnya class tersebut tetap ada dan tidak hilang dengan posisi sama seperti sebelumnya.

Balas

Asep Mulyana says:
20 Maret 2014 pukul 9:47 AM
Terima kasih gan, sangat membantu..

Balas

Dede Wahyu Hidyat says:
22 Maret 2014 pukul 12:52 PM
penjelasannya bagus banget gan. thanks gan

Balas

Fauzi Alhari says:
11 April 2014 pukul 6:04 PM
Terima kasih penjelasannya sangat membantu, Saya biasanya pakai power designer, tapi untuk robustness diagramnya susah, skrang sya mulai berusaha pakai EA, tapi tutorialnya kurang, apalagi untuk iconix, mas bisa minta referensi untuk tutorialnya?

Balas

Syamsul Mustaqim says:
15 April 2014 pukul 9:09 PM
bang, ane lagi belajar bikin class diagram, tolong dikoreksi dan komentarin bisa g bang? ane kirimin ke emailnya kalo bisa, mohon bales ye bang, email ane : somesoul.m@gmail.com

Balas

Solid Snake says:

17 April 2014 pukul 10:08 AM

Diagram yang dihasilkan berdasarkan sebuah sudut pandang atas domain yang disepakati bersama oleh tim. Saya belum tentu memahami domain permasalahan (inventory, banking, healthcare, dsb) yang kamu hadapi. Selain itu, walaupun domainnya sama, setiap arsitek bisa punya point of view-nya masing-masing. Beberapa demi alasan waktu dan infrastruktur yang tidak memadai, merancang class tanpa banyak operasi seperti layaknya tabel yang hanya berisi data. Ada juga yang menghasilkan rich domain classes yang sudah mencakup operasi dan interaksi.

Untuk mengoreksi diagram-mu, lakukan intropeksi berikut ini:
1. Apakah seluruh anggota tim dapat memahami diagram tersebut, termasuk domain expert?
2. Apakah kamu dapat mengimplementasikan diagram tersebut dalam jangka waktu yang wajar dengan teknologi atau skill yang dimiliki?

Balas

Andy Dobleh (@AndyDobleh) says:
24 April 2014 pukul 1:32 AM
Mas info dong Software yang Open Source atau Trial yg recommended untuk Point yang Use case Dan Robustness Activity

Balas

Solid Snake says:
24 April 2014 pukul 12:46 PM
Saya memakai Enterprise Architect. Software ini tidak gratis tetapi versi trial-nya dapat di-download secara bebas.

Balas

Andy Dobleh (@AndyDobleh) says:
25 April 2014 pukul 12:44 AM
terimakasih mas

Balas

haryatithio says:
07 Mei 2014 pukul 10:43 AM
mas, maaf nih masih newbie, kalo metode ICONIX Process itu bagaimana ya? apakah termasuk metode pengembangan sistem..? sy lg buat rancangan dengan rational rose, apa sesuai dengan metode itu, atau tidak ada hubunganya (artinya, bisa saja digambarkan dengan tool apapun)

Balas

Solid Snake says:
07 Mei 2014 pukul 6:32 PM
ICONIX adalah salah satu metodologi pengembangan software yang tidak agile (sama halnya seperti Scrum, Agile, Extreme Programming, Cowboy Coding, dsb).

Tidak ada hubungannya dengan jenis dan tipe diagram yang dihasilkan. Metode pengembangan software yang beragam biasanya dapat memakai diagram yang umum dipakai seperti UML Diagram.

ICONIX hanya mengatur seperti kapan kamu membuat diagram dan sejenisnya (seperti yang ditunjukkan oleh step by step pada artikel ini). Pada metode lain, misalnya RAD yang populer dipakai oleh programmer VB dan Delphi, tidak ada perencanaan intensif sehingga pengembangan langsung pada membuat UI selaku prototype yang dikembangkan bertahap.

Metode pengembangan juga menentukan apa yang kamu anggap penting dalam proyek. Bila memakai ICONIX, proses dan interaksi yang mewakili bisnis adalah hal yang paling utama. Ini tidak selalu benar pada metode lainnya. Sebagai contoh, pada RAD, hasil rancangan GUI adalah hal utama yang diperhatikan. GUI, pada RAD, biasanya adalah indikator utama dalam melihat perkembangan proyek.

Mahasiswa pemula biasanya cenderung memakai metode “code and fix” dimana mereka langsung membuat kode program tanpa rancangan terlebih dahulu. Mereka akan memperbaikinya pada saat-saat tertentu. Ini umumnya menghasilkan “code monkey” yang meniru dan melakukan hal repetitif seolah-olah pemograman adalah hal sederhana tanpa memahami kompleksitas yang ada.

Balas

Indri Veronika Ulmasembun says:
10 Agustus 2014 pukul 12:50 AM
gan bisa buat postingan tentang perbedaan extend sama include gak? lebih sederhana dan terperinci. mksh gan

Balas

Solid Snake says:
10 Agustus 2014 pukul 12:48 PM
Gw anggap yang dimaksud dengan `include` adalah *composition*. Pada *composition*, superclass dimasukkan ke dalam subclass sebagai sebuah variabel sehingga method yang ada di superclass dapat di-reuse atau dipanggil ulang.

Inheritance (melalui `extends`) bukan saja menawarkan *reuse* seperti pada *composition* tapi juga *subtyping*. Sebagai contoh, method `proses(Manusia m)` mengharapkan parameter berupa `Manusia`. Bila terdapat `Mahasiswa` dan `Dosen` yang diturunkan dari `Manusia`, maka mereka valid untuk disertakan sebagai argumen pada `proses(Manusia m)` tersebut.

Secara sederhana, *composition* adalah hubungan **has-a** (memiliki) sementara *inheritance* adalah hubungan **is-a** (adalah sebuah).

Balas

rahmadikhsan says:
19 September 2014 pukul 11:17 AM
nice info gan, terima kasih ilmunya, bagus juga nih sebagai tambahan ilmu buat ngerjain skripsi. Mohon izin share linknya ya. terima kasih

Balas

Ping-balik: Cari referensi di internet tentang UML, ketemu artikel bagus nih, Banyak membantu dalam penyusunan skripsi saya | Rahmad Ikhsan Blog

benhard says:
19 September 2014 pukul 5:07 PM
Pak, pada bagian “Kesalahan Rancangan Domain Model Akibat Berfokus Pada Penyimpanan Data”. Itu artinya kita harus fokus pada data yang terihat secara visual bagi end user, sedangkan data yang bersifat coding disembunyikan dari diagram, begitu pak?

Balas

Solid Snake says:
25 September 2014 pukul 12:46 PM

Justru sebaliknya, domain model **bukan** mewakili apa yang terlihat secara visual (model yang mewakili apa yang terlihat secara visual disebut **view model**). Domain model juga **bukan** mewakili apa yang akan disimpan ke database.

Domain model harus mewakili permasalahan yang dihadapi dan logic untuk menyelesaikannya. Semua solusi untuk permasalahan tersebut harus ada di domain model, misalnya dalam bentuk method atau pola.

Sebagai contoh, bila merancang model yang hanya berisi data, coba tanya: *'Apa saja operasi yang perlu dilakukan untuk menyimpan data ini? Apa saya yang perlu dihitung sebelum disimpan atau ditampilkan?'* Biasanya letak operasi ini tersebar di kode program UI. Pindahkan kode program sebagai method/operasi di class yang bersangkutan.

Balas

benhard says:
25 September 2014 pukul 3:07 PM
Jujur saya sewaktu kuliah dulu masih belum diajarkan mengenai pembuatan UML ini, tapi didunia kerja seperti untuk project kolaborasi penting sekali.
Ditempat kerja, saya hanya pakai usecase,flowcart dan ERD saja.

Kemudian, setiap saya baca panduan pembuatan UML, baik itu buku atau skripsi orang, mengapa selalu tampak TAK ADA YANG SAMA konsep dan simbol nya, berbeda dengan dengan Flowcart yang konsep dan tampilannya sama?
Saya kira, mungkin saya harus baca pembuatan UML yang lebih sedehana dulu, misalya inputan data barang dulu, sebelum membahas mengenai operasi trangsaksi.

Balas

Solid Snake says:
25 September 2014 pukul 10:17 PM
Flowchart dipakai untuk menggambarkan eksekusi program secara berurut. Ini adalah sesuatu yang ketat. UML Class Diagram dipakai untuk mengklasifikasikan masalah. Masing-masing perancang bisa memiliki sudut pandang mereka dalam melakukan klasifikasi. Tapi yang pasti, notasi UML selalu memiliki makna yang sama (seperti class, association, composition, dsb). UML dapat di-customize dengan memakai stereotype, tp notasi dasar-nya tetap memiliki makna yang sama.

Untuk contoh perbedaan flowchart dan class diagram, coba baca https://thesolidsnake.wordpress.com/2014/09/25/belajar-menerapkan-object-oriented-programming-oop/. Semoga bisa membantu dalam memahami OOP.

Balas

ananda 2014 says:
05 Oktober 2014 pukul 11:53 PM
waduh terimakasih banyak pak. saya lagi coba cari2 bahan tentang UML dan ini sangat membantu saya. terimaksih ya pak, karena penjelasannya sangat sederhana utk dipahami. thks

Balas

estu rizky says:
12 Oktober 2014 pukul 4:42 PM
bang, mau tanya nih, bisa ndak ERD diagram dirubah ke Class Diagram? Terus dalam kondisi ama ERD diagram dirumbah ke Class Diagram dan bagaimana caranya?

Balas

Solid Snake says:

13 Oktober 2014 pukul 12:57 AM
ERD dan Class Diagram memiliki tujuan dan fungsi yang sangat berbeda.

Class Diagram dipakai untuk menggambarkan klasifikasi kode program. Diagram ini menjawab pertanyaan seperti: “Bila saya ingin menambah fitur X, dimana saya harus melakukan perubahan kode program?”

ERD dipakai untuk menggambarkan aspek penyimpanan data, biasanya dalam bentuk tabel di database. Pada development modern yang memakai ORM seperti JPA, penyimpanan data berlangsung secara transparan dari object ke tabel. Dengan demikian, ERD akan sangat mubazir bila sudah ada Class Diagram dan penyimpanan data ditangani oleh teknologi seperti ORM dan OGM.

Balas

budakponcol says:
02 Desember 2014 pukul 8:17 AM
Pak kalo untuk persentasi ke si client menjelaskan sistem yg kita buat itu yg cocok pake diagram apa, Saya belajar uml, flow dokumen,dfd , masi bingung cara implementasinya

Balas

Solid Snake says:
02 Desember 2014 pukul 12:44 PM
Diagram seperti UML diagrams lebih ditujukan untuk kebutuhan anggota tim pengembangan software guna menyamakan visi dan misi.

Bila klien adalah *product owner* (bukan hal bagus!) yang memacu arah pengembangan software, maka klien wajib mengerti hal teknis sehingga komunikasi dapat dilakukan dengan berbagai diagram yang kamu pelajari.

Bila klien adalah *stakeholder* (sponsor atau pemberi dana), maka presentasi yang lebih efektif adalah penyampaian seperti apa saja peningkatan nilai bisnis yang dicapai setelah penggunaan software. Tampilan screenshoot mungkin akan sedikit membantu.

Balas

budakponcol says:
02 Desember 2014 pukul 1:15 PM
Di tmpat saya kuliah semester 6 ada mata kuliah opsi, saya masi bingung mau pilih opsi RPL atau sistem informasi, keduanya sepertinya saling membutuhkan,
Saya ingin menjadi developer sekaligus analyst, mohon pencerahannya pak,

Balas

Solid Snake says:
04 Desember 2014 pukul 2:19 PM
Pada metode tradisional, *analyst* (tepatnya *system analyst*) adalah seorang yang hanya memiliki spesialisasi di sistem informasi. Peran tunggal seperti ini kini dianggap tidak efektif seiring dengan ditinggalkannya waterfall.

Pada metode agile secara umum, setiap developer harus bersedia melakukan analisa. Dengan demikian, istilah *analyst* disini lebih merujuk kepada senior developer (berpengalaman) yang berperan menentukan arah pengembangan secara jangka panjang. Secara spesifiknya, bila kita melihat beberapa metode modern: DDD mewajibkan semua anggota tim memahami bisnis, tetapi SCRUM memberi kemungkinan bagi *business analyst* untuk mengambil peran sebagai

product owner. Walaupun demikian, baik *product owner* maupun *scrum master* bukan 'pemimpin' yang memberikan perintah (melainkan lebih ke arah 'hakim' yang menegakkan aturan) karena SCRUM tidak mengenal peran *project manager*.

Berdasarkan pandangan di atas, seandainya saya mencari SDM untuk proyek agile, maka saya akan mencari developer (sekaligus system analyst) yang kuliah di bidang computer science. Sementara itu, saya akan cenderung memilih *business analyst* lulusan sekolah bisnis (misalnya yang mememiliki gelar Master of Business Administration / MBA).

Untuk penjurusan yang hendak kamu ambil, silahkan konsultasikan lebih lanjut dengan dosen pembimbing atau ketua jurusan di kampusmu untuk memperoleh informasi yang lebih detail.

Balas

budakponcol says:
04 Desember 2014 pukul 5:12 PM
Terima kasih banyak atas penjelasannya pak, sangat detail dan juga mudah saya mengerti.
dengan bapak memberikan contoh atau kasus yang sekarang menjadi trend.
dengan memahami dunia kerja itu di era modern ini seperti apa dalam bidang yang sedang kita geluti saya jadi tau harus bagaimana. 🙂

pak boleh tidak saya minta contoh implementasi model proses yang saat ini sedang bapak gunakan dengan tim ?

Hariyadi Teguh says:
21 Januari 2015 pukul 8:31 AM
Reblogged this on Hariyadi Teguh.

Balas

Radhi Fadlillah says:
04 April 2015 pukul 8:02 PM
Pak, saya mau bertanya beberapa hal.

Yang pertama, langkah-langkah yang tertulis di atas merupakan contoh perancangan sistem dengan menggunakan metode ICONIX process. Di kampus saya saat ini kami cuma diajarkan metode pengembangan Waterfall saja. Apakah langkah-langkah untuk menggunakan UML di atas bisa diterapkan pada metodologi Waterfall, atau ada yang harus diubah ?

Yang kedua, pada diagram yang pertama, terlihat bahwa sebelum membuat Domain model dan Use case model, sebelumnya harus dibuat GUI prototype dulu. Pertanyaan saya, apakah harus dibuat GUI prototype dulu atau bisa langsung membuat domain model ?

Terima kasih

Balas

Solid Snake says:
05 April 2015 pukul 1:58 PM
Iya, UML tetap bisa digunakan sebagai artifact yang dihasilkan pada tahap analisa dan perancangan di waterfall.

Bila jumlah tim tidak banyak dan semuanya sudah terbiasa dengan sistem yang hendak dibangun, GUI prototype boleh dilewatkan dengan asumsi semua anggota tim punya bayangan yang sama seperti apa UI yang diharapkan. Secara pribadi, karena saya memakai generator dari domain class menjadi UI (dengan simple-jpa), saya langsung mulai dengan membuat domain model sederhana.

Balas

vina marwah says:
12 September 2015 pukul 6:46 AM
tanya pak. intinya adalah “bagaimana menulis use case yang efisien”

bagaimana menulis use case yang mempunyai lebih dari 1 aktor. misalnya seperti “Use Case Login”. Nha biasanya yang berhak untuk menggunakan Use Case Login ini kan, kalo gak Aktor Member ada juga Aktor Administrator. saya bingungnya disitu pak. kalo saya tulis satu-satu dari segi pengguna, sepertinya itu kurang efisien.
saya punya kasus gini pak. misal saya mempunyai use case “member mengisi biodata diri”. nha ketika member mengisi biodata diri kan, member ini mengisi beberapa field yang diperlukan, seperti nama, provinsi, kota, dll. nha sebernya use case “member mengisi biodata diri” ini kan juga membutuhkan use case “provinsi” dan use case “kota”. selain itu use case “provinsi” ini juga membutuhkan use case “tambah provinsi”, “edit provinsi”, dan “hapus provinsi”, begiti juga dengan use case “kota”. kalo misalnya ini saya buat, kayaknya juga agak berlebihan. tapi kalo use case – use case tersebut gak saya buat, saya bingung nulis hal tersebut pada makalah saya. ini saya lagi ngerjain tugas kuliah pak. sebelumnya terimakasih pak.

Balas

vina marwah says:
12 September 2015 pukul 6:49 AM
Maaf pak, itu ada 2 pertanyaan. udah saya kasih nomor, cuma gak tau kenapa, nomornya gak muncul.

Balas

Emm Elda Sari says:
22 Agustus 2016 pukul 12:45 AM
pake sofware apa ya untuk bikin use case diagram, activity diagram & sequence diagram

Balas

Kurniawan Ismail says:
08 September 2016 pukul 8:37 AM
ini baru tulisan mantab yang blm pernah terdengar gosipnya di bangku kuliah, btw sangat mengena sekali, selama ini bikin analisa kecil langsung koding, ini saya mau bikin skripsi udah jdi programnya, bingung mau bikin UML nya, untuk kelengkapan skripsi, terharu sekali stelah melihat membaca tulisan anda. hik hisk, ajarin dong bang tekniknya, pake tool apa yang bisa membantu, program saya presensi karyawan. deal harga fb: oktakurnaiwan@yahoo.com trims

Balas

Ping-balik: Object Oriented vs Structure Design - Irfan Darmawan's blogIrfan Darmawan's blog

Abdul Aziz Zulfikar says:
17 Desember 2017 pukul 9:39 PM
tutorial bermanfaat

Balas

Korina Dora Aini says:
06 Januari 2018 pukul 7:29 AM
mas gimana buat sequence digram yang menggunakan MVC sama penggunaan include yang benar itu gimana? soalnya saya menggunakan include masih disalahkan

Balas

Memsye Namira says:
26 April 2018 pukul 3:00 PM

trims banyak atas sharing ilmunya…

Balas

cakcak27blog says:
01 Juni 2018 pukul 8:04 AM
Masih bingung dalam pembuatan robustnes diagram , jadi robustnest diagram itu apakah sama fungsinya dengan flow chart, activity diagram, terus perbedaannya dengan sequence diagram itu apa ?

Balas

Akhi Kholid says:
10 Januari 2019 pukul 11:47 AM
terima kasih penjelasannya….

Balas

Jihan Prasasti says:
02 September 2019 pukul 10:04 PM
Permisi, mau tanya nih. Apa di domain model harus ada relasi generalisasi (is-a)?
Makasihh

Balas