

LECTURE NOTES

Software Engineering

Minggu 3

Requirement Modelling

LEARNING OUTCOMES

Setelah menyelesaikan pembelajaran ini, mahasiswa akan mampu:

- LO2 – Menjelaskan konsep dari proses model piranti lunak

Outline Materi (Sub-Topic) :

1. *Requirements Engineering*
2. *Eliciting Requirement*
3. *Developing Use Case*
4. *Negotiating Requirement*
5. *Validating Requirements*
6. Studi Kasus

ISI MATERI

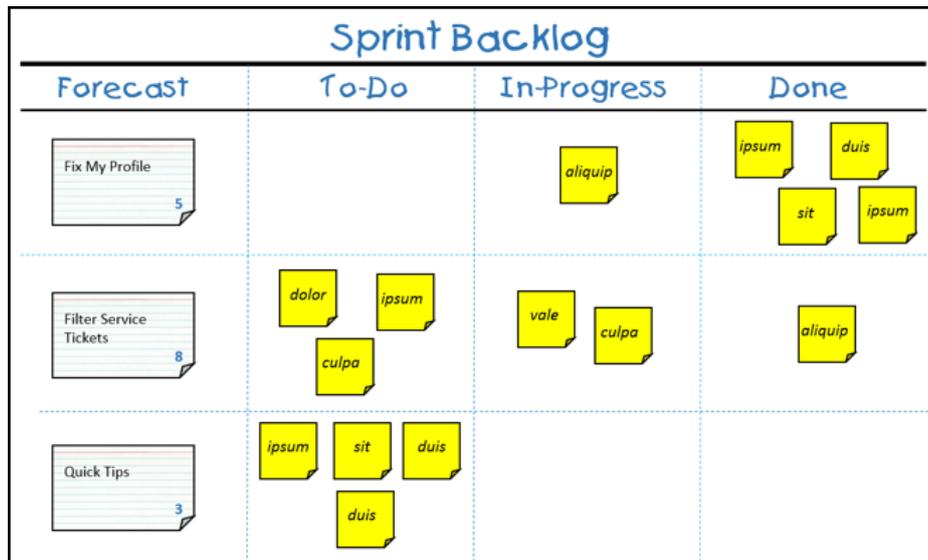
1. *Requirements Engineering*

Requirement engineering berhubungan bagaimana seorang *software engineer* melakukan proses untuk mendapatkan kebutuhan bisnis dari pengguna perangkat lunak dan mengelola kebutuhan tersebut untuk disiapkan menjadi proses selanjutnya, yaitu desain dan pengembangan. Seorang *software engineer* harus memiliki pengetahuan dan ketrampilan di dalam hal melakukan proses *requirement engineering* ini.

Requirement engineering merupakan suatu proses yang sangat penting di dalam proses pembangunan perangkat lunak. Banyak kasus yang terjadi akibat terjadi perbedaan persepsi atau ketidakkonsistenan antara kebutuhan pelanggan dan hasil dari perangkat lunak yang dibangun. Jika proses ini tidak dilakukan secara benar, maka akan terjadi potensi kegagalan implementasi perangkat lunak.

Pada proyek yang berbasis agile, *requirement engineering* ini dapat dilakukan dengan menggunakan konsep *product backlog*. *Product backlog* merupakan kumpulan dari *requirement* menggunakan konsep *user stories*. Berikut contoh dari *product backlog* menggunakan papan Kanban. Manajemen *requirement* ini menggunakan konsep *backlog refinement*.

Bahasan-bahasan selebihnya pada dokumen ini, lebih menitik beratkan proses pembentukan *requirement* dengan menggunakan notasi UML. Penggunaan notasi UML sangat baik digunakan untuk melakukan proses *requirement* untuk tipe pengembangan menggunakan konsep *waterfall*.



Requirement engineering dapat dikategorikan menjadi beberapa model:

- Model berdasarkan *scenario*, contoh: *use case diagram*, *user story*, *activity diagram*, *use case scenario*, dll. Pada pemodelan berdasarkan *scenario* bertujuan untuk membantu mendefinisikan *actor* dari sistem dan apa yang harus dilakukan oleh sistem tersebut.

Use case diagram merupakan salah satu model yang sering dipakai untuk menentukan fitur-fitur dari system yang akan dibangun. Use case menunjukkan fitur atau fungsionalitas yang ada pada system, misalnya:

- Fungsi untuk melihat saldo
- Fungsi untuk menarik uang di ATM
- Fungsi untuk transfer antar rekening
- Fungsi untuk melakukan administrasi perubahan akun

b. Model berdasarkan kelas, contoh: *class diagram*, *collaboration diagram*, dll. Model berdasarkan kelas menggambarkan beberapa hal berikut:

- *Object* dari sistem yang akan dimanipulasi
- Operasi (metode atau *service*) yang akan diterapkan terhadap *object* untuk mendapatkan efek dari manipulasi yang dilakukan.
- Hubungan antar *object*
- Kolaborasi yang muncul antara kelas yang didefinisikan.

Contoh:

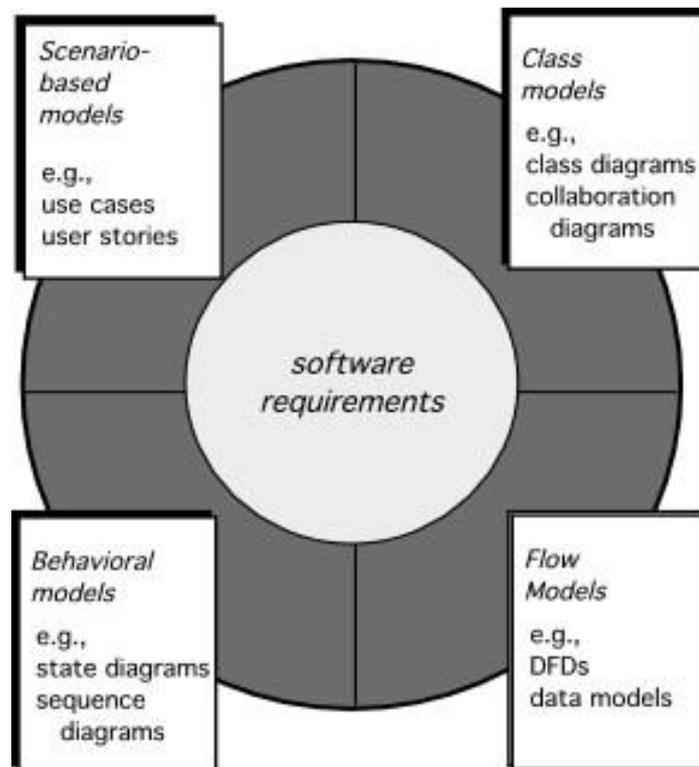
Jika kita ingin membuat suatu system penjualan yang melibatkan data produk, pelanggan dan supplier. Dengan menggunakan model berdasarkan kelas, kita dapat menggambarkan model-model untuk produk, pelanggan dan supplier sebagai kelas-kelas dan obyek yang saling berhubungan.

c. Model berdasarkan *behavior*, contoh *state diagram*, *sequence diagram*, dll. *Model behavior* mengindikasikan bagaimana *software* akan memberikan respon terhadap event atau stimulus external. Untuk menciptakan model ini, proses analisis harus dilakukan melalui tahapan berikut:

- Evaluasi semua *use case* untuk memahami secara keseluruhan urutan interaksi di dalam sistem.
- Identifikasikan sistem yang menggerakkan urutan interaksi dan memahami bagaimana event-event tersebut dikaitkan dengan *object specific*.
- Membuat *sequence* untuk setiap *use case*
- Membangun *state diagram* untuk sistem

- Melakukan *review* terhadap model *behavioral* untuk verifikasi akurasi dan konsistensi

d. Model berdasarkan aliran, contoh: data flow diagram, data *models*, dll.



2. *Eliciting Requirement*

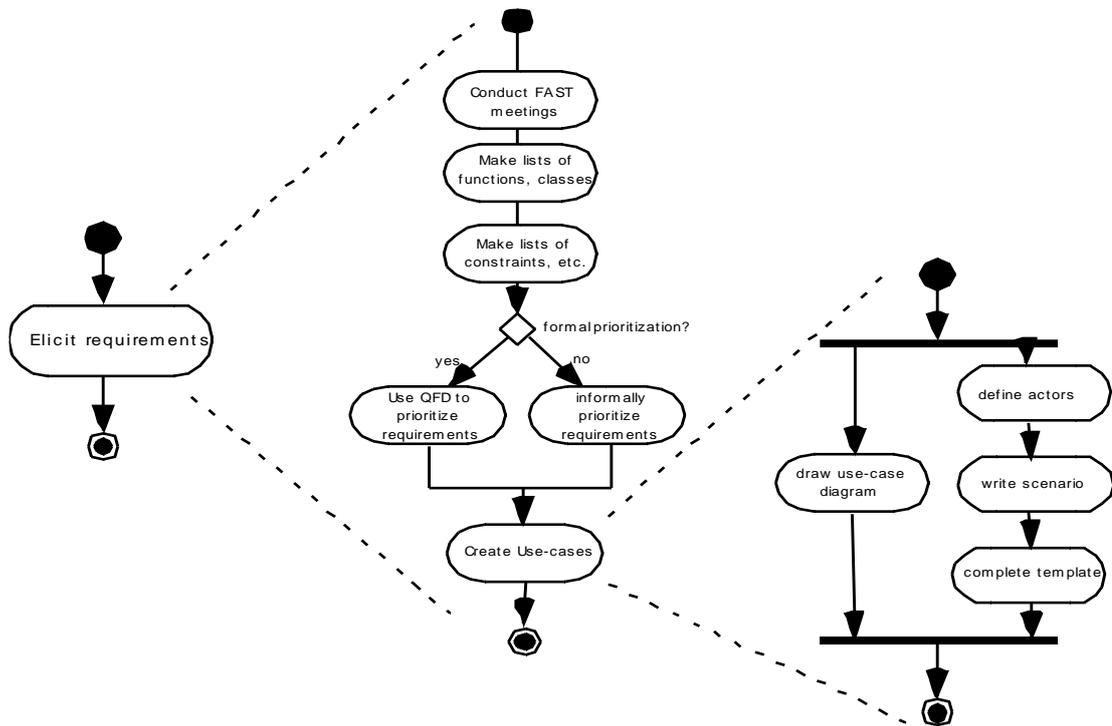
Mendapatkan (*elicit*) *requirement* merupakan salah satu kegiatan dalam *requirement engineering*. Sebelum membahas lebih dalam mengenai *requirement elicitation*, perlu dipahami secara keseluruhan apa saja kegiatan dalam *requirement engineering*:

- *Inception*, menanyakan sekumpulan pertanyaan yang telah dibuat mengenai pemahaman dasar mengenai masalah, siapa saja orang yang menginginkan solusi, bentuk dari solusi yang diinginkan, dan *efektifitate* dari komunikasi dan kolaborasi awal antara *customer* dan *developer*.

- *Elicitation*, mendapatkan semua kebutuhan dari semua *stakeholder* yang terlibat.
- *Elaboration*, membuat model analisis yang mengidentifikasi kebutuhan data, fungsi dan behavioral.
- *Negotiation*, persetujuan mengenai cara pengiriman sistem yang *realistic* untuk developer dan *customer*.
- *Specification*, dapat berupa dokumen tertulis, sekumpulan model, matematika formal, kumpulan dari user scenario, atau prototype.
- *Validation*, mekanisme *review* yang mencari error dalam konten, inkonsistensi, kebutuhan yang tidak realistic/memiliki konflik.
- *Requirement management*, manajemen untuk mengatur keseluruhan kegiatan yang ada dalam *requirement engineering*.

Dalam melakukan eliciting requirements, dibutuhkan sebuah pertemuan / meeting yang diadakan dan dihadiri oleh kedua pihak (*software engineer* dan *customer*). Dalam pertemuan tersebut, akan ditentukan aturan dalam persiapan dan bentuk partisipasi masing-masing pihak. Tujuan utama dari kegiatan eliciting requirements adalah untuk :

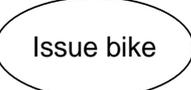
- Identifikasi masalah
- Mengajukan elemen-elemen solusi
- Negosiasi metode-metode yang berbeda
- Menspesifikasikan kebutuhan solusi diawal

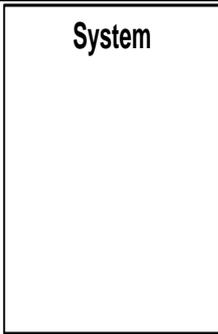


3. Developing Use Case

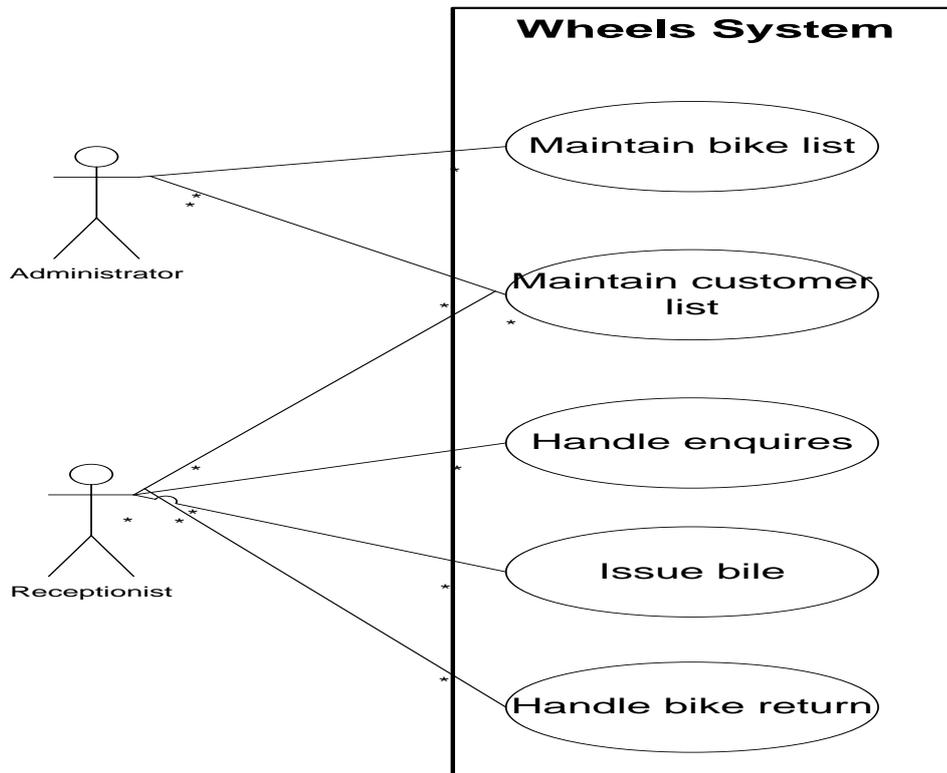
Diagram *use case* menggambarkan fungsionalitas dari sistem dan interaksi antara *user* dengan sistem. Model *use case* menyediakan fasilitas untuk mengorganisasikan, menstruktur dan mendokumentasikan informasi pada tahap pendefinisian kebutuhan sistem.

Berikut ini adalah notasi yang digunakan untuk menggambarkan diagram use case:

| Nama | Notasi | Keterangan |
|-----------------|---|---|
| <i>Use case</i> |  | Berbentuk elips dengan keterangan nama <i>use case</i> di tengah elips. Satu <i>use case</i> akan menggambarkan satu fungsionalitas yang dimiliki oleh sistem dan |

| | | |
|--------------------------|---|---|
| | | digambarkan di dalam notasi batasan sistem. |
| Aktor |  Receptionist | Semua aktor yang berhubungan langsung dengan sistem akan digambarkan dengan notasi ini. |
| Hubungan <i>use case</i> |  | Garis lurus digunakan untuk menghubungkan antara aktor dengan <i>use case</i> yang terkait. |
| Batasan sistem |  | Batasan sistem berupa persegi panjang dengan nama sistem di atas. Semua <i>use case</i> yang teridentifikasi akan digambarkan di dalam batasan ini. |

Berikut ini adalah contoh dari *use case* diagram untuk sebuah sistem “Wheel”, sistem penyewaan sepeda.



Pada gambar diagram *use case* di atas, terlihat bahwa terdapat dua actor yang akan menggunakan sistem; administrator dan *receptionist*. Administrator dapat menjalankan fungsionalitas untuk melakukan pemeliharaan terhadap daftar sepeda dan pelanggan. Sedangkan *receptionist* dalam mengakses use case untuk memelihara daftar pelanggan, melakukan penyewaan sepeda, pengembalian sepeda, dan pendataan sepeda.

4. *Negotiating Requirement*

Dalam melakukan negosiasi terhadap requiremen dari sebuah software beberapa hal yang harus diperhatikan adalah:

- Mengidentifikasi *stakeholder* utama (kunci), yang merupakan orang-orang yang akan terkait dalam negosiasi
- Menentukan “*win condition*” untuk setiap *stakeholders*. Kondisi menang disini tidak selalu jelas.
- Negosiasi, melakukan negosiasi langsung terhadap kebutuhan yang memberikan solusi “*win win*”.

5. *Validating Requirements*

Berikut ini adalah pertanyaan yang dapat digunakan untuk melakukan validasi terhadap requirement yang telah dirancang:

- Apakah setiap *requirement* dapat dicapai dalam lingkungan teknis dari sistem atau produk
- Apakah *requirement* dapat diuji ketika telah dikembangkan?
- Apakah *requirement* model secara layak merefleksikan informasi, fungsi dan *behavior* dari sistem yang akan dibuat?
- Sudahkah *requirement* dipartisi sehingga dapat menunjukkan detail informasi secara progresif.
- Sudahkah *pattern requirement* digunakan untuk menyederhanakan model *requirement*. Apakah semua *pattern* secara layak divalidasi? Apakah *pattern* konsisten dengan kebutuhan customer?

Studi Kasus

Di dalam implementasi proyek pengembangan perangkat lunak, tentunya diperlukan dokumentasi yang harus dilakukan, misalnya dalam bentuk:

- Functional Specification Document (FSD)
- Software Requirement Specification (SRS)

Pembentukan dokumen ini dapat mengikuti panduan yang telah dijelaskan, diawali dengan pembentukan use case.

Untuk metode pengembangan dengan agile, dimulai dari pembentukan product backlog dan product backlog item. Product backlog sendiri pada dasarnya merupakan suatu daftar requirement dalam bentuk yang fitur produk yang akan dibangun.

DAFTAR PUSTAKA

- Software engineering : a practitioners approach : Chapter 8/Pages 131
Chapter 9/Pages 166, Chapter 10/ Pages 184,
- Requirements Engineering / Specification,, <http://www.youtube.com/watch?v=wEr6mwquPLY>
- Collaborative Requirements Management,,
<http://www.youtube.com/watch?v=tEXizjE05LA>
- UML 2.0 Tutorial,, <http://www.youtube.com/watch?v=OkC7HKtiZC0>
- Scrum.org