SLIDE 2

Analisis kebutuhan adalah proses yang menghasilkan spesifikasi karakteristik operasional perangkat lunak, menunjukkan antarmuka perangkat lunak dengan elemen-elemen sistem lainnya, dan menetapkan batasan-batasan yang harus dipenuhi oleh perangkat lunak. Analisis kebutuhan memungkinkan para profesional (baik disebut insinyur perangkat lunak, analis, atau pemodel) untuk mengembangkan kebutuhan dasar yang telah ditetapkan selama tahap-tahap awal seperti inisiasi, pengumpulan kebutuhan, dan negosiasi yang merupakan bagian dari rekayasa kebutuhan.

Dalam analisis kebutuhan, perangkat lunak harus diuraikan secara rinci mengenai bagaimana seharusnya beroperasi. Ini mencakup fungsionalitas yang diharapkan, kinerja yang diinginkan, dan segala fitur khusus yang diperlukan oleh pengguna. Perangkat lunak seringkali harus berinteraksi dengan elemen-elemen lain dalam sistem. Ini bisa berupa interaksi dengan perangkat keras tertentu, perangkat lunak lain, atau bahkan dengan pengguna langsung. Analisis kebutuhan harus menjelaskan dengan jelas bagaimana perangkat lunak akan berkomunikasi dengan elemen-elemen ini.

Setiap perangkat lunak memiliki batasan-batasan tertentu yang harus dipatuhi. Batasan ini bisa berupa batasan teknis (misalnya, platform atau bahasa pemrograman tertentu harus digunakan), batasan waktu (perangkat lunak harus selesai dalam jangka waktu tertentu), atau batasan lainnya yang mungkin diberlakukan oleh klien atau organisasi terkait. Analisis kebutuhan harus dengan jelas menetapkan batasan-batasan ini agar perangkat lunak dapat dikembangkan sesuai dengan parameter yang telah ditetapkan.

SLIDE 3

Dari gambar dapat di lihat bahwa analisis kebutuhan yang mencakup 4 elemen, yaitu **Scenario-based Model, Class Model, Behavior Model, dan Flows Model**. Elemen-elemen ini digunakan untuk memahami, merinci, dan mendokumentasikan kebutuhan perangkat lunak secara lebih terperinci.

1. **Scenario-based Model**

Scenario-based model berfokus pada situasi atau skenario penggunaan sistem. Ini mencakup contoh-contoh kasus penggunaan di mana sistem akan digunakan. Skenario mencakup langkah-langkah yang diambil oleh pengguna atau sistem dalam situasi tertentu. Ini membantu dalam memahami bagaimana pengguna berinteraksi dengan sistem dalam konteks nyata.

2. **Class Model**

Class model melibatkan identifikasi entitas utama yang akan ada dalam sistem dan hubungan antara entitas tersebut. Ini menggunakan konsep-konsep dari pemrograman berorientasi objek (OOP) dan menggambarkan kelas-kelas, atribut-atribut, dan metode-metode yang terkait dengan entitas-entitas ini.

3. **Behavior Model**

Behavior model menggambarkan bagaimana entitas dalam sistem berinteraksi satu sama lain dan bagaimana mereka merespons terhadap peristiwa-peristiwa tertentu. Ini mencakup diagram use case, diagram aktivitas, atau diagram keadaan yang menunjukkan perilaku sistem dalam berbagai situasi.

4. **Flows Model**

Flows model menggambarkan alur data dan kontrol melalui sistem. Ini mencakup diagram alir data (Data Flow Diagrams) dan diagram alir kontrol (Control Flow Diagrams) yang menunjukkan bagaimana data mengalir melalui sistem dan bagaimana kendali mengalir di antara komponen-komponen sistem.

SLIDE 4

Tindakan pemodelan kebutuhan menghasilkan satu atau lebih jenis model berikut:

1. **Model Berbasis Skenario**. Model-model ini menggambarkan kebutuhan dari sudut pandang berbagai "aktor" dalam sistem. Dalam konteks ini, "aktor" mengacu pada elemen-elemen sistem yang memiliki peran atau

keterlibatan tertentu dalam memenuhi kebutuhan tersebut. Skenario ini memberikan gambaran tentang bagaimana sistem akan digunakan dalam situasi-situasi nyata oleh berbagai pihak yang terlibat.

2. **Model Berbasis Kelas**. Model ini mewakili kelas-kelas berorientasi objek (object-oriented classes) yang mencakup atribut dan operasi. Selain itu, model ini menjelaskan cara kelas-kelas tersebut bekerja sama untuk mencapai kebutuhan sistem. Dalam konteks pemrograman berorientasi objek, kelas merujuk pada entitas-entitas yang memiliki atribut dan perilaku tertentu.
3. **Model Berbasis Perilaku dan Pola** (Behavioral and Patterns-Based Models). Model-model ini menggambarkan bagaimana perangkat lunak berperilaku sebagai akibat dari "peristiwa-peristiwa" eksternal. Peristiwa eksternal ini dapat mencakup input pengguna, pesan dari sistem lain, atau perubahan kondisi di lingkungan sistem. Model ini membantu dalam memahami bagaimana perangkat lunak akan merespons terhadap stimulus eksternal dan bagaimana pola perilaku tertentu dapat diidentifikasi dan dipahami.
4. **Model Data**. Model-model ini menggambarkan domain informasi untuk masalah yang akan diselesaikan oleh perangkat lunak. Mereka mendefinisikan struktur dan hubungan antara data dan informasi yang akan dikelola oleh sistem. Model data membantu dalam memahami jenis data yang akan digunakan, serta cara data tersebut akan disimpan, diakses, dan dikelola oleh sistem.
5. **Model Berbasis Alur** (Flow-Oriented Models). Model-model ini mewakili elemen-elemen fungsional dalam sistem dan bagaimana mereka mengubah data saat bergerak melalui sistem. Mereka menggambarkan alur data dan kontrol melalui berbagai komponen sistem. Model ini membantu dalam memahami bagaimana data diproses dan dipindahkan melalui sistem, serta bagaimana sistem menjalankan fungsionalitas yang diperlukan.

Model-model ini memberikan informasi kepada perancang perangkat lunak yang dapat diterjemahkan ke dalam desain arsitektural, antarmuka, dan komponen. Akhirnya, model kebutuhan (dan spesifikasi kebutuhan perangkat lunak) memberikan sarana bagi pengembang dan pelanggan untuk menilai kualitas perangkat lunak setelah dibangun. Dalam konteks ini, pemahaman dan analisis

yang cermat terhadap kebutuhan adalah tahap kritis dalam pengembangan perangkat lunak.

SLIDE 5

Selama pemodelan analisis, fokus utama adalah pada "apa", bukan "bagaimana". Dalam pembahasan-pembahasan sebelumnya, telah disebutkan bahwa spesifikasi kebutuhan yang lengkap mungkin tidak selalu dapat dicapai pada tahap ini. Pelanggan mungkin tidak yakin dengan persis apa yang diperlukan untuk beberapa aspek sistem. Pengembang mungkin juga ragu apakah pendekatan tertentu akan berhasil dalam mencapai fungsi dan kinerja yang diinginkan. Oleh karena itu, pendekatan iteratif terhadap analisis dan pemodelan kebutuhan menjadi lebih disukai. Analis harus memodelkan apa yang diketahui dan menggunakan model tersebut sebagai dasar untuk desain perangkat lunak secara bertahap.

Model kebutuhan harus mencapai tiga tujuan utama:

1. **Menggambarkan apa yang dibutuhkan oleh pelanggan**. Ini adalah langkah awal yang penting dalam proses pengembangan perangkat lunak. kita perlu memahami apa yang diinginkan oleh pelanggan atau pemangku kepentingan dalam sistem yang akan dibangun.
2. **Membentuk dasar untuk pembuatan desain perangkat lunak**. Model analisis ini digunakan untuk merancang arsitektur perangkat lunak, antarmuka pengguna, dan struktur level komponen. Ini membantu dalam memahami bagaimana perangkat lunak akan dibangun untuk memenuhi kebutuhan.
3. **Menyusun seperangkat kebutuhan yang dapat divalidasi setelah perangkat lunak dibangun**. Model analisis ini harus memberikan seperangkat persyaratan yang jelas yang dapat diuji dan diverifikasi setelah perangkat lunak dibangun. Ini memastikan bahwa perangkat lunak yang dihasilkan memenuhi kebutuhan yang telah ditetapkan oleh pelanggan.

Seperti yang ada di gambar, Model analisis menjembatani kesenjangan antara deskripsi tingkat sistem yang menggambarkan fungsionalitas sistem atau bisnis secara keseluruhan dengan menerapkan perangkat lunak, perangkat keras,

data, manusia, dan elemen sistem lainnya, dengandan desain perangkat lunak yang menggambarkan arsitektur aplikasi perangkat lunak, antarmuka pengguna, dan struktur tingkat komponen.

SLIDE 6

Adapun aturan praktis analisis yang harus diikuti ketika membuat model analisis:

1. **Fokus pada Kebutuhan Terlihat**. Model harus memusatkan pada kebutuhan yang terlihat dalam domain masalah atau bisnis. Tingkat abstraksi harus relatif tinggi. Jangan terjebak dalam detail-detail yang mencoba menjelaskan bagaimana sistem akan bekerja.
2. **Setiap Elemen Menyumbang pada Pemahaman Keseluruhan**. Setiap elemen dari model kebutuhan harus menambahkan pemahaman keseluruhan tentang kebutuhan perangkat lunak dan memberikan wawasan tentang domain informasi, fungsi, dan perilaku sistem.
3. **Tunda Pertimbangan Infrastruktur**. Pertimbangkan infrastruktur dan model nonfungsional lainnya hanya pada tahap desain. Misalnya, database mungkin diperlukan, tetapi kelas-kelas yang diperlukan untuk mengimplementasikannya, fungsi-fungsi yang diperlukan untuk mengaksesnya, dan perilaku yang akan ditunjukkan saat digunakan harus dipertimbangkan hanya setelah analisis domain masalah selesai.
4. **Minimalisir Keterkaitan**. Minimalkan keterkaitan (coupling) di seluruh sistem. Penting untuk merepresentasikan hubungan antara kelas-kelas dan fungsi-fungsi. Namun, jika tingkat "keterhubungan" sangat tinggi, upaya harus dilakukan untuk menguranginya.
5. **Berikan Nilai pada Semua Pihak Terkait**. Pastikan model kebutuhan memberikan nilai kepada semua pihak terkait. Setiap kelompok pemangku kepentingan memiliki kegunaan masing-masing dari model tersebut. Misalnya, pemangku kepentingan bisnis harus menggunakannya untuk memvalidasi kebutuhan; perancang harus menggunakannya sebagai dasar untuk desain; tim QA harus menggunakannya untuk merencanakan uji penerimaan.

6. **Jaga Kesederhanaan Model**. Pertahankan model se-sederhana mungkin. Jangan tambahkan diagram tambahan jika mereka tidak menambah informasi baru. Jangan menggunakan bentuk notasi yang kompleks jika daftar sederhana sudah cukup.

SLIDE 7

Firesmith menjelaskan **analisis domain** sebagai identifikasi, analisis, dan spesifikasi kebutuhan umum dari suatu domain aplikasi tertentu, biasanya untuk digunakan kembali dalam beberapa proyek di dalam domain aplikasi tersebut. Dalam konteks analisis domain berbasis objek, ini melibatkan identifikasi, analisis, dan spesifikasi kemampuan umum yang dapat digunakan kembali dalam suatu domain aplikasi tertentu, dalam bentuk objek, kelas, sub-assemblies, dan kerangka kerja yang umum.

Domain aplikasi tertentu ini dapat mencakup berbagai bidang, mulai dari avionika hingga perbankan, dari permainan video multimedia hingga perangkat lunak tertanam dalam perangkat medis. Tujuan analisis domain adalah sederhana: menemukan atau menciptakan kelas analisis dan/atau pola analisis yang sangat berlaku agar dapat digunakan kembali.

Secara kontekstual, analisis domain dapat dilihat sebagai kegiatan payung bagi proses perangkat lunak. Dalam hal ini, analisis domain adalah kegiatan rekayasa perangkat lunak yang berkelanjutan dan tidak terkait dengan satu proyek perangkat lunak tertentu. Dalam analogi ini, peran seorang analis domain mirip dengan peran seorang perajin alat yang ahli dalam lingkungan manufaktur berat. Tugas perajin alat adalah merancang dan membangun alat-alat yang dapat digunakan oleh banyak orang dalam pekerjaan yang mirip, meskipun tidak selalu sama. Demikian pula, peran analis domain adalah menemukan dan mendefinisikan pola analisis, kelas analisis, dan informasi terkait yang dapat digunakan oleh banyak orang yang bekerja pada aplikasi yang mirip, meskipun tidak selalu identik. Dengan kata lain, analis domain bertanggung jawab untuk mengidentifikasi prinsip-prinsip umum, kelas, atau pola yang dapat diterapkan secara luas, memastikan bahwa pengetahuan ini dapat digunakan kembali di banyak proyek yang berbeda tetapi terkait dalam domain aplikasi tertentu.

SLIDE 8

Proses analisis domain melibatkan langkah-langkah sebagai berikut:

1. **Mendefinisikan Domain yang Akan Dianalisis**. Langkah pertama dalam analisis domain adalah mendefinisikan domain atau area aplikasi yang akan diselidiki. Misalnya, domain aplikasi bisa menjadi sistem keuangan, perawatan kesehatan, manufaktur, atau bidang lainnya.
2. **Mengumpulkan Sampel Representatif Aplikasi dalam Domain**. Setelah domain didefinisikan, langkah berikutnya adalah mengumpulkan sampel aplikasi yang mewakili berbagai kasus penggunaan dalam domain tersebut. Sampel ini penting untuk memahami variasi kebutuhan dan pola penggunaan dalam domain tersebut.
3. **Menganalisis Setiap Aplikasi dalam Sampel**. Analisis dilakukan terhadap setiap aplikasi dalam sampel tersebut. Ini mencakup memahami cara aplikasi bekerja, kebutuhan pengguna, pola penggunaan, dan elemen-elemen sistem lainnya yang terlibat.
4. **Mengembangkan Model Analisis untuk Objek-objek**. Setelah analisis aplikasi selesai, model analisis dikembangkan untuk objek-objek (seperti kelas atau entitas) yang ditemukan dalam domain tersebut. Model ini mencakup struktur objek, hubungan antar-objek, dan perilaku objek dalam konteks domain aplikasi tersebut.

SLIDE 9

Meskipun keberhasilan suatu sistem atau produk berbasis komputer dapat diukur dengan berbagai cara, kepuasan pengguna menduduki posisi paling penting dalam daftar tersebut. Jika kita memahami bagaimana pengguna akhir (dan aktor-aktor lainnya) ingin berinteraksi dengan suatu sistem, tim pengembangan perangkat lunak akan lebih mampu menggambarkan kebutuhan dengan benar dan membangun model analisis dan desain yang bermakna. Oleh karena itu, pemodelan kebutuhan dengan menggunakan UML dimulai dengan pembuatan skenario dalam bentuk use cases.

Use Cases digunakan untuk mendefinisikan apa yang ada di luar sistem (aktor-aktor) dan apa yang harus dilakukan oleh sistem (use cases) tersebut. Pernyataan tersebut menyajikan pertanyaan-pertanyaan penting yang perlu dijawab ketika kita menggunakan kasus penggunaan dalam pemodelan skenario:

1. Apa yang Harus Kita Tulis Tentang?
2. Seberapa Banyak yang Harus Kita Tulis Tentang Hal Ini?
3. Seberapa Rinci Harus Kita Membuat Deskripsi Kita?
4. Bagaimana Kita Harus Mengorganisasi Deskripsi Ini?

SLIDE 10

Ada dua tahap awal dalam rekayasa kebutuhan (requirements engineering), yaitu tahap inisialisasi (inception) dan tahap elicitation. Dalam tahap ini, informasi yang dibutuhkan untuk memulai penulisan use cases diperoleh. Pertemuan pengumpulan kebutuhan, teknik Quality Function Deployment (QFD), dan mekanisme rekayasa kebutuhan lainnya digunakan untuk mengidentifikasi pemangku kepentingan (stakeholders), menentukan cakupan masalah, menetapkan tujuan operasional umum, mengatur prioritas, menguraikan semua kebutuhan fungsional yang diketahui, serta mendeskripsikan objek-objek yang akan dimanipulasi oleh sistem.

Untuk memulai pengembangan serangkaian use cases, langkah pertama adalah mencantumkan fungsi atau aktivitas yang dilakukan oleh aktor tertentu. Dalam konteks ini:

1. **Identifikasi Fungsi atau Aktivitas**. Dalam tahap-tahap awal rekayasa kebutuhan, perlu diidentifikasi tindakan atau fungsi yang akan dilakukan oleh aktor-aktor yang terlibat dalam sistem. Aktor-aktor ini bisa berupa pengguna, sistem eksternal, atau entitas lain yang berinteraksi dengan sistem yang akan dikembangkan.
2. **Deskripsi Fungsi atau Aktivitas Aktor**. Untuk setiap aktor, fungsi-fungsi atau aktivitas-aktivitas yang diharapkan harus dijelaskan secara rinci. Misalnya, jika sistem adalah sistem pemesanan online, maka aktor

"Pelanggan" mungkin memiliki fungsi-fungsi seperti "Membuat Pemesanan," "Melihat Riwayat Pembelian," atau "Membatalkan Pesanan."

3. **Pemetaan Fungsi atau Aktivitas ke Use Cases**. Setelah fungsi atau aktivitas dari aktor-aktor diidentifikasi, mereka dapat dipetakan ke dalam use cases yang sesuai. Setiap use case akan mewakili suatu skenario interaksi antara satu atau lebih aktor dengan sistem, menggambarkan langkah-langkah yang diambil oleh aktor untuk mencapai tujuan tertentu dalam konteks sistem yang sedang dikembangkan.

Dengan melakukan langkah-langkah ini, tim pengembangan perangkat lunak dapat memahami peran dan tujuan masing-masing aktor serta merinci cara mereka berinteraksi dengan sistem. Ini membentuk dasar untuk mengembangkan use cases yang mendalam dan relevan, yang akan membantu dalam merinci kebutuhan sistem dan memahami bagaimana sistem harus berinteraksi dengan pengguna dan pemangku kepentingan lainnya.

SLIDE 11

Proses pengumpulan kebutuhan berlanjut ketika tim mengadakan percakapan lebih lanjut dengan pemangku kepentingan. Selama berlangsungnya diskusi dan interaksi lebih lanjut dengan pihak-pihak yang terlibat (stakeholders), tim pengumpulan kebutuhan menciptakan use cases untuk setiap fungsi atau aktivitas yang dicatat selama tahap awal analisis. Use cases ini adalah cerita rinci yang menggambarkan bagaimana sistem akan digunakan oleh aktor-aktor yang berbeda dalam konteks situasi tertentu.

Use cases pertama-tama ditulis dalam bentuk naratif yang informal. Naratif ini adalah deskripsi cerita yang menjelaskan langkah-langkah yang diambil oleh aktor (pengguna, sistem eksternal, dll.) dalam menggunakan sistem untuk mencapai tujuan tertentu. Format ini lebih santai dan bersifat deskriptif. Jika diperlukan tingkat formalitas yang lebih tinggi atau jika proyek membutuhkan dokumen yang lebih terstruktur, use cases yang awalnya ditulis dalam bentuk naratif informal dapat diubah ke dalam format terstruktur. Format terstruktur ini dapat mengikuti suatu standar tertentu atau templat yang mencakup bagian-bagian khusus, seperti deskripsi aktor, prakondisi, langkah-langkah, dan postkondisi.