



Pertemuan 7: Understanding Requirements

Meuthia Rachmaniah
Departemen Ilmu Komputer, FMIPA IPB

Software Engineering: A Practitioner's Approach, 9th Ed.
Roger S. Pressman dan Bruce R. Maxim
Copyright © 2020 McGraw-Hill Education



Hello!

Ir. Meuthia Rachmaniah, M.Sc.

Associate Professor

NIP 195907111984032006

Computer Science Department

Software Engineering & Information Sciences

meuthiara@apps.ipb.ac.id

SATYALANCANA KARYA SATYA 20 TAHUN

SATYALANCANA KARYA SATYA 30 TAHUN

7. Understanding Requirements

- 7.1 Requirements Engineering
- 7.2 Establishing the Groundwork
- 7.3 Requirements Gathering
- 7.4 Developing Use Cases
- 7.5 Building the Analysis Model
- 7.6 Negotiating Requirements
- 7.7 Requirements Monitoring
- 7.8 Validating Requirements
- 7.9 Rangkuman

QUICK LOOK

WHAT IS IT?



- Sebelum Anda memulai pekerjaan teknis apa pun, ada baiknya Anda membuat serangkaian kebutuhan untuk tugas-tugas teknik.
- Dengan menetapkan serangkaian kebutuhan, Anda akan memperoleh pemahaman tentang apa dampak bisnis dari PL, apa yang diinginkan pelanggan, dan bagaimana pengguna akhir akan berinteraksi dengan PL.

WHO DOES IT



- Software engineer dan stakeholder proyek lainnya (manajer, pelanggan, dan pengguna akhir) semuanya berpartisipasi dalam rekayasa kebutuhan.

WHY IS IT IMPORTANT?



- Untuk memahami apa yang diinginkan pelanggan sebelum Anda mulai merancang dan membangun sistem berbasis komputer.
 - Membangun program komputer elegan yang memecahkan masalah yang salah tidak membantu siapa pun.



WHAT ARE THE STEPS

- Inception - tugas yang mendefinisikan ruang lingkup dan sifat problem yang akan dipecahkan.
- Elisitasi - tugas yang membantu stakeholder menentukan apa yang diperlukan
- Elaborasi - kebutuhan dasar disempurnakan dan dimodifikasi.
- Negosiasi - Saat pemangku kepentingan mendefinisikan masalah, negosiasi terjadi (apa prioritasnya, apa yang esensial, kapan dibutuhkan?).
- Spesifikasi - problemnya ditentukan dalam beberapa cara
- Validasi – problem ditinjau atau divalidasi untuk memastikan bahwa pemahaman Anda tentang problem dan pemahaman stakeholder tentang problem tersebut bertepatan.



WHAT IS THE WORK PRODUCT


- Rekayasa kebutuhan menyediakan semua pihak pemahaman tertulis tentang problem.
- Produk kerja dapat mencakup: skenario penggunaan, daftar fungsi dan fitur, dan model kebutuhan.




HOW DO I ENSURE THAT I'VE DONE IT RIGHT

- Work product rekayasa kebutuhan adalah: ditinjau dengan stakeholder untuk memastikan bahwa semua orang berada di halaman yang sama.
- Sebuah kata peringatan: Bahkan setelah semua pihak setuju, segalanya akan berubah, dan mereka akan terus berubah sepanjang proyek

- ◆ Customer: *"Saya tahu Anda pikir Anda mengerti apa yang saya katakan, tetapi apa yang Anda tidak mengerti adalah apa yang saya katakan bukanlah apa yang saya maksudkan."*
- ◆ Selalu, ini terjadi di akhir proyek, setelah komitmen tenggat waktu dibuat, reputasi dipertaruhkan, dan uang serius dipertaruhkan.
- ◆ Kita semua yang telah bekerja di bisnis sistem dan perangkat lunak selama lebih dari beberapa tahun telah mengalami mimpi buruk ini, namun, hanya sedikit dari kita yang belajar untuk menghilangkannya.
- ◆ Kita berjuang ketika kita mencoba untuk mendapatkan kebutuhan dari pelanggan kita. Kita kesulitan memahami informasi yang kita peroleh.
- ◆ Kita sering mencatat kebutuhan dengan cara yang tidak teratur, dan kita menghabiskan terlalu sedikit waktu untuk memverifikasi apa yang kita rekam.
- ◆ Kita mengizinkan perubahan untuk mengendalikan kita, daripada membangun mekanisme untuk mengontrol perubahan.
- ◆ Singkatnya, kita gagal membangun fondasi yang kokoh untuk sistem atau PL. Masing-masing problem ini menantang. Ketika digabungkan, prospeknya menakutkan bahkan bagi manajer dan praktisi yang paling berpengalaman sekalipun. Tapi solusi memang ada.



7.1 Requirements Engineering

- 7.1.1 Inception
 - 7.1.2 Elicitation
 - 7.1.3 Elaboration
 - 7.1.4 Negotiation
 - 7.1.5 Specification
 - 7.1.6 Validation
 - 7.1.7 Requirements Management
- 

7.1 Requirements Engineering (Rekayasa Kebutuhan)

- 7.1.1 Inception
- 7.1.2 Elicitation
- 7.1.3 Elaboration
- 7.1.4 Negotiation
- 7.1.5 Specification
- 7.1.6 Validation
- 7.1.7 Requirements Management

Penting untuk diperhatikan bahwa beberapa tasks (7.2.1 s.d. 7.17) terjadi secara paralel dan semuanya disesuaikan dengan kebutuhan proyek

- ◆ Argumen Pengembang PL berpendapat:
 - ◆ bahwa segala sesuatunya akan menjadi jelas saat mereka membangun,
 - ◆ bahwa stakeholder proyek akan dapat memahami kebutuhan hanya setelah memeriksa iterasi awal PL,
 - ◆ bahwa segala sesuatunya berubah begitu cepat sehingga setiap upaya untuk memahami kebutuhan secara rinci adalah buang-buang waktu, itu intinya adalah menghasilkan program kerja, dan yang lainnya adalah sekunder.
- ◆ Apa yang membuat argumen ini menggiurkan adalah karena mengandung unsur kebenaran. Tetapi setiap argumen cacat dan dapat menyebabkan proyek perangkat lunak gagal.

7.1 Rekayasa Kebutuhan

- 7.1.1 Inception
- 7.1.2 Elicitation
- 7.1.3 Elaboration
- 7.1.4 Negotiation
- 7.1.5 Specification
- 7.1.6 Validation
- 7.1.7 Requirements Management

Penting untuk diperhatikan bahwa beberapa tasks (7.2.1 s.d. 7.17) terjadi secara paralel dan semuanya disesuaikan dengan kebutuhan proyek

- ◆ Requirements Engineering (Rekayasa Kebutuhan) adalah istilah untuk spektrum tasks dan teknik yang luas yang mengarah pada pemahaman tentang kebutuhan.
- ◆ Perspektif Proses PL tentang Rekayasa Kebutuhan:
 - ◆ adalah tindakan RPL utama yang dimulai selama aktivitas komunikasi dan berlanjut ke aktivitas pemodelan.
 - ◆ menetapkan dasar yang kokoh untuk desain dan konstruksi. Tanpa itu, PL yang dihasilkan memiliki kemungkinan besar untuk tidak memenuhi kebutuhan pelanggan.
 - ◆ harus disesuaikan dengan kebutuhan proses, proyek, produk, dan orang yang melakukan pekerjaan.
- ◆ Penting untuk disadari bahwa setiap task dilakukan secara berulang karena tim proyek dan stakeholder terus berbagi informasi tentang problem mereka masing-masing.

7.1 .1 Insepsi (Inception)

- ◆ Secara umum, sebagian besar proyek dimulai dengan kebutuhan bisnis yang teridentifikasi atau ketika pasar atau layanan baru yang potensial ditemukan



- ◆ Membangun pemahaman dasar tentang masalah, orang-orang yang menginginkan solusi, dan sifat solusi yang diinginkan.
- ◆ Komunikasi antara semua stakeholder dan tim PL perlu dibangun selama task ini untuk memulai kolaborasi yang efektif

7.1.2 Elisitasi (Elicitation)

- ◆ Tanyakan kepada pelanggan, pengguna, dan orang lain apa tujuan sistem atau produk, apa yang harus dicapai, bagaimana sistem atau produk cocok dengan kebutuhan bisnis, dan akhirnya, bagaimana sistem atau produk akan digunakan sehari-hari
- ◆ Untuk memahami tujuan bisnis.
 - ◆ Tujuan adalah tujuan jangka panjang yang harus dicapai oleh sistem atau produk.
 - ◆ Tujuan dapat berhubungan dengan masalah fungsional atau nonfungsional (misalnya, keandalan, keamanan, kegunaan)
- ◆ Tugas Anda adalah melibatkan stakeholder dan mendorong mereka untuk membagikan tujuan mereka dengan jujur.
- ◆ Setelah tujuan ditangkap, Anda membuat mekanisme prioritas dan membuat alasan desain untuk arsitektur potensial (yang memenuhi tujuan stakeholder)
- ◆ Agility merupakan aspek penting dari requirement engineering.:
 - ◆ Maksud dari elisitasi adalah untuk mentransfer ide dari stakeholder ke tim PL dengan lancar dan tanpa penundaan.
 - ◆ Sangat mungkin bahwa kebutuhan baru akan terus muncul saat pengembangan produk berulang terjadi

7.1.3 Elaborasi (Elaboration)

- ◆ Berfokus pada pengembangan model kebutuhan yang disempurnakan yang mengidentifikasi berbagai aspek fungsi PL, perilaku, dan informasi
- ◆ Didorong oleh pembuatan dan penyempurnaan skenario pengguna (user scenario) yang diperoleh selama elisitasi
- ◆ Skenario ini menggambarkan bagaimana pengguna akhir (dan aktor lain) akan berinteraksi dengan system
- ◆ Setiap user scenario diuraikan untuk mengekstrak kelas analisis—entitas domain bisnis yang terlihat oleh pengguna akhir
- ◆ Atribut dari setiap kelas analisis didefinisikan, dan layanan yang dibutuhkan oleh setiap kelas diidentifikasi.
- ◆ Hubungan dan kolaborasi antar kelas diidentifikasi.
- ◆ Elaborasi adalah hal yang baik, tetapi Anda perlu tahu kapan harus berhenti.
- ◆ Kuncinya adalah untuk menggambarkan problem dengan cara yang menetapkan dasar yang kuat untuk desain dan kemudian melanjutkan.
- ◆ Jangan terobsesi dengan detail yang tidak perlu

7.1.4 Negosiasi (Negotiation)

- ◆ Bukan hal yang aneh bagi pelanggan dan pengguna untuk meminta lebih dari yang dapat dicapai, mengingat sumber daya bisnis yang terbatas.
- ◆ Juga relatif umum bagi pelanggan atau pengguna yang berbeda untuk mengusulkan kebutuhan yang bertentangan, dengan alasan bahwa versi mereka "penting untuk kebutuhan khusus kami."



- ◆ Konflik-konflik ini perlu didamaikan melalui proses negosiasi.
- ◆ Pelanggan, pengguna, dan stakeholder lainnya diminta untuk membuat peringkat kebutuhan dan kemudian mendiskusikan konflik dalam prioritas.
- ◆ Seharusnya tidak ada pemenang dan tidak ada pecundang dalam negosiasi yang efektif.
- ◆ Kedua belah pihak menang, karena "kesepakatan" yang dapat dijalani keduanya telah dipadatkan.
- ◆ Anda harus menggunakan pendekatan berulang yang memprioritaskan kebutuhan, menilai biaya dan risikonya, dan mengatasi konflik internal.
- ◆ Dengan cara ini, kebutuhan dihilangkan, digabungkan, dan/atau dimodifikasi sehingga masing-masing pihak mencapai beberapa ukuran kepuasan

7.1.5 Spesifikasi (Specification)

Dalam konteks sistem berbasis komputer (dan perangkat lunak), istilah spesifikasi memiliki arti yang berbeda bagi orang yang berbeda.



Spesifikasi dapat berupa dokumen tertulis, sekumpulan model grafis, model matematika formal, kumpulan skenario penggunaan, prototipe, atau kombinasi dari semuanya.



Beberapa menyarankan bahwa "template standar" harus dikembangkan dan digunakan untuk spesifikasi, dengan alasan bahwa ini mengarah pada persyaratan yang disajikan secara konsisten dan karena itu lebih mudah dipahami.



Skenario penggunaan (usage scenario) mungkin adalah yang diperlukan untuk produk atau sistem yang lebih kecil yang berada dalam lingkungan teknis yang dipahami dengan baik

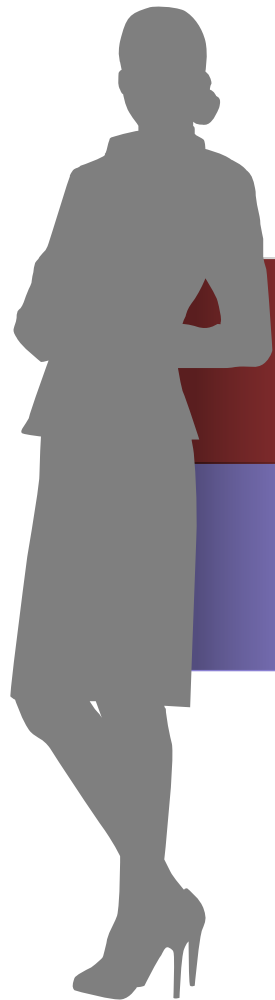


- Namun, terkadang perlu untuk tetap fleksibel ketika spesifikasi akan dikembangkan.
- Formalitas dan format spesifikasi bervariasi menurut ukuran dan kompleksitas PL yang akan dibangun.
 - Untuk sistem besar, dokumen tertulis, yang menggabungkan deskripsi bahasa alami dan model grafis, mungkin merupakan pendekatan terbaik.



Templat dokumen formal software requirements specification tersedia di https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc

7.1.6 Validasi (Validation)



Kualitas
Work Product

Technical
Review



KUALITAS WORK PRODUCT

- Pada validasi, produk kerja yang dihasilkan selama rekayasa kebutuhan **dinilai kualitasnya**.
- Perhatian utama selama validasi kebutuhan adalah **konsistensi**.
- Gunakan **model analisis** untuk memastikan bahwa kebutuhan telah dinyatakan **konsisten**:
 - Validasi kebutuhan **memeriksa spesifikasi** untuk memastikan bahwa semua kebutuhan PL telah dinyatakan dengan jelas;
 - bahwa inkonsistensi, kelalaian, dan kesalahan **telah dideteksi dan diperbaiki**;
 - bahwa produk kerja **sesuai dengan standar** yang ditetapkan untuk proses, proyek, dan produk.



TECHNICAL REVIEW

- **Mekanisme utama** validasi kebutuhan adalah tinjauan teknis (Technical review)
- **Tim review** termasuk **software engineer**, pelanggan (**customer**), pengguna (**end users**), dan **stakeholder** lain memvalidasi kebutuhan, yaitu:
 - **Memeriksa spesifikasi** mencari kesalahan dalam konten atau interpretasi,
 - area di mana **klarifikasi** mungkin diperlukan,
 - **informasi yang hilang**,
 - **inkonsistensi** (masalah utama ketika produk atau sistem besar direkayasa),
 - **kebutuhan yang saling bertentangan**, atau
 - **kebutuhan yang tidak realistis** (tidak dapat dicapai).

Ilustrasi beberapa masalah yang terjadi selama validasi persyaratan, perhatikan dua kebutuhan berikut yang tampaknya tidak berbahaya:



7.1.6 Manajemen Kebutuhan

- ❑ Kebutuhan untuk sistem berbasis komputer berubah, dan keinginan untuk mengubah kebutuhan tetap ada sepanjang umur sistem.
- ❑ Manajemen kebutuhan adalah serangkaian kegiatan yang membantu tim proyek mengidentifikasi, mengontrol, dan melacak kebutuhan dan perubahan kebutuhan setiap saat saat proyek berlangsung.



1

Apakah kebutuhan dinyatakan dengan jelas? Bisakah kebutuhan disalahartikan?

2

Apakah sumber (misalnya, seseorang, peraturan, dokumen) dari kebutuhan diidentifikasi?

Apakah pernyataan akhir dari kebutuhan telah diperiksa oleh atau terhadap sumber aslinya?

3

Apakah kebutuhan dibatasi dalam istilah kuantitatif?

4

Kebutuhan lain apa yang terkait dengan kebutuhan ini?

Apakah kebutuhan dicatat dengan jelas melalui matriks referensi silang atau mekanisme lain?

5

Apakah kebutuhan melanggar batasan domain sistem apa pun?

6

Apakah kebutuhan dapat diuji? Jika demikian, dapatkah kita menentukan pengujian (terkadang disebut kriteria validasi) untuk menjalankan kebutuhan?

7

Apakah kebutuhan dapat dilacak ke model sistem apa pun yang telah dibuat?

8

Apakah kebutuhan dapat dilacak ke keseluruhan sistem dan tujuan produk?

9

Apakah spesifikasi terstruktur dengan cara yang mengarah pada pemahaman yang mudah, referensi yang mudah, dan terjemahan yang mudah ke dalam produk kerja yang lebih teknis?

10

Apakah indeks untuk spesifikasi telah dibuat?

11

Apakah kebutuhan yang terkait dengan kinerja, perilaku, dan karakteristik operasional telah dinyatakan dengan jelas? Kebutuhan apa yang tampak implisit?

7.2 Establishing the Groundwork (Membangun Landasan)

- 7.2.1 Identifikasi Stakeholder
- 7.2.2 Mengenal Banyak Sudut Pandang
- 7.2.3 Bekerja menuju ke Kolaborasi
- 7.2.4 Mengajukan Pertanyaan Pertama
- 7.2.5 Kebutuhan Nonfungsional
- 7.2.6 Ketertelusuran (Traceability)

7.2 Establishing the Groundwork

Langkah-langkah yang diperlukan untuk membangun dasar untuk memahami kebutuhan PL— untuk memulai proyek dengan cara yang akan membuatnya terus bergerak maju menuju solusi yang sukses

- 7.2.1 Identifikasi Stakeholder
- 7.2.2 Mengenali Banyak Sudut Pandang
- 7.2.3 Bekerja menuju ke Kolaborasi
- 7.2.4 Mengajukan Pertanyaan Pertama
- 7.2.5 Kebutuhan Nonfungsional
- 7.2.6 Ketertelusuran (Traceability)

- ◆ Pada pengaturan yang ideal, stakeholder dan software engineer bekerja sama dalam satu tim
- ◆ Dalam kasus seperti itu, rekayasa kebutuhan hanyalah masalah melakukan percakapan yang bermakna dengan rekan kerja yang merupakan anggota tim yang dikenal baik.
- ◆ Tetapi kenyataan seringkali sangat berbeda.

- ◆ Pelanggan atau pengguna akhir mungkin:
 - ◆ tinggal di kota atau negara yang berbeda,
 - ◆ hanya memiliki gagasan yang kabur tentang apa yang diperlukan,
 - ◆ memiliki pendapat yang bertentangan tentang sistem yang akan dibangun,
 - ◆ memiliki pengetahuan teknis yang terbatas, dan
 - ◆ memiliki waktu yang terbatas untuk berinteraksi dengan perekrutan kebutuhan
- ◆ Tidak satu pun dari hal-hal ini yang diinginkan, tetapi semuanya umum, dan Anda sering dipaksa untuk bekerja dalam batasan yang dipaksakan oleh situasi ini.

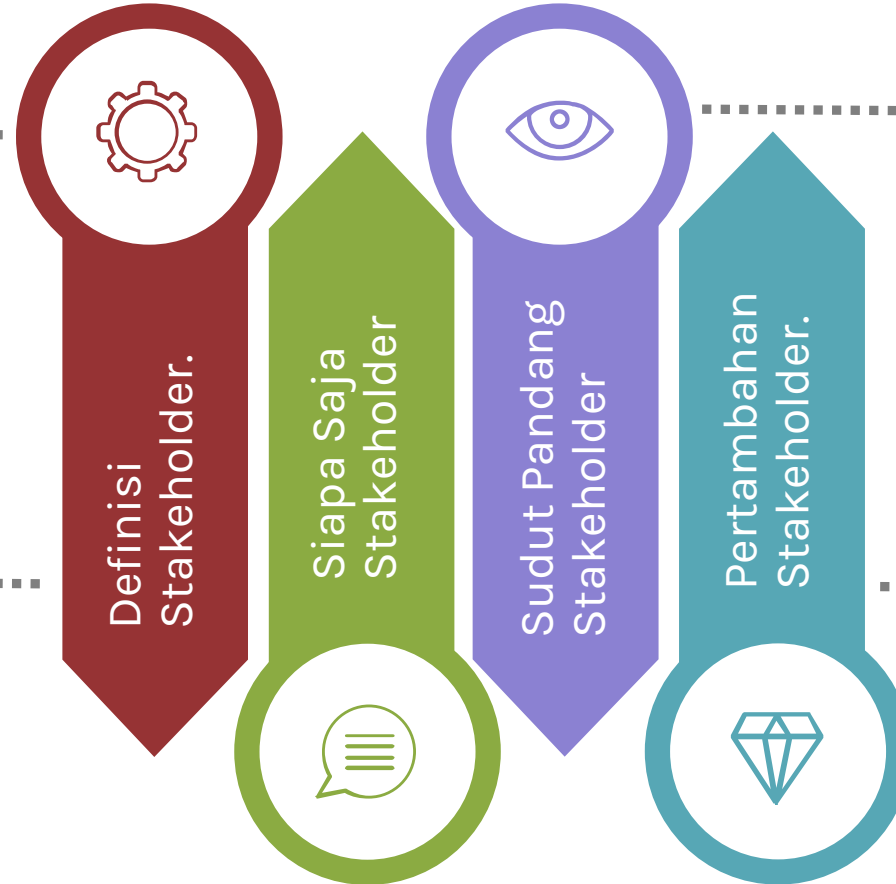
7.2.1 Identifikasi Stakeholder

Definisi

Stakeholder adalah “setiap orang yang memperoleh manfaat baik secara langsung maupun tidak langsung dari sistem yang sedang dikembangkan”.

Siapa Saja:

Manajer Operasi Bisnis, Manajer Produk, Orang Pemasaran, Pelanggan Internal dan Eksternal, Pengguna Akhir, Konsultan, Insinyur Produk, Software Engineer, Insinyur Dukungan dan Pemeliharaan, dan lain-lain



Sudut Pandang

Setiap stakeholder memiliki pandangan yang berbeda tentang sistem, mencapai manfaat yang berbeda ketika sistem berhasil dikembangkan, dan terbuka terhadap risiko yang berbeda jika upaya pengembangan gagal

Pertambahan Stakeholder

Daftar awal akan bertambah ketika para stakeholder dihubungi karena setiap stakeholder akan ditanyai: “Menurut Anda dengan siapa lagi saya harus berbicara?”.

7.2.2 Mengenali Banyak Sudut Pandang



Banyak Stakeholder Berbeda Kebutuhan

- Grup pemasaran tertarik pada fitur yang akan menggairahkan pasar potensial, membuat sistem baru mudah dijual
- Manajer bisnis tertarik pada serangkaian fitur yang dapat dibangun sesuai anggaran dan yang akan siap untuk memenuhi market window yang ditentukan
- Pengguna akhir menginginkan fitur yang familier dan mudah dipelajari serta digunakan
- Software engineer memperhatikan fungsi yang tidak terlihat oleh stakeholder nonteknis tetapi memungkinkan infrastruktur yang mendukung lebih banyak fungsi dan fitur yang dapat dipasarkan



Kontribusi Informasi Stakeholder

- Masing-masing konstituen (dan lainnya) akan menyumbangkan informasi untuk proses rekayasa kebutuhan.
- Karena informasi dari berbagai sudut pandang dikumpulkan, kebutuhan yang muncul mungkin tidak konsisten atau mungkin bertentangan satu sama lain.
- Anda harus mengkategorikan semua informasi stakeholder termasuk kebutuhan yang tidak konsisten dan bertentangan dengan cara yang memungkinkan pengambil keputusan untuk memilih serangkaian kebutuhan yang konsisten secara internal untuk sistem.



Tujuan Rekayasa Kebutuhan

- Beberapa hal dapat menyulitkan untuk memperoleh persyaratan perangkat lunak yang memuaskan penggunaannya:
 - tujuan proyek tidak jelas,
- prioritas pemangku kepentingan berbeda,
 - orang memiliki asumsi yang tidak diucapkan,
 - pemangku kepentingan menafsirkan makna secara berbeda, dan
- persyaratan dinyatakan dengan cara yang membuat mereka sulit untuk diverifikasi.
 - Tujuan dari rekayasa kebutuhan yang efektif adalah untuk menghilangkan atau setidaknya mengurangi masalah ini

7.2.3 Bekerja Menuju ke Kolaborasi



PLANNING POKER

- Gunakan skema “voting” berdasarkan poin prioritas
- Semua stakeholder diberikan sejumlah poin prioritas yang dapat “dibelanjakan” pada sejumlah kebutuhan
- Daftar kebutuhan disajikan, dan setiap stakeholder menunjukkan kepentingan relatif dari masing-masing (dari sudut pandangnya) dengan menggunakan satu atau lebih poin prioritas untuk itu.
- Poin yang dihabiskan tidak dapat digunakan kembali.
- Setelah poin prioritas stakeholder habis, tidak ada tindakan lebih lanjut tentang kebutuhan yang dapat diambil oleh orang tersebut.
- Poin keseluruhan yang dihabiskan untuk setiap kebutuhan oleh semua stakeholder memberikan indikasi pentingnya keseluruhan setiap kebutuhan



- Ketika 5 stakeholder terlibat dalam proyek PL, maka Anda mungkin memiliki lima (atau lebih) pendapat berbeda tentang kumpulan kebutuhannya yang tepat
- Pelanggan (dan stakeholder lainnya) harus berkolaborasi di antara mereka sendiri (menghindari pertempuran kecil-kecilan) dan dengan praktisi RPL jika ingin menghasilkan sistem yang sukses.
- Tapi bagaimana kolaborasi ini dicapai?



- Pekerjaan seorang perekayasa kebutuhan adalah untuk mengidentifikasi area kesamaan (yaitu, kebutuhan yang disetujui semua stakeholder) dan
- Area konflik atau inkonsistensi (yaitu, kebutuhan yang diinginkan oleh satu stakeholder tetapi bertentangan dengan kebutuhan stakeholder lain).
- Tentu saja, kategori terakhir inilah yang menghadirkan tantangan.



- Kolaborasi tidak berarti bahwa kebutuhan “ditentukan oleh komite.”
- Dalam banyak kasus, stakeholder berkolaborasi dengan memberikan pandangan mereka tentang kebutuhan, tetapi “project champion” yang kuat (mis., Manajer bisnis atau teknolog senior) dapat membuat keputusan akhir tentang kebutuhan mana yang tepat.

7.2.4 Mengajukan Pertanyaan Pertama



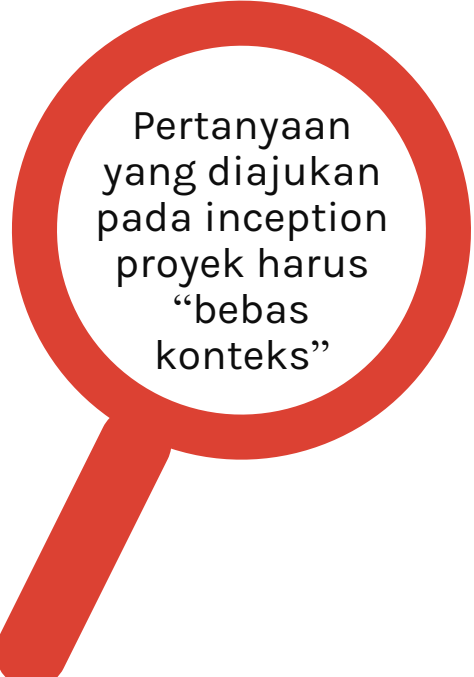
- Rangkaian pertanyaan bebas konteks pertama berfokus pada pelanggan dan stakeholder lainnya serta tujuan dan manfaat proyek secara keseluruhan:
 - Siapa di balik permintaan pekerjaan ini?
 - Siapa yang akan menggunakan solusinya?
 - Apa manfaat ekonomi dari solusi yang berhasil?
 - Apakah ada sumber lain untuk solusi yang Anda butuhkan?
- Pertanyaan-pertanyaan ini membantu mengidentifikasi semua stakeholder yang akan tertarik dengan PL yang akan dibangun.
- Selain itu, pertanyaan mengidentifikasi manfaat terukur dari implementasi yang sukses dan kemungkinan alternatif untuk pengembangan PL khusus



- Kumpulan pertanyaan berikutnya memungkinkan Anda untuk mendapatkan pemahaman yang lebih baik tentang masalah dan memungkinkan pelanggan untuk menyuarakan persepsinya tentang solusi:
 - Bagaimana Anda mencirikan keluaran "baik" yang akan dihasilkan oleh solusi yang sukses?
 - Problem apa yang akan ditangani oleh solusi ini?
 - Dapatkah Anda menunjukkan kepada saya (atau menjelaskan) lingkungan bisnis di mana solusi akan digunakan?
 - Apakah problem atau kendala kinerja khusus akan mempengaruhi cara pendekatan solusi?



- Rangkaian pertanyaan terakhir berfokus pada efektivitas aktivitas komunikasi itu sendiri.
- Gause dan Weinberg (1989)] menyebut ini "meta-questions" dan mengusulkan daftar (disingkat) berikut:
 - Apakah Anda orang yang tepat untuk menjawab pertanyaan-pertanyaan ini?
 - Apakah jawaban Anda "resmi"?
 - Apakah pertanyaan saya relevan dengan masalah yang Anda miliki?
 - Apakah saya terlalu banyak bertanya?
 - Adakah yang bisa memberikan informasi tambahan?
 - Haruskah saya menanyakan hal lain pada Anda?



Pertanyaan yang diajukan pada inception proyek harus “bebas konteks”

7.2.5 Kebutuhan Nonfungsional (Nonfunctional Requirements – NFR)



Definisi Kebutuhan Nonfungsional (NFR)

- NFR adalah atribut kualitas, atribut kinerja, atribut keamanan, atau batasan umum pada suatu sistem.
- Ini seringkali tidak mudah bagi para stakeholder untuk mengartikulasikannya.
- Chung [Chu09] menunjukkan bahwa ada penekanan yang berat sebelah pada fungsionalitas PL, namun PL mungkin tidak berguna atau dapat digunakan tanpa karakteristik nonfungsional yang diperlukan.



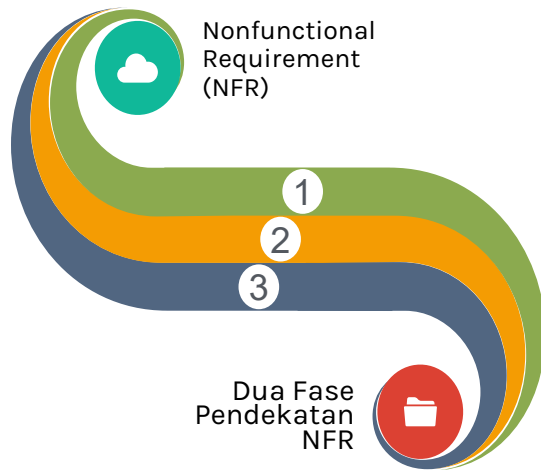
Fase Pertama identifikasi NFR

- Seperangkat pedoman RPL ditetapkan untuk sistem yang akan dibangun.
- Termasuk pedoman untuk praktik terbaik, membahas gaya arsitektural (Bab 10) dan penggunaan pola desain/design pattern (Bab 14).
- Daftar NFR (misalnya, kebutuhan yang membahas kegunaan, kemampuan pengujian, keamanan, atau pemeliharaan) kemudian dikembangkan.
- Tabel sederhana mencantumkan NFR sebagai label kolom dan pedoman RPL sebagai label baris.
- Matriks hubungan membandingkan setiap pedoman dengan pedoman lainnya, membantu tim menilai apakah setiap pasangan pedoman saling melengkapi, tumpang tindih, bertentangan, atau independen.



Fase Kedua identifikasi NFR

- Tim memprioritaskan setiap kebutuhan nonfungsional dengan membuat seperangkat NFR yang homogen menggunakan seperangkat aturan keputusan yang menetapkan pedoman mana yang harus diterapkan dan mana yang ditolak.

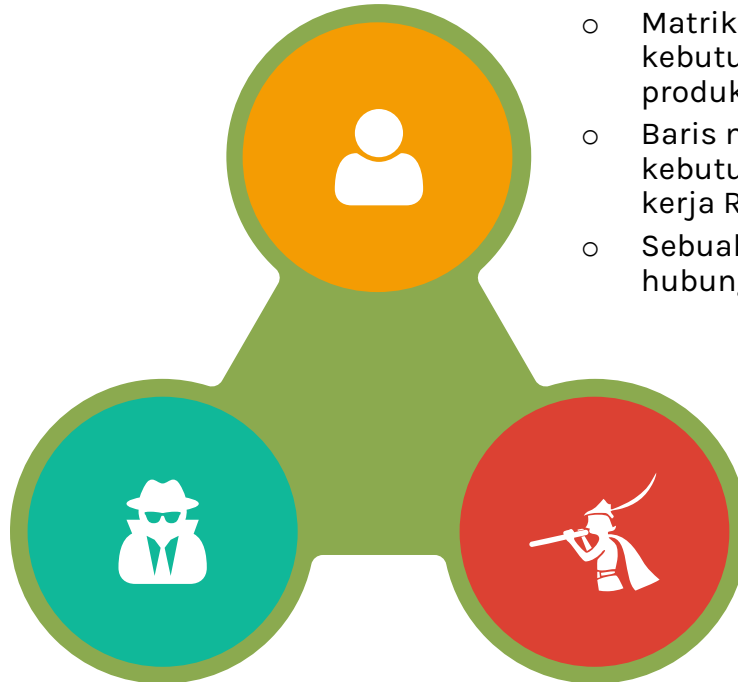


7.2.6 Ketertelusuran (Traceability)

Ketertelusuran adalah istilah RPL yang mengacu pada tautan terdokumentasi antara produk kerja (work product) RPL (misalnya, kebutuhan dan kasus uji).

Kebutuhan dan Work Product Berkembang

- Seiring bertambahnya jumlah kebutuhan dan jumlah produk kerja, maka semakin sulit untuk menjaga agar matriks ketertelusuran tetap mutakhir.
 - Meskipun demikian, penting untuk membuat beberapa cara untuk melacak dampak dan evolusi kebutuhan produk



Matriks Ketertelusuran

- Matriks ketertelusuran memungkinkan seorang perekrutan kebutuhan untuk mewakili hubungan antara kebutuhan dan produk kerja RPL lainnya.
- Baris matriks ketertelusuran diberi label menggunakan nama kebutuhan, dan kolom dapat diberi label dengan nama produk kerja RPL (misalnya, elemen desain atau kasus uji).
- Sebuah sel matriks ditandai untuk menunjukkan adanya hubungan antara keduanya.

Manfaat Matriks Ketertelusuran

- Matriks ketertelusuran dapat mendukung berbagai kegiatan pengembangan rekayasa.
- Matriks ini dapat memberikan kesinambungan bagi pengembang saat proyek bergerak dari satu fase proyek ke fase lainnya, terlepas dari model proses yang digunakan.
- Matriks ketertelusuran sering dapat digunakan untuk memastikan produk kerja rekayasa telah mempertimbangkan semua persyaratan



7.3 Requirements Gathering

7.3.1 Collaborative Requirements Gathering

7.3.2 Usage Scenarios

7.3.3 Elicitation Work Products

7.3 Requirements Gathering (Pengumpulan Kebutuhan)

7.3.1 Collaborative
Requirements Gathering

7.3.2 Usage Scenarios

7.3.3 Elicitation Work Products



- ◆ Pengumpulan persyaratan menggabungkan elemen pemecahan masalah, elaborasi, negosiasi, dan spesifikasi.
- ◆ Untuk mendorong pendekatan kolaboratif, berorientasi tim untuk pengumpulan persyaratan, pemangku kepentingan bekerja sama untuk mengidentifikasi masalah, mengusulkan elemen solusi, menegosiasikan pendekatan yang berbeda, dan menentukan serangkaian persyaratan solusi awal

7.3.1 Collaboration Requirements Gathering (Pengumpulan Kebutuhan Kolaboratif)

- Banyak pendekatan yang berbeda untuk pengumpulan kebutuhan kolaboratif telah diusulkan.
- Masing-masing menggunakan skenario yang sedikit berbeda, tetapi semuanya menerapkan beberapa variasi pada pedoman dasar berikut:
 - ✓ Rapat/meeting (face-to-face atau virtual) dilakukan dan dihadiri oleh software engineer stakeholder lainnya.
 - ✓ Aturan untuk persiapan dan partisipasi ditetapkan
 - ✓ Sebuah agenda disarankan yang cukup formal untuk mencakup semua poin penting tetapi cukup informal untuk mendorong aliran ide yang bebas.
 - ✓ Seorang “fasilitator” (dapat berupa pelanggan, pengembang, atau orang luar) mengontrol rapat.
 - ✓ Sebuah "mekanisme definisi" (bisa berupa lembar kerja, flip chart, atau stiker dinding atau papan buletin elektronik, ruang obrolan, atau forum virtual) digunakan.
- Tujuannya adalah untuk mengidentifikasi problem, mengusulkan elemen solusi, menegosiasikan pendekatan yang berbeda, dan menentukan serangkaian kebutuhan solusi awal



Satu atau dua halaman “Kebutuhan Produk” dibuat saat inception (Bagian 7.2)



- Tempat, waktu, dan tanggal pertemuan dipilih;
- Fasilitator dipilih; dan
- Peserta dari tim perangkat lunak dan organisasi pemangku kepentingan lainnya diundang untuk berpartisipasi



Jika suatu sistem atau produk akan melayani banyak pengguna, pastikan bahwa kebutuhan diperoleh dari perwakilan pengguna.



Jika hanya satu pengguna yang mendefinisikan semua kebutuhan, maka risiko penerimaan (acceptance) tinggi (artinya mungkin ada beberapa stakeholder lain yang tidak akan menerima produk).



Permintaan produk didistribusikan ke semua peserta sebelum tanggal meeting.

7.3.1 Collaboration Requirements Gathering (Pengumpulan Kebutuhan Kolaboratif)

➤ Contoh narasi permintaan produk home security function dari personel pemasaran yang terlibat dalam proyek SafeHome:

- Penelitian kami menunjukkan bahwa pasar untuk sistem manajemen rumah tumbuh pada tingkat 40 persen per tahun. Fungsi SafeHome pertama yang kami bawa ke pasar adalah fungsi keamanan rumah. Kebanyakan orang akrab dengan "sistem alarm", jadi ini akan menjadi penjualan yang mudah. Kami mungkin juga mempertimbangkan untuk menggunakan kontrol suara sistem menggunakan beberapa teknologi seperti Alexa.
- Fungsi keamanan rumah akan melindungi dan/atau mengenali berbagai "situasi" yang tidak diinginkan seperti masuk secara ilegal, kebakaran, banjir, kadar karbon monoksida, dan lain-lain. Ini akan menggunakan sensor nirkabel kami untuk mendeteksi setiap situasi, dapat diprogram oleh pemilik rumah, dan secara otomatis akan menghubungi agen pemantau dan ponsel pemilik ketika situasi terdeteksi.



Pada kenyataannya, orang lain akan berkontribusi pada narasi tersebut selama pertemuan pengumpulan kebutuhan dan lebih banyak informasi akan tersedia.



Tetapi bahkan dengan informasi tambahan, terdapat ambiguitas, kelalaian mungkin ada, dan kesalahan mungkin terjadi. Untuk saat ini, "deskripsi fungsional" sebelumnya sudah cukup.



Saat meninjau permintaan produk di hari-hari sebelum pertemuan, setiap peserta diminta untuk membuat:

- Daftar objek yang merupakan bagian dari lingkungan yang mengelilingi sistem,
- Objek lain yang akan diproduksi oleh sistem, dan
- Objek yang digunakan oleh sistem untuk menjalankan fungsinya.
- Daftar layanan lain (proses atau fungsi) yang memanipulasi atau berinteraksi dengan objek.
- Daftar kendala (misalnya, biaya, ukuran, aturan bisnis) dan
- Kriteria kinerja (misalnya, kecepatan, akurasi, keamanan) juga dikembangkan.



Para peserta diberitahu bahwa daftar tersebut tidak diharapkan lengkap tetapi diharapkan mencerminkan persepsi setiap orang tentang sistem.

7.3.1 Collaboration Requirements Gathering (Pengumpulan Kebutuhan Kolaboratif)

Object pada SafeHome:

- Control Panel
- Smoke Detector
- Sensor jendela dan pintu
 - Motion detector
 - Alarm
- An event (suatu sensor telah diaktivasi)
 - Display
 - Tablet
- Nomor telepon
- Telephone calls
- Dan lain lain

Performance Criteria pada SafeHome

- dikembangkan oleh setiap peserta meeting. Misalnya:
 - Sensor event harus dikenali dalam 1 detik
- Skema prioritas event harus diimplementasikan



Service pada SafeHome

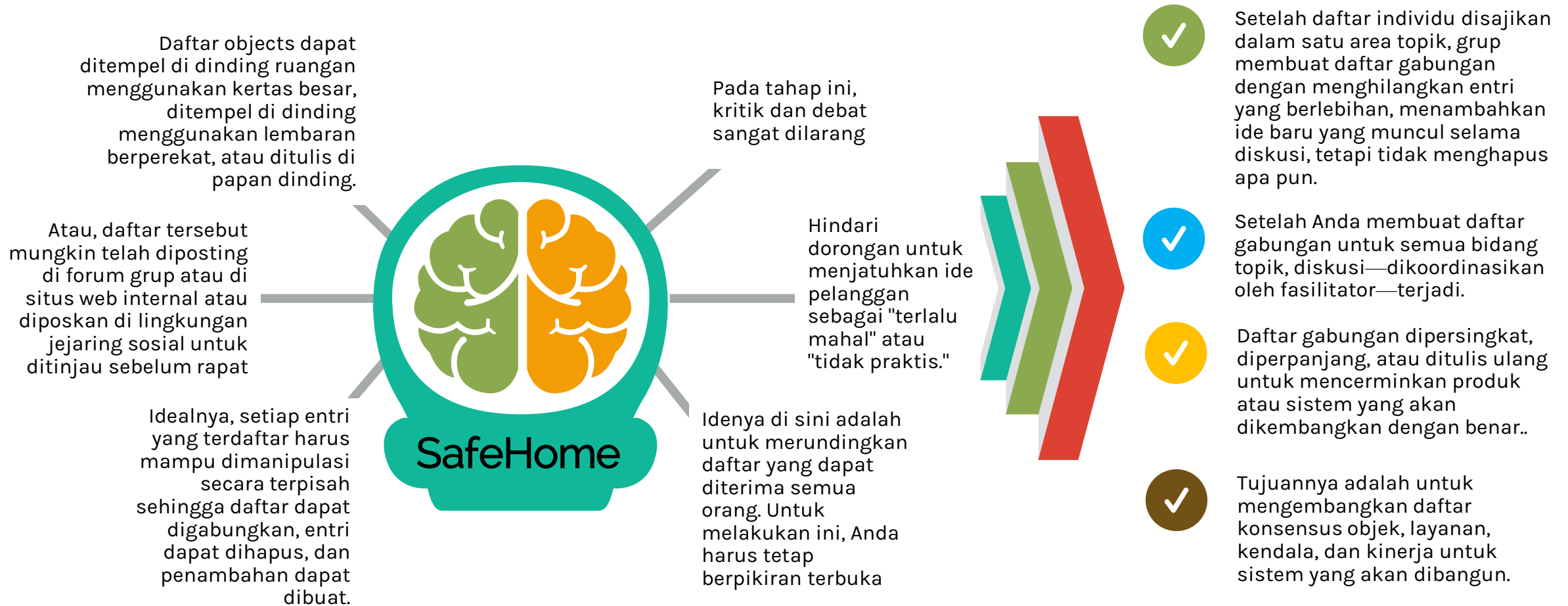
(perhatikan bahwa layanan bekerja pada objek):

- Mengonfigurasi sistem,
- Menyetel alarm,
- Memantau sensor,
- Memanggil telepon menggunakan router nirkabel,
- Memprogram panel kontrol,
- Membaca tampilan.

Constraints pada SafeHome

- dikembangkan oleh setiap peserta meeting. Misalnya:
 - Sistem harus mengenali ketika sensor tidak beroperasi,
 - Harus ramah pengguna
 - Harus berinteraksi langsung ke saluran telepon standar

7.3.1 Collaboration Requirements Gathering (Pengumpulan Kebutuhan Kolaboratif)



7.3.1 Collaboration Requirements Gathering (Pengumpulan Kebutuhan Kolaboratif)



Dalam banyak kasus, objek atau service yang dijelaskan dalam daftar akan memerlukan penjelasan lebih lanjut. Untuk mencapai ini, stakeholder mengembangkan spesifikasi mini (**Mini Specification**) untuk entri pada daftar atau dengan membuat kasus penggunaan yang melibatkan objek atau service

Mini Specification untuk object Control Panel SafeHome

- Control Panel adalah unit yang dipasang di dinding dengan ukuran kira-kira 230 × 130 mm.
- Control Panel memiliki konektivitas nirkabel ke sensor dan tablet.
- Interaksi pengguna terjadi melalui keypad yang berisi 12 tombol.
- Layar warna OLED 75 × 75 mm memberikan umpan balik pengguna.
- Perangkat lunak menyediakan petunjuk interaktif, gema, dan fungsi serupa

Dipresentasikan dan Dibahas dalam Meeting

- Spesifikasi mini dipresentasikan kepada semua pemangku kepentingan untuk didiskusikan.
- Penambahan, penghapusan, dan penjabaran lebih lanjut dilakukan.
- Dalam beberapa kasus, pengembangan spesifikasi mini akan mengungkap objek, layanan, kendala, atau persyaratan kinerja baru yang akan ditambahkan ke daftar asli.
- Selama semua diskusi, tim dapat mengangkat masalah yang tidak dapat diselesaikan selama rapat.
- Daftar masalah dipertahankan sehingga ide-ide ini akan ditindaklanjuti nanti.



Banyak kekhawatiran pemangku kepentingan (misalnya, akurasi, aksesibilitas data, keamanan) adalah dasar untuk kebutuhan sistem yang tidak berfungsi (Bagian 7.2).

- Saat pemangku kepentingan menyatakan keprihatinan ini, software engineer harus mempertimbangkannya dalam konteks sistem yang akan dibangun.
- Pertanyaan yang harus dijawab adalah:
 - Bisakah kita membangun sistem?
 - Akankah proses pengembangan ini memungkinkan kita untuk mengalahkan pesaing kita ke pasar?
 - Apakah tersedia sumber daya yang memadai untuk membangun dan memelihara sistem yang diusulkan?
 - Akankah kinerja sistem memenuhi kebutuhan pelanggan kami?

7.3.2 Usage Scenario

Saat persyaratan dikumpulkan, visi keseluruhan fungsi dan fitur sistem mulai terwujud.

Beralih ke Aktivitas lebih Teknis

Namun, sulit untuk beralih ke aktivitas RPL yang lebih teknis sampai Anda memahami bagaimana fitur akan digunakan oleh kelas pengguna akhir yang berbeda.

Skenario vs Use Case

Skenario, sering disebut use case, memberikan deskripsi tentang bagaimana sistem akan digunakan.



Skenario Penggunaan Sistem

Untuk mencapai ini, pengembang dan pengguna dapat membuat serangkaian skenario yang mengidentifikasi rangkaian penggunaan untuk sistem yang akan dibangun.

Penggunaan Use Case di bagian 7.4

Kasus penggunaan dibahas secara lebih rinci di Bagian 7.4.

SAFEHOME



Developing a Preliminary User Scenario

The scene: A meeting room, continuing the first requirements gathering meeting.

The players: Jamie Lazar, software team member; Vinod Raman, software team member; Ed Robbins, software team member; Doug Miller, software engineering manager; three members of marketing; a product engineering representative; and a facilitator.

The conversation:

Facilitator: We've been talking about security for access to *SafeHome* functionality that will be accessible via the Internet. I'd like to try something. Let's develop a usage scenario for access to the home security function.

Jamie: How?

Facilitator: We can do it a couple of different ways, but for now, I'd like to keep things really informal. Tell us (he points at a marketing person) how you envision accessing the system.

Marketing person: Um . . . well, this is the kind of thing I'd do if I was away from home and I

had to let someone into the house, say a housekeeper or repair guy, who didn't have the security code.

Facilitator (smiling): That's the reason you'd do it . . . tell me how you'd actually do this.

Marketing person: Um . . . the first thing I'd need is a PC. I'd log on to a website we'd maintain for all users of *SafeHome*. I'd provide my user ID and . . .

Vinod (interrupting): The Web page would have to be secure, encrypted, to guarantee that we're safe and . . .

Facilitator (interrupting): That's good information, Vinod, but it's technical. Let's just focus on how the end user will use this capability. OK?

Vinod: No problem.

Marketing person: So as I was saying, I'd log on to a website and provide my user ID and two levels of passwords.

Jamie: What if I forget my password?

Facilitator (interrupting): Good point, Jamie, but let's not address that now. We'll make a note of that and call it an *exception*. I'm sure there'll be others.

Marketing person: After I enter the passwords, a screen representing all *SafeHome* functions will appear. I'd select the home security function. The system might request that I verify who I am, say, by asking for my address or phone number or something. It would then display a picture of the security system control

panel along with a list of functions that I can perform—arm the system, disarm the system, disarm one or more sensors. I suppose it might also allow me to reconfigure security zones and other things like that, but I'm not sure.

(As the marketing person continues talking, Doug takes copious notes; these form the basis for the first informal usage scenario. Alternatively, the marketing person could have been asked to write the scenario, but this would be done outside the meeting.)

7.3.3 Elisitasi Work Products

- ❖ Produk kerja (work products) yang dihasilkan selama elisitasi kebutuhan akan bervariasi tergantung pada ukuran sistem atau produk yang akan dibangun.
- ❖ Masing-masing produk kerja ini ditinjau oleh semua orang yang telah berpartisipasi dalam elisitasi kebutuhan





7.4 Developing Use Cases

7.4 Developing Use Cases

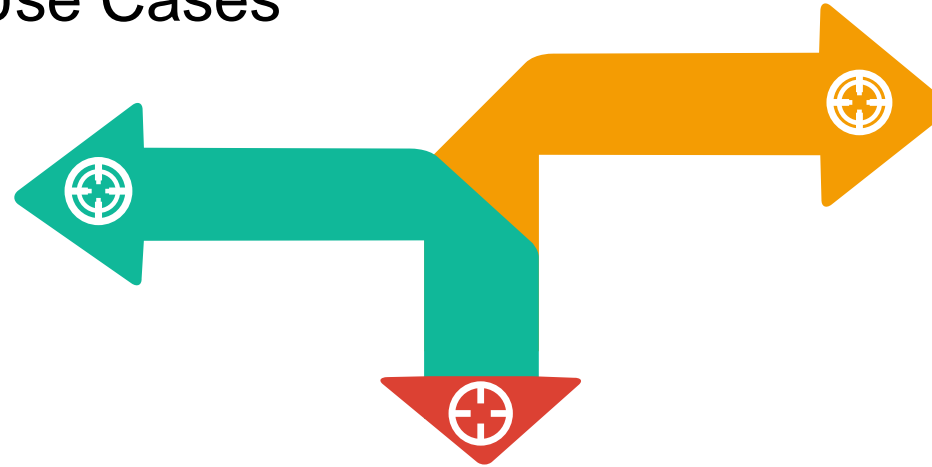
- **Use Case** menceritakan sebuah cerita bergaya tentang bagaimana pengguna akhir (memainkan salah satu dari beberapa kemungkinan peran) berinteraksi dengan sistem pada keadaan tertentu.
 - Teks naratif (cerita pengguna),
 - Garis besar tugas atau interaksi,
 - Deskripsi berbasis template, atau
 - Representasi diagram.
- Terlepas dari bentuknya, use case menggambarkan PL atau sistem dari sudut pandang pengguna akhir.



- ◆ Langkah pertama dalam menulis use case adalah **mendefinisikan set "aktor"** yang akan terlibat dalam cerita.
 - ◆ **Didefinisikan** secara lebih formal, aktor adalah segala sesuatu yang berkomunikasi dengan sistem atau produk dan yang berada di luar sistem itu sendiri.
 - ◆ Aktor adalah orang yang berbeda (atau perangkat) yang menggunakan sistem atau produk dalam konteks fungsi dan perilaku yang akan dijelaskan.
 - ◆ Aktor akan mewakili peran yang dimainkan orang (atau perangkat) saat sistem beroperasi.
 - ◆ Setiap aktor memiliki **satu atau lebih tujuan** saat menggunakan sistem,.
- ◆ Penting untuk dicatat bahwa **aktor dan pengguna akhir tidak selalu sama**.
 - ◆ Seorang pengguna biasa dapat memainkan beberapa peran berbeda saat menggunakan sistem,
 - ◆ Sedangkan aktor mewakili kelas entitas eksternal (seringkali, tetapi tidak selalu, orang) yang hanya memainkan satu peran dalam konteks use case.

7.4 Developing Use Cases

- Misalkan **pengguna** berinteraksi dengan program yang memungkinkan bereksperimen dengan **konfigurasi sensor alarm** di **gedung virtual**.
- Setelah meninjau kebutuhan dengan cermat, PL untuk **komputer kontrol** memerlukan **empat mode (peran)** yang berbeda untuk interaksi:
 - mode placement,
 - mode testing,
 - mode monitoring, dan
 - mode pemecahan masalah (troubleshooting).
- Jadi, diperlukan 4 aktor:
 - Editor
 - Tester
 - Monitor
 - Troubleshooter
- Dalam beberapa kasus, pengguna dapat memainkan semua peran.
- Di tempat lain, orang yang berbeda mungkin memainkan peran masing-masing aktor



- Karena elisitasi kebutuhan adalah aktivitas evolusioner, tidak semua aktor diidentifikasi selama iterasi pertama.
- Kemungkinan identifikasi aktor primer pada iterasi pertama dan aktor sekunder setelah sistem dipelajari lebih banyak.
 - Aktor primer berinteraksi untuk mencapai fungsi sistem yang diperlukan dan memperoleh manfaat yang diinginkan dari sistem.
 - Aktor primer bekerja secara langsung dan sering dengan perangkat lunak.
 - Aktor sekunder mendukung sistem sehingga aktor primer dapat melakukan pekerjaannya

- Setelah para aktor diidentifikasi, maka langkah selanjutnya ialah membuat Use Case
- pertanyaan yang harus dijawab oleh use case:
 - 1) Siapa aktor utama, aktor sekunder?
 - 2) Apa tujuan aktor?
 - 3) Prasyarat apa yang harus ada sebelum cerita/story dimulai?
 - 4) Apa tugas atau fungsi utama yang dilakukan oleh aktor?
 - 5) Pengecualian apa yang mungkin dipertimbangkan saat story dijelaskan?
 - 6) Variasi apa dalam interaksi aktor yang mungkin terjadi?
 - 7) Informasi sistem apa yang akan diperoleh, diproduksi, atau diubah oleh aktor?
 - 8) Apakah aktor harus menginformasikan sistem tentang perubahan lingkungan eksternal?
 - 9) Informasi apa yang diinginkan aktor dari sistem?
 - 10) Apakah aktor ingin diberitahu tentang perubahan yang tidak terduga?



7.4 Developing Use Cases

○ Empat **Aktor** SafeHome

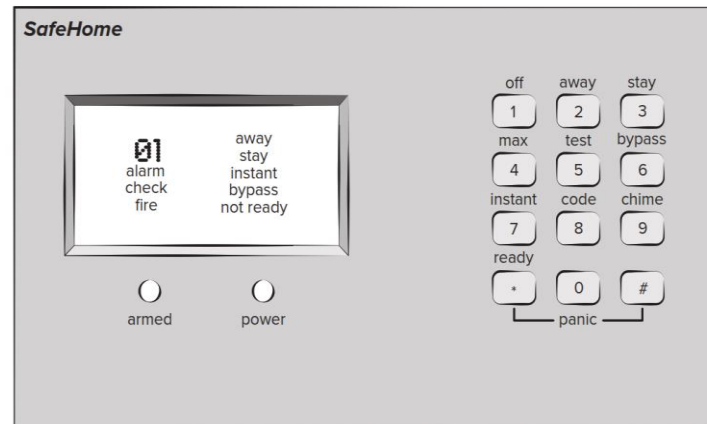
- **Homeowner** – seorang pengguna
- **Setup Manager** – orang yang sama dengan homeowner tetapi pada peran berbeda
- **Sensor** – perangkat (device) yang melekat pada sistem
- **Monitoring & Response Subsystem** - central station yang memonitor fungsi SafeHome Home Security



○ **Aktor Homeowner**

- Aktor Homeowner **berinteraksi** dengan **fungsi home security** dalam cara berbeda, yaitu **alarm control panel**, **tablet**, atau **ponsel**.
- Homeowner akan:
 - 1) Enter password agar bisa melakukan semua interaksi
 - 2) Menanyakan tentang status zona keamanan (security zone)
 - 3) Menanyakan tentang status sensor
 - 4) Menekan tombol panik (panic button) saat darurat (emergency)
 - 5) Aktivasi dan deaktivasi security system

SafeHome Control Panel



- Use case dasar menyajikan user story tingkat tinggi yang menggambarkan interaksi antara aktor dan sistem
- Dalam banyak kasus, use case dijabarkan lebih lanjut untuk memberikan lebih banyak detail tentang interaksi
- Semua interaksi use case pada actor Homeowner dikembangkan secara informal dengan user story atau dengan templat use case (next slide)
- Setiap use case harus direviu secara hati-hati. Jika elemen interaksi ambigu, maka biasanya mengindikasikan problem.
- Hal ini sangat penting untuk sistem di mana keselamatan atau keamanan pengguna menjadi perhatian stakeholder.

○ Use case dasar untuk aktivasi sistem pada Panel Kontrol:

- 1) Aktor Homeowner mencermati Panel Kontrol SafeHome untuk menentukan apakah sistem siap menerima input. Jika sistem tidak siap, maka muncul pesan **Not Ready** di layar display LCD.
- 2) Homeowner menggunakan **keypad** untuk mengetik **4-digit password**. Password dibandingkan dengan password valid pada sistem. Jika **password tidak benar** maka Kontrol Panel akan berbunyi **beep** satu kali dan **reset** sendiri untuk meminta input lanjutan. Jika input **password benar**, maka Kontrol Panel akan **menunggu aksi** berikutnya.
- 3) Homeowner memilih dan menekan tombol 'stay' atau 'away' untuk mengaktivasi sistem. **Stay** hanya mengaktivasi sensor perimeter (sensor pendeteksi gerakan di dalam dinonaktifkan). **Away** mengaktivasi seluruh sensor.
- 4) Ketika aktivasi terjadi, lampu merah alarm bisa dilihat oleh Homeowner

Use Case Detail

Use case : Inisiasi Monitoring

Primary Actor : Homeowner

Goal in context: Untuk mengatur sistem untuk memonitor sensor ketika Homeowner meninggalkan rumah atau tetap berada di dalam rumah

Preconditions : Sistem telah diprogram untuk password dan untuk mengenali berbagai sensor

Trigger : Homeowner memutuskan untuk “set” sistem, yaitu menyalakan fungsi-fungsi alarm

Scenario :

1. Homeowner mengamati Panel Kontrol
2. Homeowner enter password
3. Homeowner memilih “stay” atau “away”
4. Homeowner mengamati lampu alarm merah untuk menunjukkan bahwa SafeHome telah dipersenjatai (armed)

Exceptions:

1. Panel Kontrol *not ready*: Homeowner cek seluruh sensor untuk untuk menentukan mana yang terbuka dan kemudian menutupnya
2. Password tidak benar (Panel Kontrol berbunyi sekali): Homeowner enter ulang password yang benar
3. Password tidak dikenali: Monitoring & Response Subsystem harus dikontak untuk program ulang password
4. Stay dipilih: Panel Kontrol beep dua kali, dan lampu stay menyala; sensor perimeter diaktivasi
5. Away dipilih: Panel Kontrol beep tiga kali, dan lampu away menyala; seluruh sensor diaktivasi

Priority : Esensial, harus diimplementasikan

When available : Increment pertama

Frequency of Use : Per hari seringkali

Channel to Actor : Via interface Kontrol Panel

Secondary Actors : Support technician, sensors

Channel to Secondary Actors:

Support technician: phone line

Sensor: interface hardwire dan frekuensi radio

Open Issues:

1. Haruskah ada cara untuk mengaktifkan sistem tanpa menggunakan password atau dengan password yang disingkat?
2. Haruskah Panel Kontrol menampilkan pesan teks tambahan?
3. Berapa lama Homeowner harus memasukkan password sejak tombol pertama ditekan?
4. Apakah ada cara untuk menonaktifkan sistem sebelum benar-benar diaktifkan?



7.5 Building the Analysis Model

7.5.1 Elemen Model Analisis

7.5.2 Analysis Pattern (Pola Analisis)

7.5 Building the Analysis Model

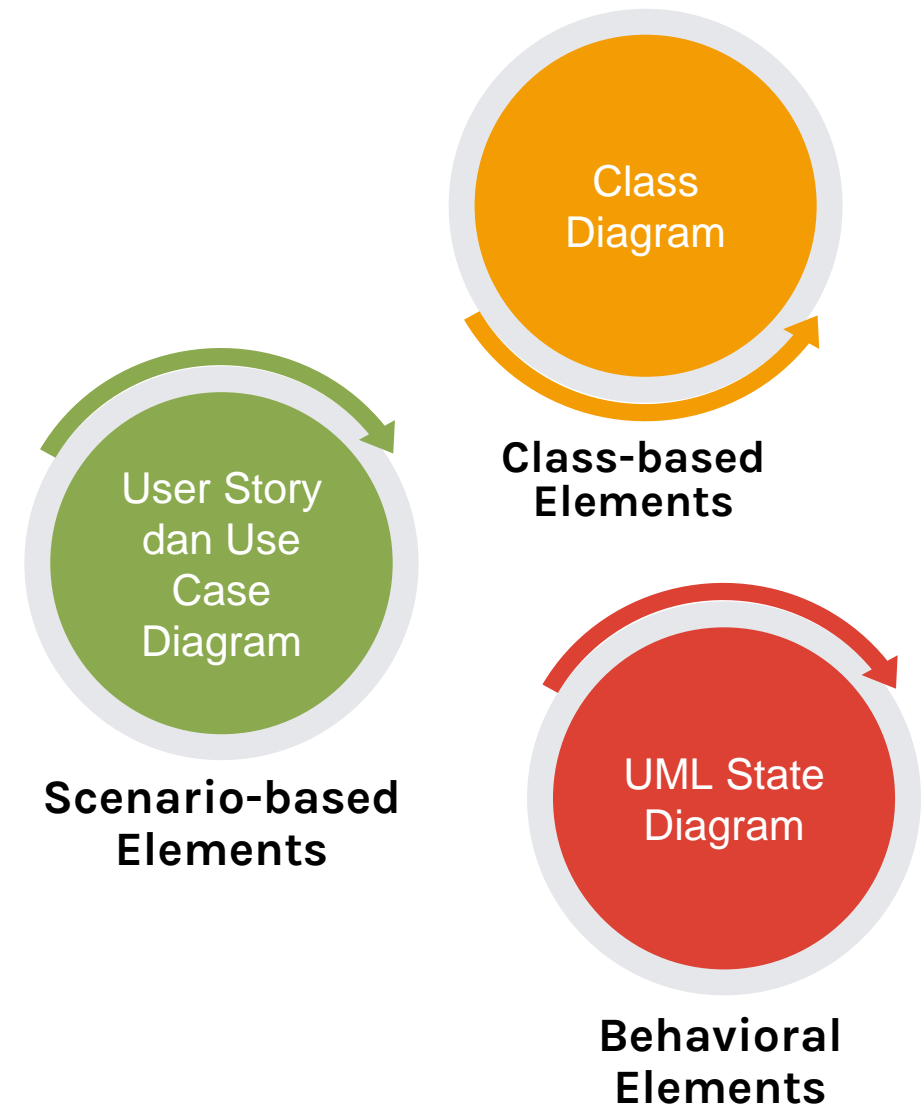
7.5.1 Elemen Model Analisis

7.5.2 Analysis Pattern (Pola Analysis)

- ◆ Maksud dari model analisis adalah untuk memberikan deskripsi tentang domain informasi, fungsional, dan perilaku yang diperlukan untuk sistem berbasis komputer.
- ◆ Model berubah secara dinamis saat Anda mempelajari lebih lanjut tentang sistem yang akan dibangun, dan saat stakeholder lebih memahami apa yang sebenarnya mereka butuhkan.
- ◆ Oleh karena itu, model analisis adalah gambaran kebutuhan pada waktu tertentu. Anda harus mengharapkannya untuk berubah.
- ◆ Saat model analisis berkembang, elemen tertentu akan menjadi relatif stabil, memberikan dasar yang kuat untuk tugas desain berikutnya.
- ◆ Namun, elemen model lainnya mungkin lebih fluktuatif, menunjukkan bahwa stakeholder belum sepenuhnya memahami persyaratan sistem.
- ◆ Jika tim Anda tidak menggunakan elemen tertentu dari model analisis saat proyek bergerak ke desain dan konstruksi, maka elemen tersebut tidak boleh dibuat di masa mendatang dan tidak boleh dipertahankan karena kebutuhan berubah dalam proyek saat ini.
- ◆ Model analisis dan metode yang digunakan untuk membangunnya disajikan secara rinci pada Bab 8.

7.5.1 Elemen Model Analisis

- Beberapa orang berpendapat bahwa yang terbaik adalah **memilih satu mode representasi** (misalnya, **use case**) dan menerapkannya dengan mengesampingkan semua mode lainnya.
- Praktisi lain percaya bahwa ada gunanya menggunakan **beberapa mode representasi** yang berbeda untuk menggambarkan model analisis.
- Menggunakan **mode representasi yang berbeda** memaksa Anda untuk mempertimbangkan kebutuhan dari sudut pandang yang berbeda — sebuah pendekatan yang **memiliki kemungkinan lebih tinggi untuk mengungkap kelalaian, inkonsistensi, dan ambiguitas**.
- Hal ini selalu merupakan ide yang baik untuk **melibatkan stakeholder**. Salah satu cara terbaik untuk melakukannya adalah **meminta setiap stakeholder menulis use case** yang menjelaskan bagaimana PL akan digunakan.

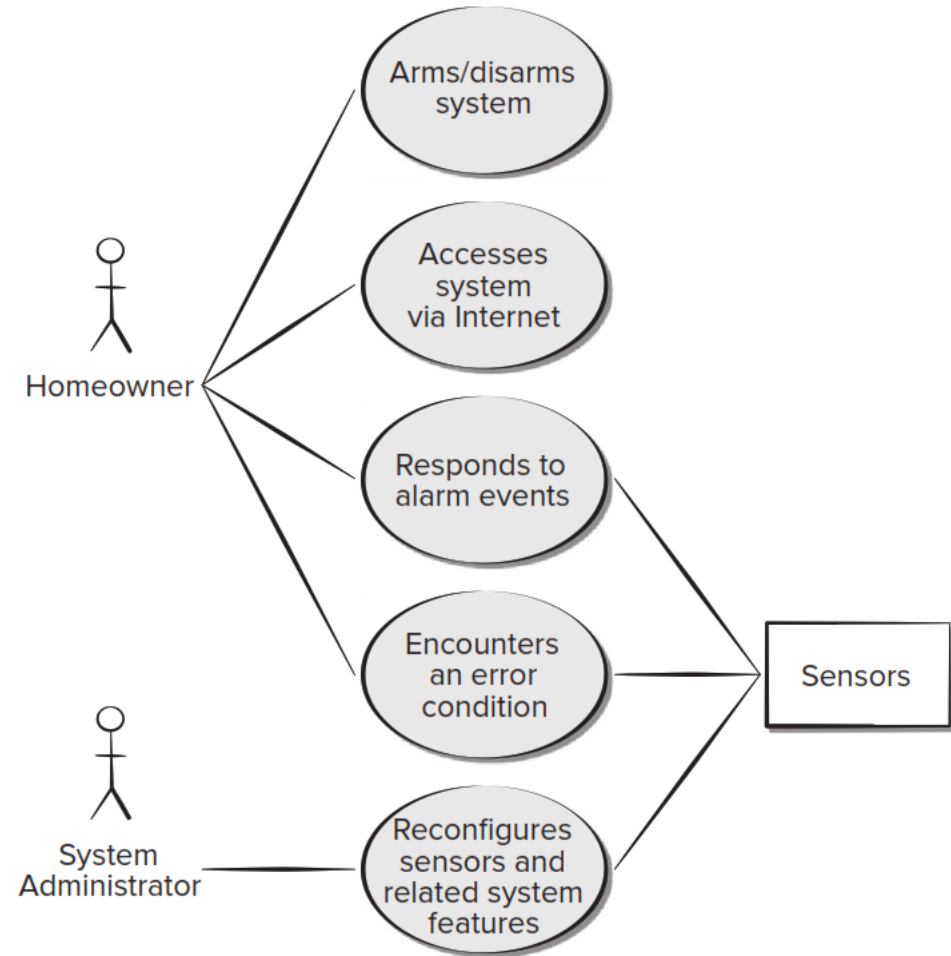


7.5.1 Elemen Model Analisis



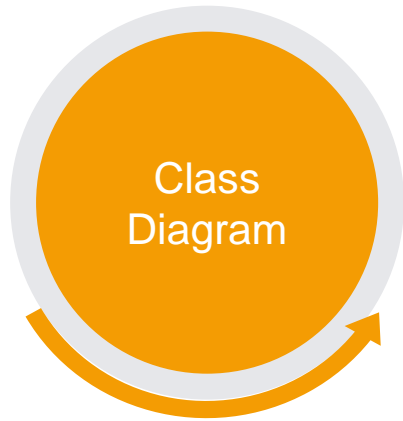
Scenario-based Elements

- o Elemen berbasis skenario pada model kebutuhan merupakan bagian dari model yang **pertama kali dikembangkan**.
- o Elemen berbasis skenario **menjelaskan sistem dari sudut pandang pengguna**
- o User story dan use case diagram yang sesuai dapat **dilaborasi menjadi use case berbasis templat**
- o Elemen berbasis skenario berfungsi **sebagai input untuk pembuatan elemen pemodelan lainnya**



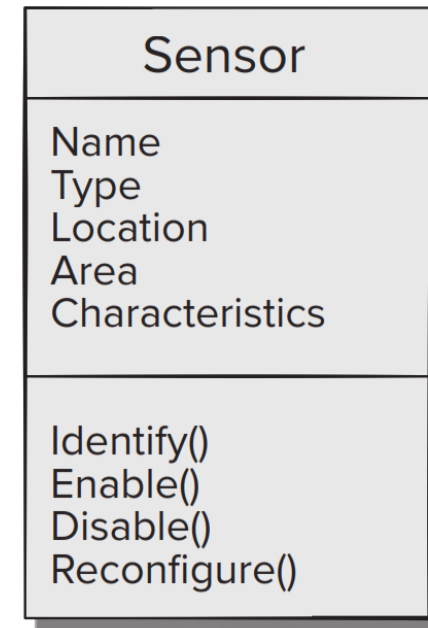
UML use case diagram untuk
SafeHome home security function

7.5.1 Elemen Model Analisis



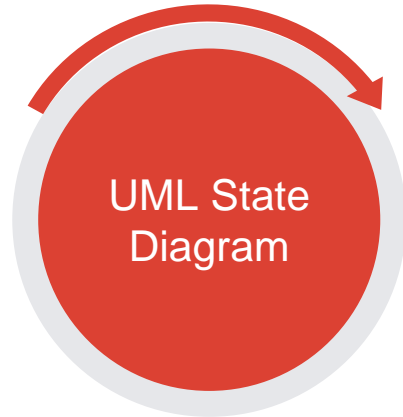
Class-based Elements

- Setiap usage scenario menyiratkan satu set objek yang dimanipulasi sebagai aktor berinteraksi dengan sistem
- Objek-objek ini dikategorikan ke dalam **class** (misalnya dengan diagram kelas UML) — kumpulan **things** yang memiliki atribut dan perilaku umum yang serupa
- Diagram mencantumkan daftar atribut dari sensor (nama, tipe) dan operations (*identify*, *enable*) yang diterapkan untuk memodifikasi atribut.
- **Class** berkolaborasi satu sama lain dan hubungan serta interaksi antar **class**
- Salah satu cara untuk mengisolasi **class** adalah dengan mencari **kata benda** deskriptif dalam skrip use case
- **Kata kerja** yang ditemukan dalam skrip kasus penggunaan dapat dianggap sebagai kandidat **method** untuk class ini



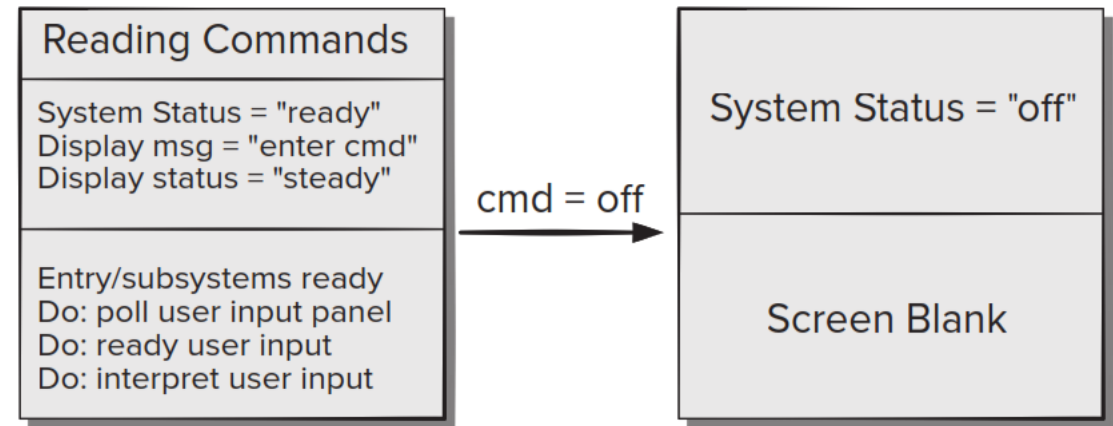
Class Diagram untuk Sensor

7.5.1 Elemen Model Analisis



Behavioral Elements

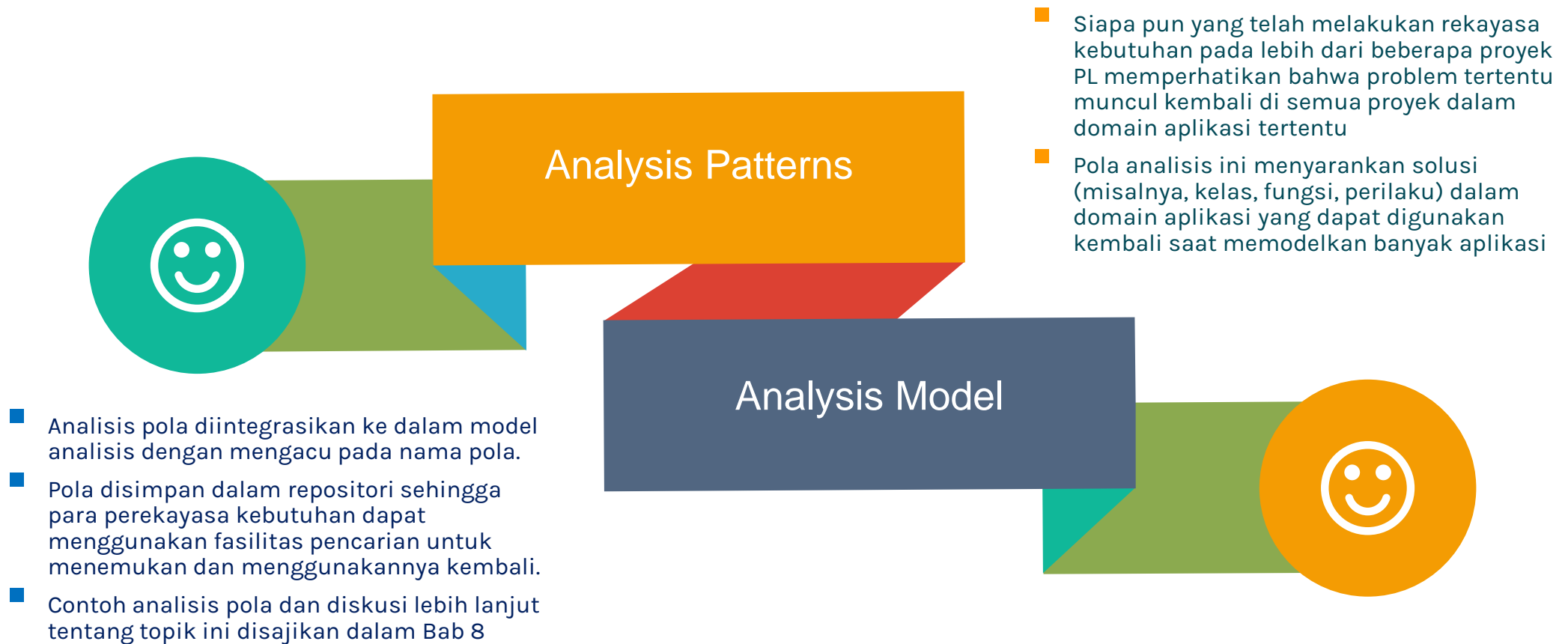
- Model kebutuhan harus menyediakan elemen pemodelan yang menggambarkan perilaku
- **State diagram** adalah salah satu **metode** untuk **merepresentasikan perilaku** suatu sistem dengan menggambarkan keadaannya (**state**) dan kejadian (**event**) yang menyebabkan sistem berubah state
- State adalah setiap mode perilaku yang dapat diamati secara eksternal.
- State diagram menunjukkan **action** apa (misalnya, aktivasi proses) yang diambil **ketika event terjadi**.
- Stimulus eksternal (event) menyebabkan transisi antar state.



UML State Diagram Notation

Ilustrasi penggunaan state diagram untuk PL yang tertanam (embedded software) di dalam panel kontrol SafeHome yang bertanggung jawab untuk membaca input pengguna

7.5.2 Analysis Pattern (Analisis Pola)





7.6 Negotiating Requirements

7.6 Negotiating Requirements (Negosiasi Kebutuhan)

- ◆ Task rekayasa kebutuhan (insepsi, elisitasim dan elaborasi) dilakukan untuk memperoleh detail kebutuhan pelanggan. Namun sayangnya sulit terpenuhi, sehingga diperlukan negosiasi.
- ◆ Negosiasi dilakukan terhadap 1 atau lebih stakeholder
 - ◆ Stakeholder diminta diminta untuk menyeimbangkan fungsionalitas, kinerja, dan karakteristik produk atau sistem lainnya terhadap biaya dan waktu pengiriman.

- ◆ Maksud dari negosiasi ini adalah untuk mengembangkan rencana proyek yang memenuhi kebutuhan stakeholder sementara pada saat yang sama mencerminkan kendala dunia nyata (misalnya, waktu, orang, anggaran) yang terdapat pada tim PL
- ◆ Negosiasi yang baik memberikan hasil “win-win”
 - ◆ Stakeholder “win” dengan mendapatkan sistem atau produk yang paling memenuhi kebutuhan mereka,
 - ◆ Pengembang “win” dengan bekerja untuk anggaran dan tenggat waktu yang realistis dan dapat dicapai

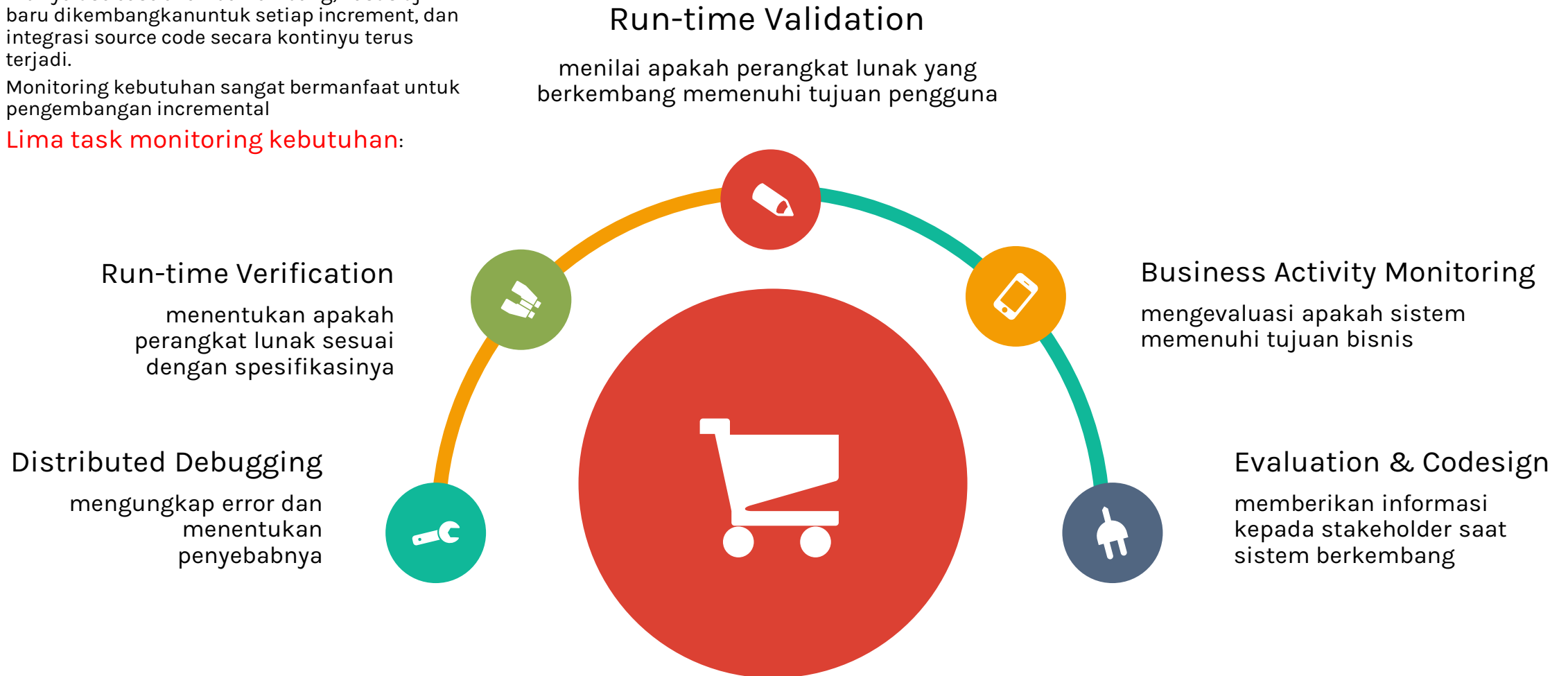
- ◆ ‘Handshaking’ merupakan salah satu cara untuk mendapatkan hasil “win-win”
 - ◆ tim perangkat lunak mengusulkan solusi untuk kebutuhan, menjelaskan dampaknya, dan mengomunikasikan niat mereka kepada perwakilan pelanggan
 - ◆ Perwakilan pelanggan meninjau solusi yang diusulkan, dengan fokus pada fitur yang hilang dan mencari klarifikasi kebutuhan baru
- ◆ Persyaratan ditentukan cukup baik jika pelanggan menerima solusi yang diusulkan.
- ◆ ‘Handshaking’ cenderung meningkatkan identifikasi, analisis, dan pemilihan varian dan mendorong negosiasi ‘win-win’



7.7 Requirements Monitoring

7.7 Requirements Monitoring (Monitoring Kebutuhan)

- Pengembangan incremental umum dilakukan. Artinya use case akan berkembang, kasus uji baru dikembangkan untuk setiap increment, dan integrasi source code secara kontinyu terus terjadi.
- Monitoring kebutuhan sangat bermanfaat untuk pengembangan incremental
- Lima task monitoring kebutuhan:





7.8 Validating Requirements

7.8 Validating Requirements (Validasi Kebutuhan)

- ◆ Ketika setiap elemen dari model kebutuhan dibuat, dilakukan pemeriksaan ketidakkonsistenan, kelalaian, dan ambiguitas.
- ◆ Pemeriksaan dilakukan juga untuk model proses agile di mana kebutuhan cenderung ditulis sebagai user story dan/atau kasus uji.
- ◆ Kebutuhan yang diwakili oleh model diprioritaskan oleh stakeholder dan dikelompokkan dalam paket kebutuhan yang akan diimplementasikan sebagai increment perangkat lunak.

Reviu model kebutuhan menjawab pertanyaan-pertanyaan berikut:

- 1) Apakah setiap kebutuhan konsisten dengan tujuan keseluruhan untuk sistem atau produk?
- 2) Apakah semua kebutuhan telah ditentukan pada tingkat abstraksi yang tepat? Artinya, apakah beberapa kebutuhan memberikan tingkat detail teknis yang tidak sesuai pada tahap ini?
- 3) Apakah kebutuhan benar-benar diperlukan atau apakah itu mewakili fitur tambahan yang mungkin tidak penting untuk tujuan sistem?
- 4) Apakah setiap kebutuhan dibatasi dan tidak ambigu?
- 5) Apakah setiap kebutuhan memiliki atribusi? Artinya, apakah sumber (umumnya, individu tertentu) dicatat untuk setiap kebutuhan?

- 6) Apakah ada kebutuhan yang bertentangan dengan kebutuhan lain?
- 7) Apakah setiap kebutuhan dapat dicapai dalam lingkungan teknis yang akan menampung sistem atau produk?
- 8) Apakah setiap kebutuhan dapat diuji, setelah diterapkan?
- 9) Apakah model kebutuhan benar mencerminkan informasi, fungsi, dan perilaku sistem yang akan dibangun?
- 10) Apakah model kebutuhan telah "dipartisi" dengan cara yang secara progresif memperlihatkan informasi yang lebih rinci tentang sistem?
- 11) Apakah pola kebutuhan telah digunakan untuk menyederhanakan model kebutuhan? Apakah semua pola telah divalidasi dengan benar? Apakah semua pola konsisten dengan kebutuhan pelanggan?



7.9 Rangkuman

7.9 Rangkuman

TASK REKAYASA KEBUTUHAN

- ◆ Dilakukan agar terbentuk fondasi kokoh untuk desain dan konstruksi
- ◆ Dilakukan saat aktivitas komunikasi dan aktivitas pemodelan yang telah ditetapkan untuk proses PL generic
- ◆ Tujuh aktivitas rekayasa kebutuhan—inception, elisitasi, elaborasi, negosiasi, spesifikasi, validasi, dan manajemen—dilakukan oleh anggota tim PL dan stakeholder produk



INSEPSI & ELISITASI

- ◆ Stakeholder menetapkan kebutuhan masalah dasar, menentukan kendala proyek utama, dan mengatasi fitur dan fungsi utama yang harus ada agar sistem dapat memenuhi tujuannya.
- ◆ Informasi ini disempurnakan dan diperluas selama elisitasi—aktivitas pengumpulan persyaratan yang memanfaatkan pertemuan yang difasilitasi dan pengembangan skenario penggunaan (cerita pengguna)

Rangkuman...

ELABORASI

- ◆ Elaborasi memperluas kebutuhan dalam model kumpulan elemen berbasis skenario, berbasis aktivitas, berbasis class, dan perilaku.
- ◆ Model dapat mereferensikan pola analisis, karakteristik domain problem yang telah terlihat muncul kembali di berbagai aplikasi.



NEGOSIASI & VALIDASI

- ◆ Saat kebutuhan telah diidentifikasi dan model persyaratan dibuat, tim PL dan stakeholder proyek lainnya menegosiasikan prioritas, ketersediaan, dan biaya relatif dari setiap kebutuhan.
- ◆ Maksud dari negosiasi ini adalah untuk mengembangkan rencana proyek yang realistis.
- ◆ Setiap kebutuhan perlu divalidasi terhadap kebutuhan pelanggan untuk memastikan bahwa sistem yang tepat akan dibangun



Pertemuan 7: Understanding Requirements

Meuthia Rachmaniah
Departemen Ilmu Komputer, FMIPA IPB

Software Engineering: A Practitioner's Approach, 9th Ed.
Roger S. Pressman dan Bruce R. Maxim
Copyright © 2020 McGraw-Hill Education