Teori & Problem

RPL

(Rekayasa Perangkat Lunak)

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

Teori & Problem RPL (Rekayasa Perangkat Lunak)

Penulis :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom

ISBN : 978-623-8642-15-1 (PDF)

Editor :
Muhammad Sholikan, M.Kom

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Anggota IKAPI No: 279 / ALB / JTE / 2023

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. 08122925000
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. 08122925000
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Rekayasa Perangkat Lunak tidak hanya memerlukan pemahaman tentang teknik dan terminologi, tetapi juga melibatkan penguasaan berbagai teknik yang harus dikuasai oleh mahasiswa. Buku ini ditujukan bagi mahasiswa tingkat sarjana yang mengambil mata kuliah rekayasa perangkat lunak. Dalam buku ini penulis menyadari pentingnya menyediakan contoh-contoh yang lengkap dan panduan yang memadai untuk membantu siswa memahami teknik-teknik ini. Buku ini dirancang untuk digunakan bersama dengan buku teks atau catatan kuliah yang ada mengenai rekayasa perangkat lunak. Latar belakang, motivasi, diagram, notasi, dan teknik tidak hanya dijelaskan tetapi juga diberikan aturan yang jelas mengenai cara membangun diagram dengan benar. Instruksi tentang penerapan teknik juga disertakan, dan yang tidak kalah pentingnya, terdapat contoh-contoh dan masalah yang dipecahkan untuk membantu siswa memahami lebih dalam tentang diagram, notasi, dan teknik yang diajarkan.

Dalam bab 1 buku ini akan membahas siklus hidup perangkat lunak dijelaskan sebagai urutan aktivitas yang berbeda selama proses pengembangan perangkat lunak. Setiap aktivitas dalam siklus ini erat berkaitan dengan kiriman-kiriman yang dihasilkan, seperti kode sumber atau panduan pengguna. Milestones atau pencapaian-pencapaian tertentu, seperti penyelesaian panduan pengguna, sangat penting karena mereka menandai peristiwa penting dalam proyek yang memungkinkan manajer untuk menilai kemajuan secara sistematis. Hal ini membantu dalam manajemen proyek dengan memungkinkan evaluasi yang akurat terhadap status proyek, memastikan proyek berjalan sesuai jadwal dan memungkinkan identifikasi masalah yang memerlukan perhatian lebih lanjut. Selanjutnya bab 2 membahas model proses perangkat lunak (SPM) yang digunakan untuk mengatur aktivitas dalam pengembangan perangkat lunak. Salah satu alat yang diperkenalkan adalah diagram aliran data, yang menunjukkan bagaimana data mengalir di antara komponen-komponen seperti tugas, komponen perangkat lunak, atau abstraksi fungsi dalam sistem. Selain itu, bab juga memperkenalkan model Petri Net, sebuah model matematis yang menggunakan node kondisi, busur, dan node peristiwa untuk menggambarkan perubahan keadaan dalam sistem perangkat lunak. Dengan memahami dan menggunakan model-model ini, pengembang dapat merencanakan dan mengelola proses pengembangan perangkat lunak secara lebih efisien.

Bab 3 ini mengenalkan konsep manajemen proyek perangkat lunak yang krusial dalam merencanakan, mengarahkan, memotivasi, dan mengkoordinasikan tim profesional untuk mencapai tujuan pengembangan perangkat lunak. Manajemen proyek ini melibatkan penerapan konsep-konsep umum dalam manajemen, tetapi juga menghadapi tantangan unik terkait dengan pengembangan perangkat lunak. Salah satu perhatian utamanya adalah visibilitas proyek, yaitu kemampuan untuk melihat dan memahami kemajuan serta status proyek secara jelas dan tepat waktu. Selanjutnya bab 4 ini menguraikan pentingnya perencanaan dalam pengembangan perangkat lunak. Perencanaan proyek perangkat lunak mencakup identifikasi tugas-tugas yang harus dilakukan, metode pelaksanaannya, dan sumber daya yang dibutuhkan untuk menyelesaikan setiap tugas. Salah satu alat yang digunakan dalam perencanaan ini adalah Struktur Rincian Kerja (WBS), yang memecah tugas-tugas besar menjadi tugas-tugas yang lebih kecil dan terukur. WBS direpresentasikan dalam bentuk struktur pohon, dengan tingkat atasnya sesuai dengan model siklus hidup (LCM) yang digunakan dalam organisasi. Tingkat selanjutnya mempartisi tugas-tugas menjadi unit yang

lebih terkelola, memfasilitasi pengawasan dan pengendalian yang lebih baik dalam pengembangan perangkat lunak.

Dalam bab ke 5 membahas pentingnya pengukuran dalam pengembangan perangkat lunak, yang melibatkan pemetaan simbol ke objek untuk mengukur atribut seperti ukuran proyek dan upaya pengembangan. Bab ini menekankan validasi metrik untuk memastikan relevansi dan kegunaan metrik dalam penggunaannya. Kriteria untuk metrik yang valid termasuk kemampuan untuk membedakan entitas yang berbeda, mematuhi kondisi representasi, kontribusi setara dari setiap unit atribut, dan kemungkinan nilai atribut yang sama untuk entitas yang berbeda.

Bab 6 akan menjelaskan risiko sebagai kemungkinan terjadinya peristiwa tidak diinginkan dalam manajemen proyek. Pendekatan proaktif digunakan untuk meminimalkan dampak negatif dari risiko yang terjadi. Terdapat perbedaan pendapat mengenai risiko yang harus dikelola, di mana beberapa ahli menyarankan inklusi risiko unik proyek, sementara yang lain menganggap risiko umum sebagai bagian dari perencanaan proyek standar. Pada bagian "Identifikasi Risiko" menjelaskan proses mengidentifikasi risiko berdasarkan pengaruhnya terhadap rencana proyek (risiko proyek), kualitas (risiko teknis), atau kelayakan produk (risiko bisnis). Beberapa ahli mengesampingkan risiko umum yang terjadi pada semua proyek dari perhatian manajemen risiko, menganggapnya sebagai bagian dari rutinitas perencanaan proyek. Dan dalam bab ke 7 memperkenalkan konsep kualitas dalam pengembangan perangkat lunak. Kualitas didefinisikan sebagai fitur dan karakteristik yang mempengaruhi kemampuan produk untuk memenuhi kebutuhan pengguna. Kualitas perangkat lunak juga diukur dari kemampuannya untuk berfungsi sesuai yang diharapkan. Teknik utama untuk mencapai kualitas adalah melalui tinjauan atau penelusuran perangkat lunak, dengan inspeksi formal sebagai metode terstruktur di mana desainer menyajikan desain kepada sekelompok rekan untuk mengevaluasi aspek teknisnya dan menemukan kesalahan. Metrik utama dalam inspeksi meliputi jumlah kesalahan per ribu baris kode (KLOC) dan efisiensi dalam menemukan kesalahan per jam yang dihabiskan.

Bab 8 "Persyaratan" bertujuan untuk mengumpulkan persyaratan dari pengguna dalam pengembangan perangkat lunak. Proses ini melibatkan pengembangan diagram dan spesifikasi kebutuhan setelah diskusi dengan pengguna. Diagram dan spesifikasi tersebut dievaluasi oleh pengguna untuk memastikan pengembang memahami persyaratan dengan baik. Komunikasi kembali aspek penting dari perangkat lunak yang akan dikembangkan kepada pengguna adalah fokus utama dalam fase ini. Bab 9 "Desain Perangkat Lunak" menjelaskan bahwa desain adalah proses menerapkan teknik dan prinsip untuk mendefinisikan perangkat, proses, atau sistem dengan detail yang cukup untuk implementasi fisiknya. Ini adalah tahap kreatif dalam pengembangan perangkat lunak, di mana tujuannya adalah mengubah persyaratan dari "apa" menjadi "bagaimana". Hasilnya adalah dokumen yang menyediakan detail implementasi sistem tanpa memerlukan interaksi tambahan dengan pembuat spesifikasi atau pengguna. Desain juga melibatkan perubahan terminologi dari ruang masalah persyaratan menjadi ruang solusi implementasi, membedakan antara Objek Analisis Berorientasi Objek (OOA) untuk ruang masalah/domain, dan Objek Desain Berorientasi Objek (OOD) untuk ruang solusi/implementasi.

Bab 10 "Pengujian Perangkat Lunak" menjelaskan bahwa pengujian perangkat lunak melibatkan eksekusi perangkat lunak dengan menggunakan data pengujian yang sebenarnya.

Proses ini kadang-kadang disebut sebagai pengujian perangkat lunak dinamis, yang berbeda dengan analisis statis yang melibatkan analisis kode sumber untuk mengidentifikasi masalah. Meskipun teknik analisis lainnya penting untuk validasi perangkat lunak, pengujian dengan menggunakan data pengujian yang nyata merupakan langkah krusial dalam memastikan kualitas dan kinerja perangkat lunak yang dikembangkan. Bab 11 "Pembangunan Berorientasi Objek" mengenalkan pendekatan pengembangan perangkat lunak yang berbeda dari metode konvensional. Hal ini dicapai dengan memodelkan domain masalah menggunakan objek yang merepresentasikan entitas penting, mengintegrasikan fungsi dan data dalam objek, mengadopsi prinsip reuse objek di dalam proyek dan antar proyek, serta menciptakan solusi yang lebih terhubung secara intelektual dengan masalah yang dihadapi. Unified Modeling Language (UML) digunakan sebagai notasi standar untuk menggambarkan model berorientasi objek. UML menyediakan spesifikasi yang dapat diakses melalui Web.

Dalam bab 12 membahas mengenai pentingnya pengukuran dalam konteks perangkat lunak berorientasi objek. Tujuan pengukuran perangkat lunak berorientasi objek mirip dengan pengukuran perangkat lunak konvensional, yaitu untuk memahami karakteristiknya. Perangkat lunak berorientasi objek menggunakan konsep enkapsulasi, pewarisan, dan pengikatan dinamis untuk menciptakan model yang lebih terhubung dengan dunia nyata. Penggunaan metrik tradisional seperti grafik aliran kontrol terlihat kurang relevan dalam konteks ini, karena kompleksitasnya sering kali terletak pada pola pemanggilan antar metode. Abstraksi dalam pengembangan perangkat lunak berorientasi objek, seperti yang digunakan dalam Unified Modeling Language (UML), juga menjadi fokus dalam upaya untuk mengembangkan metrik yang lebih baik dan mencapai konsensus mengenai abstraksi yang paling penting untuk diperhatikan. Selanjutnya dalam bab 13 membahas mengenai tantangan dan pendekatan dalam pengujian perangkat lunak berorientasi objek dibandingkan dengan perangkat lunak konvensional. Pengujian fungsional pada dasarnya tetap sama antara keduanya, tetapi pengujian struktural menghadapi kompleksitas baru karena struktur perangkat lunak berorientasi objek tidak dapat direpresentasikan dengan baik oleh diagram alur kendali. Hal ini memunculkan kebutuhan untuk mengembangkan metode pengujian yang lebih sesuai dengan model objek, seperti pengujian MM yang mencakup semua panggilan ke metode. dalam bab ke 14 membahas Notasi formal dalam pengembangan perangkat lunak menggunakan dasar matematis untuk menghindari ambiguitas yang umum terjadi dalam bahasa alami. Meskipun memiliki potensi besar untuk mengurangi kesalahan, penggunaannya masih terbatas karena kesulitan dalam pembangunan dan implementasi spesifikasi formal. Spesifikasi formal mampu memberikan jawaban yang lebih tepat terhadap pertanyaan tentang perilaku yang diinginkan dalam perangkat lunak, berbeda dengan spesifikasi non-formal yang lebih mengandalkan semantik kata. Akhir kata semoga buku ini berguna bagi para pembaca. Terima Kasih.

Semarang, Juli 2024
Penulis

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

DAFTAR ISI

BAB 1
PENGANTAR PERANGKAT LUNAK

1.1 PENDAHULUAN

Siklus hidup perangkat lunak adalah urutan aktivitas berbeda yang terjadi selama pengembangan perangkat lunak. Ada juga kiriman berbeda yang dihasilkan. Meskipun kiriman bisa berupa perjanjian atau evaluasi, biasanya kiriman berupa objek, seperti kode sumber atau panduan pengguna. Biasanya, aktivitas dan kiriman berkaitan erat. Milestones adalah peristiwa yang dapat digunakan untuk memberitahukan status proyek. Misalnya, penyelesaian panduan pengguna bisa menjadi sebuah pencapaian. Untuk tujuan manajemen, pencapaian penting karena penyelesaian pencapaian memungkinkan manajer menilai kemajuan pengembangan perangkat lunak.

Jenis Aktivitas Siklus Hidup Perangkat Lunak

- *Kelayakan*—Menentukan apakah pembangunan yang diusulkan bermanfaat.
  Analisis pasar—Menentukan apakah terdapat pasar potensial untuk produk ini.
- *Persyaratan*—Menentukan fungsionalitas apa yang harus dimiliki perangkat lunak.
  Requirement elisitasi—Mendapatkan persyaratan dari pengguna.
  Analisis domain—Menentukan tugas dan struktur apa yang umum terjadi pada masalah ini.
- *Perencanaan proyek*—Menentukan cara mengembangkan perangkat lunak.
  Analisis biaya—Menentukan perkiraan biaya.
  Penjadwalan—Membuat jadwal untuk pengembangan.
  Jaminan kualitas perangkat lunak—Menentukan aktivitas yang akan membantu memastikan kualitas produk.
  Struktur rincian kerja—Menentukan subtugas yang diperlukan untuk mengembangkan produk.
- *Desain*—Menentukan bagaimana perangkat lunak harus menyediakan fungsionalitasnya.
  Desain arsitektur—Merancang struktur sistem. Desain antarmuka—Menentukan antarmuka antar bagian sistem.
  Desain terperinci—Merancang algoritme untuk masing-masing bagian.
- *Implementasi*—Membangun perangkat lunak.
- *Pengujian*—Mengeksekusi perangkat lunak dengan data untuk membantu memastikan bahwa perangkat lunak berfungsi dengan benar.
  Pengujian unit—Pengujian oleh pengembang asli.
  Pengujian integrasi—Pengujian selama integrasi perangkat lunak.
  Pengujian sistem—Menguji perangkat lunak di lingkungan yang sesuai dengan lingkungan operasional.
  Pengujian alfa—Pengujian oleh pelanggan di situs pengembang.

Pengujian beta—Pengujian oleh pelanggan di situs pelanggan.
Pengujian penerimaan—Pengujian untuk memuaskan pembeli.
Pengujian regresi—Menyimpan pengujian dari versi sebelumnya untuk memastikan bahwa versi baru mempertahankan kemampuan sebelumnya.

- *Pengiriman*—Memberikan solusi perangkat lunak yang efektif kepada pelanggan.
  Instalasi—Membuat perangkat lunak tersedia di lokasi operasional pelanggan.
  Pelatihan—Mengajarkan pengguna untuk menggunakan perangkat lunak.
  Meja bantuan—Menjawab pertanyaan pengguna.
- *Pemeliharaan*—Memperbarui dan meningkatkan perangkat lunak untuk memastikan kegunaannya secara berkelanjutan.

Dokumen Khusus

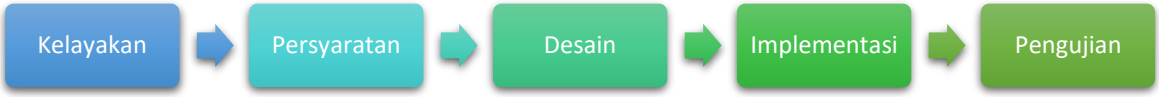
- *Pernyataan kerja*—Deskripsi awal tentang kemampuan yang diinginkan, sering kali dihasilkan oleh pengguna.
- *Spesifikasi kebutuhan perangkat lunak*—Menjelaskan apa yang akan dilakukan oleh perangkat lunak yang telah selesai.
  Model objek—Menampilkan objek/kelas utama.
  Skenario kasus penggunaan—Menampilkan urutan perilaku yang mungkin terjadi dari sudut pandang pengguna.
- *Jadwal proyek*—Menjelaskan urutan tugas dan perkiraan waktu serta upaya yang diperlukan.
- *Rencana pengujian perangkat lunak*—Menjelaskan bagaimana perangkat lunak akan diuji untuk memastikan perilaku yang tepat.
  Tes penerimaan—Tes yang ditunjuk oleh pelanggan untuk menentukan penerimaan sistem.
- *Desain perangkat lunak*—Menjelaskan struktur perangkat lunak. Desain arsitektur—Struktur tingkat tinggi dengan interkoneksi.
  Desain detail—Desain modul atau objek tingkat rendah.
- *Rencana jaminan kualitas perangkat lunak (rencana SQA)*—Menjelaskan aktivitas yang akan dilakukan untuk memastikan kualitas.
- *Panduan pengguna*—Menjelaskan cara menggunakan perangkat lunak yang sudah jadi.
- *Kode sumber*—Kode produk sebenarnya.
- *Laporan pengujian*—Menjelaskan pengujian apa yang dilakukan dan bagaimana sistem berperilaku.
- *Laporan kerusakan*—Menjelaskan ketidakpuasan pelanggan terhadap perilaku spesifik sistem; biasanya, ini adalah kegagalan atau kesalahan perangkat lunak.

1.2 MODEL SIKLUS HIDUP PERANGKAT LUNAK

Empat model siklus hidup perangkat lunak berbeda yang disajikan pada bagian berikut adalah model siklus hidup perangkat lunak yang paling umum.

Model Urutan Linear

Model ini, ditunjukkan pada Gambar 1-1, juga disebut model air terjun, karena diagram tipikal terlihat seperti rangkaian kaskade. Pertama kali dijelaskan oleh Royce pada tahun 1970, ini merupakan realisasi pertama dari rangkaian tugas standar.



Gambar 1.1 Model Waterfall

Ada banyak versi model air terjun. Meskipun tugas-tugas pembangunan tertentu akan terjadi di hampir setiap pembangunan, ada banyak cara untuk membaginya ke dalam beberapa fase. Perhatikan bahwa dalam versi air terjun ini, aktivitas perencanaan proyek dimasukkan dalam fase persyaratan. Demikian pula, tahap pengiriman dan pemeliharaan telah ditinggalkan.

Model Prototyping

Model siklus hidup perangkat lunak ini membangun versi sekali pakai (atau prototipe). Prototipe ini dimaksudkan untuk menguji konsep dan persyaratannya. Prototipe akan digunakan untuk mendemonstrasikan perilaku yang diusulkan kepada pelanggan. Setelah mendapat persetujuan dari pelanggan, maka pengembangan perangkat lunak biasanya mengikuti tahapan yang sama dengan model sekuensial linier. Upaya yang dilakukan untuk membuat prototipe biasanya membuahkan hasil dengan tidak mengembangkan fitur-fitur yang tidak perlu.

Model Peningkatan

D. L. Parnas mengusulkan model inkremental.1 Tujuannya adalah untuk merancang dan memberikan kepada pelanggan subset minimal dari keseluruhan sistem yang masih merupakan sistem yang berguna. Proses ini akan terus berulang sepanjang siklus hidup dengan tambahan sedikit peningkatan. Keuntungannya antara lain memberikan pelanggan sistem kerja lebih awal dan peningkatan kerja.

Model Spiral Boehm

B. Boehm memperkenalkan model spiral.2 Gambaran model ini adalah spiral yang dimulai dari tengah dan terus meninjau kembali tugas-tugas dasar komunikasi pelanggan, perencanaan, analisis risiko, rekayasa, konstruksi dan pelepasan, serta evaluasi pelanggan.

LATIHAN SOAL

1. Bagaimana model siklus hidup bertahap membantu manajemen perangkat lunak?
2. Apa dua karakteristik yang diperlukan dari suatu pencapaian?
3. Untuk masing-masing dokumen berikut, tunjukkan fase mana dari siklus hidup perangkat lunak yang dihasilkan: panduan pengguna akhir, desain arsitektur, rencana SQA, spesifikasi modul, kode sumber, pernyataan kerja, rencana pengujian, panduan pengguna awal, desain rinci, perkiraan biaya, rencana proyek, laporan pengujian, dokumentasi.

4. Urutkan tugas-tugas berikut dalam model air terjun: pengujian penerimaan, perencanaan proyek, pengujian unit, tinjauan persyaratan, estimasi biaya, desain tingkat tinggi, analisis pasar, desain tingkat rendah, pengujian sistem, tinjauan desain, implementasi, persyaratan spesifikasi.
5. Gambarlah diagram yang mewakili model siklus hidup berulang.

BAB 2
PROSES PERANGKAT LUNAK DAN MODEL LAINNYA

2.1 MODEL PROSES PERANGKAT LUNAK

Model proses perangkat lunak (SPM) menggambarkan proses yang dilakukan untuk mencapai pengembangan perangkat lunak. Model proses perangkat lunak biasanya mencakup hal-hal berikut:

- Tugas
- Artefak (file, data, dll.)
- Aktor
- Keputusan (opsional)

Notasi yang digunakan bisa bermacam-macam. Model proses perangkat lunak standar menggunakan oval untuk tugas dan proses. Artefak direpresentasikan dalam bentuk persegi panjang dan aktornya digambarkan dalam gambar tongkat. Banyak model proses perangkat lunak tidak menyertakan keputusan. Kami akan menggunakan berlian setiap kali kami menunjukkan keputusan. Alirannya ditunjukkan oleh busur dan biasanya dari kiri ke kanan dan dari atas ke bawah. Busur biasanya tidak diberi label.

Berikut ini adalah aturan dan interpretasi untuk model proses yang benar:

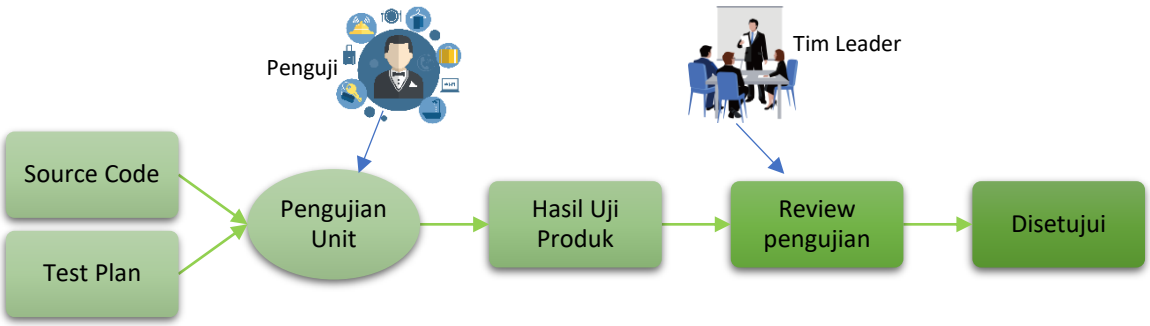
- Dua tugas tidak dapat dihubungkan dengan busur. Tugas harus dipisahkan berdasarkan artefak.
- Tugas tidak dapat dijalankan sampai artefak masukannya ada.
- Terdapat satu atau lebih tugas awal dan satu atau lebih tugas terminal.
- Semua tugas harus dapat dijangkau dari tugas awal.
- Terdapat jalur dari setiap tugas ke tugas terminal.

Model proses perangkat lunak dapat bersifat deskriptif; artinya, mereka dapat menggambarkan apa yang terjadi dalam suatu proyek pembangunan. Model deskriptif sering kali dibuat sebagai bagian dari analisis postmortem suatu proyek. Hal ini dapat berguna dalam hal mengidentifikasi masalah dalam proses pengembangan perangkat lunak. Atau, model proses perangkat lunak dapat bersifat preskriptif; artinya, model proses perangkat lunak dapat menggambarkan apa yang seharusnya terjadi. Model proses perangkat lunak preskriptif dapat digunakan untuk menggambarkan proses pengembangan perangkat lunak standar. Ini dapat digunakan sebagai alat pelatihan bagi karyawan baru, sebagai referensi untuk kejadian yang tidak biasa, dan untuk mendokumentasikan apa yang seharusnya terjadi.

Contoh 2.1

Gambar 2.1 adalah model proses untuk perangkat lunak pengujian unit. Ada dua aktor: penguji dan pemimpin tim. Penguji unit, tentu saja, bertanggung jawab atas pengujian unit. Penguji unit menggunakan kode sumber dan rencana pengujian untuk menyelesaikan pengujian unit. Hasil dari kegiatan ini berupa artefak, hasil pengujian. Ketua tim meninjau hasil pengujian, dan hasil kegiatan ini harus menjadi persetujuan unit pengujian. Model ini tidak

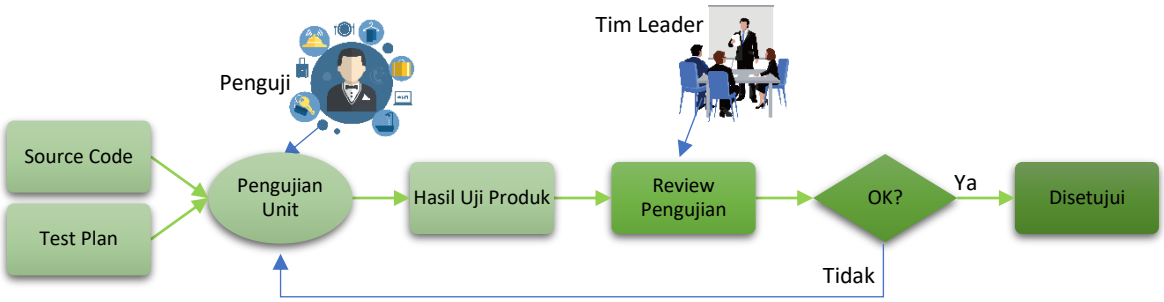
secara eksplisit menunjukkan apa yang terjadi bila proses tidak berhasil. Dapat disimpulkan bahwa penguji unit terus menguji sampai dia puas. Demikian pula jika ketua tim belum siap memberikan persetujuan, maka proses dapat dicadangkan untuk mengulangi pengujian unit.



Gambar 2.1. Diagram proses untuk pengujian unit.

Contoh 2.2

Gambarkan model proses yang menunjukkan keputusan. Menambahkan keputusan memungkinkan model proses menjadi lebih eksplisit tentang apa yang terjadi dalam semua keadaan, seperti yang ditunjukkan pada Gambar 2-2.



Gambar 2-2. Model proses dengan keputusan.

2.2 DIAGRAM ALIRAN DATA

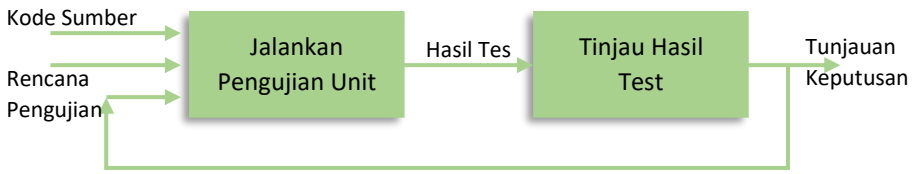
Salah satu diagram paling dasar dalam pengembangan perangkat lunak adalah diagram aliran data. Diagram aliran data menunjukkan aliran data di antara sekumpulan komponen. Komponen tersebut dapat berupa tugas, komponen perangkat lunak, atau bahkan abstraksi dari fungsionalitas yang akan disertakan dalam sistem perangkat lunak. Para aktor tidak disertakan dalam diagram aliran data. Urutan tindakan seringkali dapat disimpulkan dari urutan kotak aktivitas. Berikut ini adalah aturan dan interpretasi diagram aliran data yang benar:

1. Kotak adalah proses dan harus berupa frase kata kerja.
2. Busur mewakili data dan harus diberi label dengan frase kata benda.
3. Kontrol tidak ditampilkan. Beberapa urutan dapat disimpulkan dari pengurutan.
4. Suatu proses mungkin merupakan aktivitas yang dilakukan satu kali saja, atau dapat juga berarti suatu proses yang berkelanjutan.

5. Dua busur yang keluar dari sebuah kotak dapat menunjukkan bahwa kedua keluaran diproduksi atau salah satu keluaran diproduksi.

Contoh 2.3

Contoh pengujian unit dari bagian sebelumnya dapat digambarkan sebagai diagram aliran data, seperti yang ditunjukkan pada Gambar 2-3.



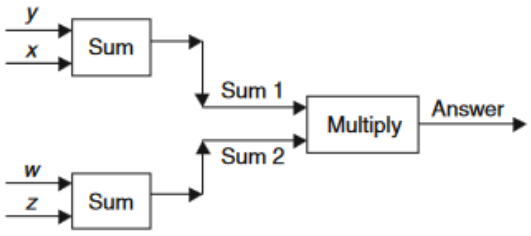
Gambar 2-3. Aliran data untuk pengujian unit.

Gambar 2-3 mengilustrasikan beberapa aturan. Frasa yang ada di dalam kotak adalah frasa kata kerja. Mereka mewakili tindakan. Setiap panah/garis diberi label dengan frase kata benda yang mewakili beberapa artefak.

Diagram aliran data tidak menunjukkan keputusan secara eksplisit. Contoh tersebut menunjukkan bahwa hasil pengujian dapat mempengaruhi pengujian selanjutnya dan bahwa hasil tindakan peninjauan pengujian juga dapat mempengaruhi pengujian (atau pengujian ulang).

Contoh 2.4

Perhitungan rumus matematika $(x + y) * (w + z)$ dapat ditampilkan sebagai rangkaian operasi, seperti ditunjukkan pada Gambar 2-4:



Gambar 2-4 Rangkaian Perhitungan Rumus

2.3 MODEL PETRI NET

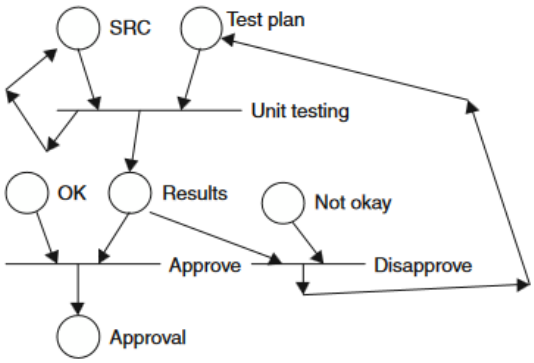
Model dasar petri net terdiri dari node kondisi, busur, node peristiwa, dan token. Jika semua node kondisi input untuk sebuah node peristiwa memiliki token, maka peristiwa tersebut dapat diaktifkan, token tersebut akan dihapus dari node input, dan token ditempatkan pada semua node output dari posisi penembakan. Node kondisi biasanya direpresentasikan dengan lingkaran dan node kejadian direpresentasikan dengan garis horizontal atau persegi panjang.

Dalam model petri net, node kondisi biasanya mewakili beberapa kondisi yang diperlukan—misalnya, keberadaan rencana pengujian. Token pada kondisi berarti kondisi tersebut terpenuhi. Suatu simpul peristiwa (garis horizontal) mewakili suatu peristiwa yang dapat terjadi (kebakaran) ketika semua persyaratan terpenuhi (token di semua simpul

kondisi). Token kemudian ditempatkan di semua node kondisi yang mengikuti peristiwa tersebut.

Contoh 2.5

Model pengujian petri net ditunjukkan pada Gambar 2-5.



Gambar 2-5. Model jaring petri.

Ada sejumlah variasi berbeda pada model dasar jaring petri.

2.4 MODEL OBJEK

Dalam pengembangan berorientasi objek (Bab 11), baik masalah dalam domain masalah maupun solusi dalam ruang mesin dijelaskan dalam bentuk objek. Dalam solusinya, objek-objek ini biasanya menjadi kelas. Seiring dengan kemajuan persyaratan dan fase desain pengembangan perangkat lunak, objek beralih dari representasi hal-hal dalam domain masalah menjadi struktur pemrograman dalam perangkat lunak.

Model objek mewakili entitas dan hubungan antar entitas. Setiap kotak mewakili jenis objek, dan nama, atribut, dan metode objek tercantum di dalam kotak. Bagian atas kotak untuk nama objek, bagian kedua untuk atribut, dan bagian bawah untuk metode. Busur antara dua objek melambangkan hubungan antar objek. Busur dapat diberi label di tengahnya dengan nama asosiasi. Peran tersebut mungkin diberi label di ujung yang berlawanan. Selain itu, pada setiap ujung dapat diberikan multiplisitas yang menunjukkan berapa banyak asosiasi berbeda dari jenis yang sama yang diperbolehkan.

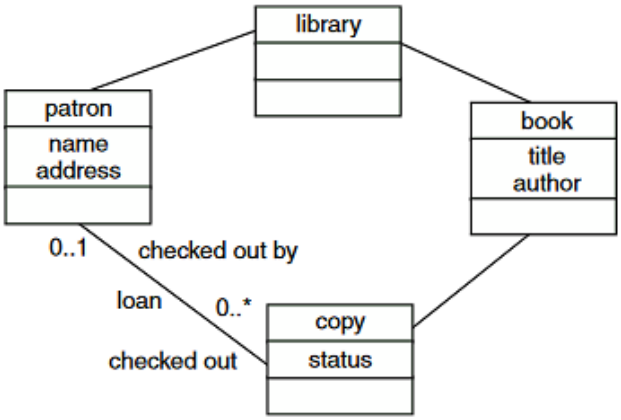
Tiga jenis hubungan utama adalah (1) pewarisan, (2) agregasi, dan (3) asosiasi. Hubungan pewarisan menyiratkan bahwa objek di bagian bawah busur adalah kasus khusus dari objek di bagian atas busur. Misalnya, objek teratas mungkin adalah kendaraan dan objek terbawah adalah mobil, yang merupakan sejenis kendaraan. Hal ini sering disebut dengan hubungan “is-a”. Hubungan agregasi menyiratkan bahwa objek di bagian bawah busur merupakan komponen objek di bagian atas busur. Misalnya, objek teratas mungkin adalah mobil dan objek terbawah mungkin adalah mesin. Hal ini sering disebut hubungan “bagian dari”. Jenis hubungan terakhir adalah asosiasi, dan busur ini menyiratkan bahwa salah satu objek dikaitkan dengan objek lainnya. Misalnya, hubungan “ayah-anak” adalah sebuah asosiasi. Hubungan ini mungkin bersifat dua arah, atau mungkin hanya satu arah.

Meskipun terdapat banyak notasi yang berbeda, kami akan menggunakan notasi yang sesuai dengan standar Unified Modeling Language (UML).

Contoh 2.6

Membangun model objek untuk perpustakaan. Benda-benda pada perpustakaan sederhana yang ditunjukkan pada Gambar 2.6 terdiri dari perpustakaan, buku, salinan buku, dan pengunjung.

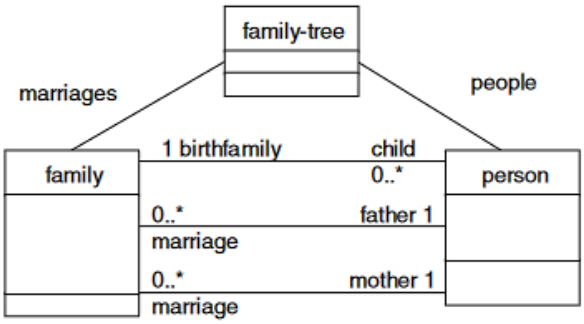
Tak satu pun dari metode objek yang ditampilkan. Perpustakaan mempunyai hubungan agregasi dengan buku dan dengan patron. Artinya, perpustakaan benar-benar terdiri dari buku dan pengunjung. Hubungan antara buku dan salinan bukanlah agregasi atau pewarisan. Buku objek mewakili abstraksi sebuah buku, sedangkan salinannya adalah benda fisik yang dipinjamkan. Hubungan antara patron dan salinan disebut ''pinjaman''. Dari sudut pandang salinan, perannya adalah ''diperiksa oleh'' dan dari patron perannya adalah ''check out.'' Keberagaman tersebut menunjukkan bahwa salinan dapat tidak dapat diperiksa atau dapat memiliki hubungan ini hanya dengan satu patron dalam satu waktu (''0,1''). Multiplisitas lainnya, ''0.*'', menunjukkan bahwa patron dapat memiliki nol atau satu atau banyak hubungan ''check out'' dalam satu waktu.



Gambar 2-6. Model objek perpustakaan sederhana

Contoh 2.7

Buatlah model objek untuk sistem silsilah keluarga yang menyimpan informasi silsilah tentang anggota keluarga. Gambar 2-7 menunjukkan bahwa setiap orang mempunyai keluarga kandung. Setiap pernikahan mempunyai pribadi ayah dan ibu. Banyak atribut yang tidak dimasukkan dalam diagram, dan tidak ada fungsi yang ditampilkan.



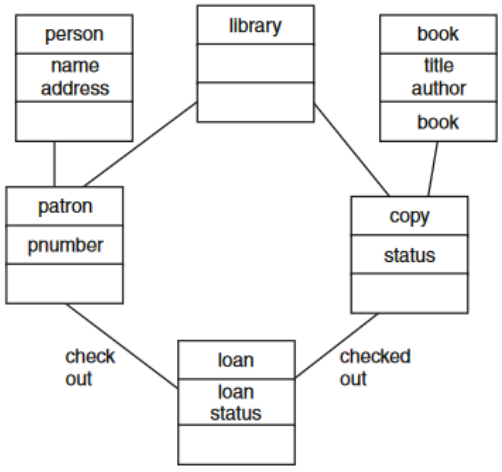
Gambar 2-7. Model objek pohon keluarga

Ketergantungan Keberadaan

Salah satu pendekatan untuk memperjelas hubungan tersebut adalah dengan memperkenalkan hubungan berbeda yang disebut ketergantungan keberadaan (ED). Hubungan ketergantungan keberadaan didefinisikan sebagai berikut: Suatu kelas (induk) dapat dikaitkan dengan kelas bawah (anak) jika kelas bawah (anak) hanya ada ketika kelas atas (induk) ada dan setiap turunan dari kelas bawah (anak) dikaitkan dengan tepat satu instance dari kelas atas (induk). Hubungan dan pewarisan ini dapat digunakan untuk mewakili domain masalah apa pun.

Contoh 2.8

Bangun model objek untuk perpustakaan menggunakan hubungan ketergantungan yang ada. Seperti yang ditunjukkan pada Gambar 2-6 pada contoh 2.6, semua hubungan kecuali ''pinjaman'' dan buku perpustakaan memenuhi persyaratan keberadaan ketergantungan. Hubungan ''pinjaman'' tidak memuaskan, karena objek salinan bisa ada sebelum keberadaan objek pelindung yang memeriksanya. Namun, objek pinjaman dapat dibuat yang memenuhi hubungan ED. Objek ''buku'' tidak bisa menjadi anak perpustakaan, karena buku bisa ada sebelum dan sesudah perpustakaan tertentu. ''Orang'' ditambahkan ke diagram (lihat Gambar 2-8) untuk menunjukkan bagian pelindung yang tidak bergantung pada keberadaan ''perpustakaan''.



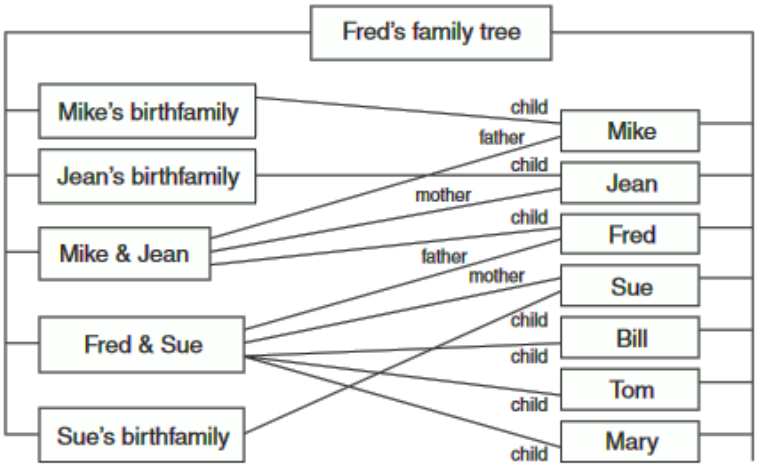
Gambar 2-8. Model objek perpustakaan menggunakan ED.

Diagram Instan

Diagram objek mewakili jenis objek. Jadi, kotak berlabel ''mobil'' mewakili atribut dan fungsi semua mobil. Terkadang hubungan antar instance objek tidak begitu jelas dalam diagram objek. Diagram contoh menunjukkan contoh contoh objek dan dapat memperjelas hubungan.

Contoh 2.9

Gambarlah model contoh yang memperlihatkan Fred, istrinya Sue, anak-anak mereka Bill, Tom, dan Mary, serta orang tuanya Mike dan Jean. (Lihat Gambar 2-9.)



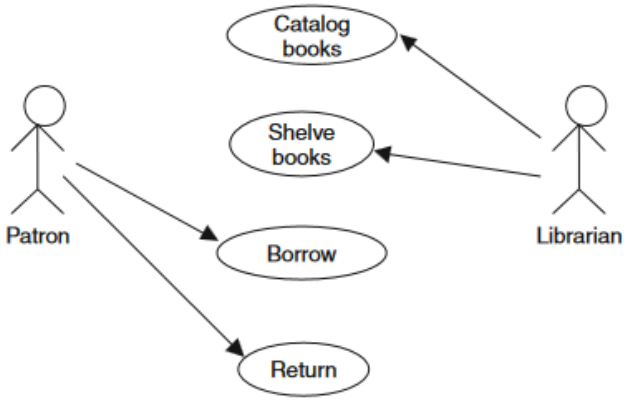
Gambar 2-9. Diagram contoh pohon keluarga Fred.

2.5 DIAGRAM KASUS PENGGUNAAN

Diagram use case adalah bagian dari kumpulan diagram UML. Ini menunjukkan aktor penting dan fungsionalitas suatu sistem. Aktor diwakili oleh figur tongkat dan fungsi diwakili oleh oval. Aktor dikaitkan dengan fungsi yang dapat mereka lakukan.

Contoh 2.10

Gambarlah diagram kasus penggunaan untuk perpustakaan sederhana. (Lihat Gambar 2-10.)



Gambar 2-10. Kasus penggunaan untuk perpustakaan sederhana.

Fungsi dalam oval adalah metode kelas dalam model objek. Objek patron dapat meminjam dan mengembalikan salinannya. Aktor pustakawan bukanlah sebuah objek pada model objek. Pustakawan dalam use case menunjukkan bahwa beberapa fungsi—misalnya, katalog dan rak buku—bukan fungsi yang tersedia bagi patron.

2.6 SKENARIO

Skenario adalah deskripsi dari satu rangkaian tindakan yang dapat terjadi dalam domain masalah ini.

Contoh 2.11

Tulis skenario untuk masalah perpustakaan.

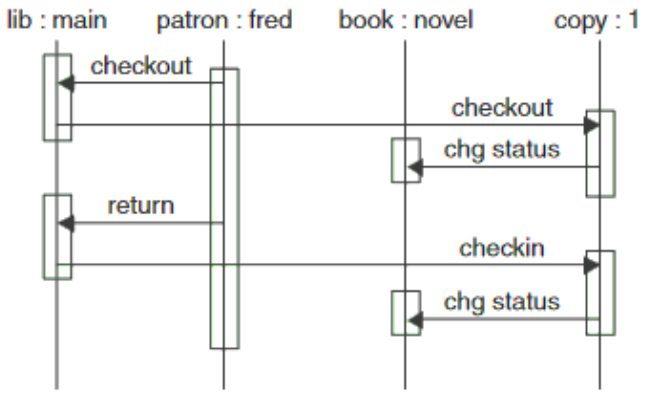
Fred, seorang pelindung, pergi ke perpustakaan dan memeriksa buku. Dua bulan kemudian, dia membawa buku perpustakaan yang sudah lewat waktunya kembali ke perpustakaan.

Diagram Urutan

Diagram urutan adalah bagian dari kumpulan diagram UML. Diagram memiliki garis vertikal, yang mewakili instance kelas. Setiap garis vertikal diberi label di bagian atas dengan nama kelas diikuti dengan titik dua diikuti dengan nama instance. Misalnya, baris pertama diberi label dengan lib:main untuk instance main dari perpustakaan kelas. Panah horizontal menggambarkan pemanggilan fungsi. Ekor anak panah berada pada garis kelas pemanggil, dan kepala anak panah berada pada garis kelas pemanggil. Nama fungsinya ada pada panah. Blok lebar pada garis vertikal menunjukkan waktu eksekusi fungsi yang dipanggil. Pengembalian biasanya tidak ditampilkan. Beberapa panggilan ke fungsi yang sama sering kali ditampilkan hanya sebagai satu panah.

Contoh 2.12

Gambarlah diagram urutan untuk skenario Contoh 2.11. (Lihat Gambar 2-11.)



Gambar 2-11. Diagram urutan untuk skenario checkout.

Diagram ini lebih dekat dengan tahap desain dibandingkan model objek yang disajikan pada Contoh 2.8. Ada fungsi yang digunakan dalam diagram ini yang tidak terwakili dalam model objek sebelumnya. Selain itu, urutan panggilan yang direpresentasikan dalam diagram ini bergantung pada desain sebenarnya.

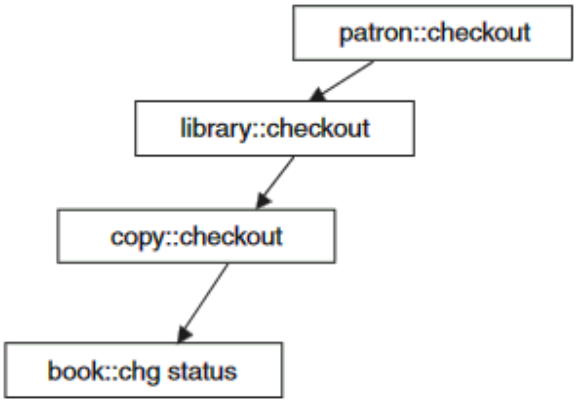
2.7 DIAGRAM HIERARKI

Diagram hierarki menunjukkan struktur pemanggilan suatu sistem. Setiap kotak mewakili suatu fungsi. Suatu garis ditarik dari satu fungsi ke fungsi lainnya jika fungsi pertama dapat memanggil fungsi kedua. Semua kemungkinan panggilan ditampilkan.

Ini bukan salah satu rangkaian diagram UML dan sering kali tidak digunakan dalam pengembangan berorientasi objek. Namun, diagram ini bisa menjadi diagram yang sangat berguna untuk memahami struktur dinamis suatu sistem.

Contoh 2.13

Gambarlah diagram hierarki untuk program perpustakaan yang digunakan pada Contoh 2.12. (Lihat Gambar 2-12.)



Gambar 2-12. Diagram hierarki.

Grafik Aliran Kontrol

Grafik aliran kendali (CFG) menunjukkan struktur kendali kode. Setiap node (lingkaran) mewakili blok kode yang hanya memiliki satu cara untuk melewati kode tersebut. Artinya, terdapat satu pintu masuk di awal blok dan satu pintu keluar di akhir. Jika ada pernyataan dalam blok yang dieksekusi, maka semua pernyataan dalam blok tersebut akan dieksekusi. Busur antar node mewakili kemungkinan aliran kendali. Artinya, jika memungkinkan blok B dieksekusi, tepat setelah blok A, maka harus ada busur dari blok A ke blok B.

Berikut ini adalah aturan diagram alur kendali yang benar:

1. Harus ada satu simpul awal.
2. Dari node awal, harus ada jalur menuju setiap node.
3. Dari setiap node harus ada jalur menuju node halt.

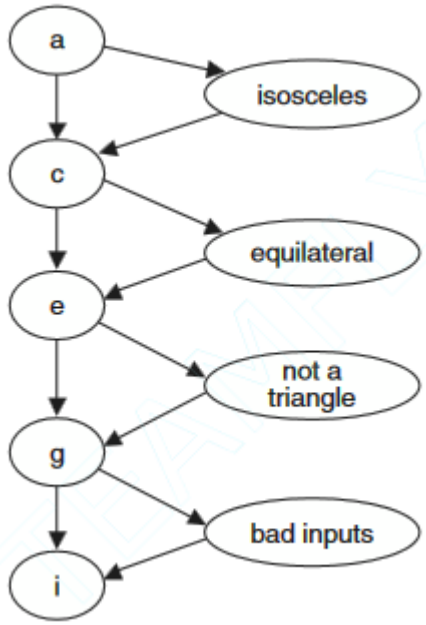
Contoh 2.14

Gambarlah grafik aliran kendali untuk permasalahan segitiga berikut.

```
read x,y, z;
type="'scalene'';
if (x == y or x == z or y == z) type ='''isosceles'';
if (x == y and x == z) type=''equilateral'';
if (x >= y+z or y >= x+z or z >= x+y) type ='''not a
triangle'';
```

```
if (x <= 0 or y <= 0 or| z <= 0) type =''bad inputs'';
print type;
```

Pada Gambar 2-13, node ''a'' mewakili dua pernyataan pertama dan pernyataan if. ''Tipe = sama kaki'' ada di simpul berlabel ''sama kaki''. Demikian pula, node ''c'' mewakili pernyataan if berikutnya, dan node ''equilateral'' mewakili isi pernyataan if.



Gambar 2-13. Grafik aliran kontrol untuk program segitiga.

Diagram Keadaan

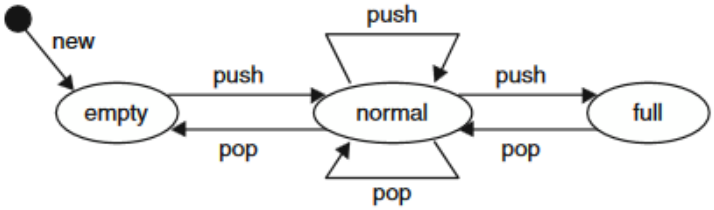
Keadaan suatu mesin atau program adalah kumpulan semua nilai dari semua variabel, register, dan sebagainya. Diagram keadaan menunjukkan keadaan sistem dan kemungkinan transisi antara keadaan tersebut. Suatu program atau mesin akan memiliki jumlah status berbeda yang sangat banyak. Namun, banyak negara bagian yang memiliki perilaku serupa pada input berikutnya, dan seterusnya. Sekelompok negara dengan perilaku serupa dapat dikelompokkan menjadi suatu negara. Negara-negara bagian ini dapat digambarkan untuk menunjukkan transisi antar negara bagian. Banyak program yang paling baik digambarkan dengan diagram keadaan.

Berikut ini adalah aturan untuk diagram keadaan yang benar:

1. Ada satu keadaan awal.
2. Setiap state dapat dicapai dari state awal.
3. Dari setiap state pasti ada jalur menuju stop state.
4. Setiap transisi antar negara bagian harus diberi label dengan peristiwa yang menyebabkan transisi tersebut.

Contoh 2.15

Gambarlah diagram keadaan untuk tumpukan berukuran tetap. (Lihat Gambar 2-14.)

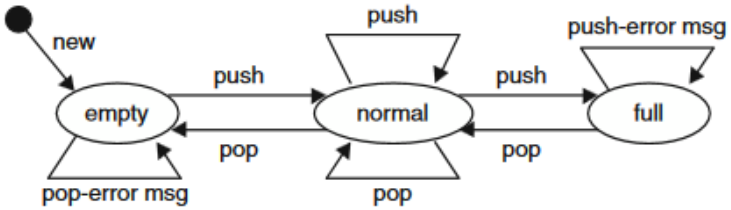


Gambar 2-14. Diagram keadaan untuk tumpukan berukuran tetap.

Ada dua pendekatan untuk menyatakan diagram. Pada Gambar 2-14, hanya transisi legal atau non-kesalahan yang ditentukan. Diasumsikan bahwa transisi apa pun yang tidak ditampilkan adalah ilegal. Misalnya, tidak ada transisi push dari keadaan penuh. Pendekatan lain adalah dengan menunjukkan semua transisi, termasuk transisi yang menyebabkan kesalahan.

Contoh 2.16

Gambarlah diagram keadaan untuk tumpukan dengan semua transisi kesalahan yang ditampilkan. (Lihat Gambar 2-15.)



Gambar 2-15. Diagram keadaan menunjukkan transisi kesalahan.

Diagram keadaan dapat diambil langsung dari kode sumber. Setiap fungsi harus diperiksa untuk pengambilan keputusan yang didasarkan pada nilai variabel (keadaan sistem).

Contoh 2.17

Kode berikut untuk metode push pada tumpukan terbatas:

```
int push (int item) {
     if (stackindex == MAX) {return ERROR; }
     stack[stackindex++] = item;
     return 0;
     }
```

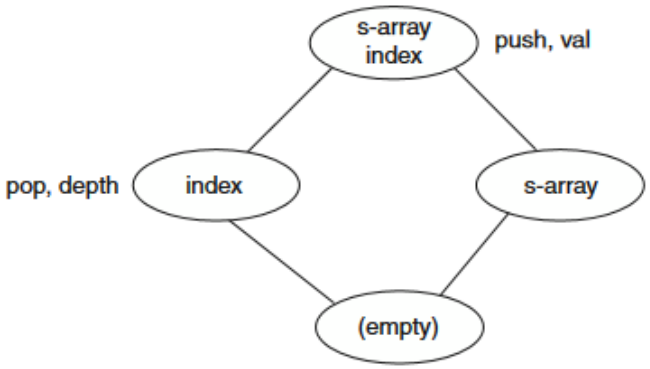
Dari kode ini, dua keadaan dengan perilaku fungsi yang berbeda dapat diidentifikasi. Perhatikan bahwa indeks tumpukan dimulai dari nol. Satu status terkait dengan kondisi stackindex == MAX, dan status lainnya terkait dengan kondisi stackindex != MAX. Analisis kenaikan akan menunjukkan bahwa status kedua adalah stackindex <MAX (setidaknya dari metode ini). Menganalisis metode pop akan mengungkapkan keadaan tumpukan yang kosong.

Model Kisi

Kisi adalah struktur matematika yang menunjukkan hubungan himpunan. Meskipun tidak terlalu sering digunakan dalam pengembangan perangkat lunak, ini lebih banyak digunakan untuk menunjukkan hubungan antara kumpulan fungsi dan atribut.

Contoh 2.18

Gambarlah model kisi untuk implementasi tumpukan yang memiliki atribut untuk array (s-array) dan indeks elemen teratas tumpukan (index) dan metode untuk push, pop, val (menampilkan nilai teratas), dan kedalaman. (Lihat Gambar 2-16.)



Gambar 2-16. Model kisi untuk contoh tumpukan.

Node atas mewakili himpunan semua atribut dan node bawah mewakili himpunan kosong. Node diberi anotasi dengan nama fungsi yang menggunakan subset atribut tersebut.

LATIHAN SOAL

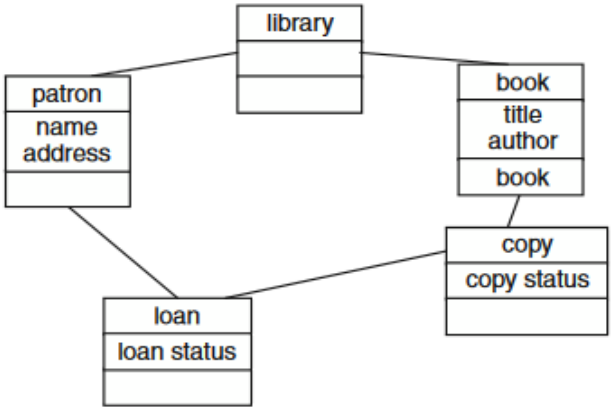
1. Apa perbedaan antara model siklus hidup perangkat lunak dan model proses?
2. Apa perbedaan antara model proses deskriptif dan model proses preskriptif?
3. Mengapa pengambilan keputusan lebih umum dilakukan pada model proses preskriptif dibandingkan model proses deskriptif?
4. Mengapa tugas-tugas dalam model proses harus dipisahkan berdasarkan artefak?
5. Mengapa tugas dalam model proses tidak dapat dimulai sampai artefak masukannya ada?
6. Mengapa setiap node dalam model proses harus mempunyai jalur dari node awal ke node itu sendiri dan jalur dari node itu sendiri ke node terminal?
7. Dalam Contoh 2.3, bagaimana seseorang dapat membedakan antara tinjauan pengujian yang hanya terjadi setelah semua pengujian unit selesai dan tinjauan pengujian yang terjadi setelah setiap modul diuji unit?
8. Apa yang ditentukan oleh diagram aliran data tentang aliran kontrol?
9. Kapan posisi penembakan jaring petri menyala?
10. Apa yang terjadi bila jaring petri menyala?
11. Apa perbedaan antara domain masalah dan ruang solusi?
12. Perubahan apa saja yang terjadi pada model objek mulai dari persyaratan hingga desain?

13. Klasifikasikan masing-masing hubungan berikut sebagai hubungan warisan, hubungan agregasi, atau asosiasi umum:
    Mobil — Mobil Kota Lincoln Orang — Perpustakaan Siswa — Buku Pelindung Perpustakaan — Mobil Fotokopi — Pelindung Pengemudi — Kelas Peminjaman Buku — Siswa
14. Klasifikasikan masing-masing berikut ini sebagai suatu kelas atau turunan dari suatu kelas:
    Mobil saya
    Orang
    Fred
    Kendaraan
    Profesor
    Departemen CIS
15. Apa hubungan antara skenario dan diagram keadaan yang menunjukkan semua kemungkinan rangkaian tindakan?
16. Dalam diagram interaksi, apakah kelas pemanggil atau kelas yang dipanggil berada di ujung panah?
17. Jelaskan mengapa relasi agregasi merupakan relasi pada domain masalah dan bukan pada domain implementasi.
18. Mengapa diagram aliran data tidak memiliki aturan tentang jangkauan antar node?

Latihan Tambahan (Diskusi)

1. Menggambar model proses tugas pengecatan dinding suatu ruangan. Meliputi tugas-tugas berikut: memilih warna, membeli cat, membersihkan dinding, mengaduk cat, mengecat dinding.
2. Penulis menggunakan sesi interaktif ketika dia mengajar mata kuliah yang mencakup siswa pembelajaran jarak jauh. Penulis membagi siswa menjadi beberapa tim dan memposting masalahnya di halaman Web. Tim mengerjakan masalah menggunakan ruang obrolan, mengajukan pertanyaan kepada instruktur menggunakan papan pesan, dan menyampaikan solusi melalui email. Instruktur kemudian menilai solusi menggunakan lembar penilaian. Gambarkan model proses untuk sesi interaktif.
3. Gambarlah diagram aliran data untuk masalah perpustakaan sederhana.
4. Gambarlah diagram aliran data untuk masalah pabrik.
5. Gambarlah diagram aliran data untuk masalah toko kelontong.
6. Gambarlah model objek untuk pohon biner.
7. Gambarlah diagram contoh model objek pohon biner.
8. Gambarlah model objek untuk soal toko kelontong.
9. Gambarlah model objek untuk soal pabrik.
10. Tuliskan skenario tambahan untuk patron yang memeriksa buku dari Contoh 2.11.

11. Gambarlah diagram keadaan untuk antarmuka pengguna grafis yang memiliki menu utama, menu file dengan perintah buka file, dan perintah keluar pada setiap menu. Asumsikan hanya satu file yang dapat dibuka dalam satu waktu.
12. Perluas model objek yang ditunjukkan pada Gambar 2-17 untuk masalah perpustakaan dengan menyertakan objek reservasi sehingga pengunjung dapat memesan buku yang semua salinannya sudah diperiksa.



Gambar 2-17. Model objek masalah perpustakaan.

13. Membangun mesin negara untuk masalah perpustakaan dengan kemampuan memesan buku

BAB 3
MANAJEMEN PROYEK PERANGKAT LUNAK

3.1 PENDAHULUAN

Meskipun kata ''manajer'' mungkin mengingatkan kita pada manajer dalam komik ''Dilbert'', manajemen itu penting. Manajemen proyek perangkat lunak adalah tugas penting dalam merencanakan, mengarahkan, memotivasi, dan mengoordinasikan sekelompok profesional untuk mencapai pengembangan perangkat lunak. Manajemen proyek perangkat lunak menggunakan banyak konsep manajemen secara umum, namun juga mempunyai beberapa permasalahan unik dalam pengembangan perangkat lunak. Salah satu kekhawatiran tersebut adalah visibilitas proyek.

urangnya visibilitas produk perangkat lunak selama pengembangan perangkat lunak membuatnya sulit untuk dikelola. Di banyak bidang lain, mudah untuk melihat kemajuan atau kekurangan kemajuan. Banyak proyek perangkat lunak terhenti saat penyelesaiannya mencapai 90 persen. Tanyakan kepada programmer mana pun apakah bug yang dia temukan adalah bug terakhir dalam perangkat lunaknya, dan jawabannya hampir selalu berupa ya. Banyak teknik dalam manajemen perangkat lunak ditujukan untuk mengatasi kurangnya visibilitas ini.

3.2 PENDEKATAN MANAJEMEN

Masalah mendasar dalam manajemen proyek perangkat lunak adalah apakah proses atau proyek merupakan fitur penting yang dikelola. Dalam manajemen berorientasi proses, pengelolaan tugas-tugas kecil dalam siklus hidup perangkat lunak ditekankan. Dalam manajemen proyek, tim yang mencapai proyek ditekankan. Hal ini menghasilkan perbedaan sudut pandang yang penting. Dalam pendekatan manajemen proses, jika tim tidak mengikuti siklus hidup perangkat lunak yang ditentukan, hal ini akan menjadi kesulitan besar. Dalam pendekatan manajemen proyek, keberhasilan atau kegagalan secara langsung dikaitkan dengan tim.

Pendekatan Tim

Mengorganisir sekelompok orang menjadi tim yang efisien dan efektif dapat menjadi tugas yang sulit. Membiarkan tim mengembangkan paradigmanya sendiri bisa berisiko. Memilih organisasi tim berdasarkan proyek dan anggota tim dapat membantu menghindari bencana.

Salah satu aspek dari sebuah tim adalah jumlah struktur dalam tim. Meskipun beberapa kelompok pemrogram dapat bekerja secara mandiri, kelompok lain memerlukan struktur yang kuat untuk mencapai kemajuan. Tim kepala pemrogram yang disebutkan di bagian selanjutnya adalah contoh tim yang sangat terstruktur. Dalam tim yang sangat terstruktur, tugas-tugas kecil dibuat untuk setiap anggota. Ini sering disebut “kerikil inci” karena tugasnya merupakan tonggak sejarah kecil. Dalam tim yang berstruktur lemah, tugas biasanya berdurasi lebih lama dan lebih terbuka.

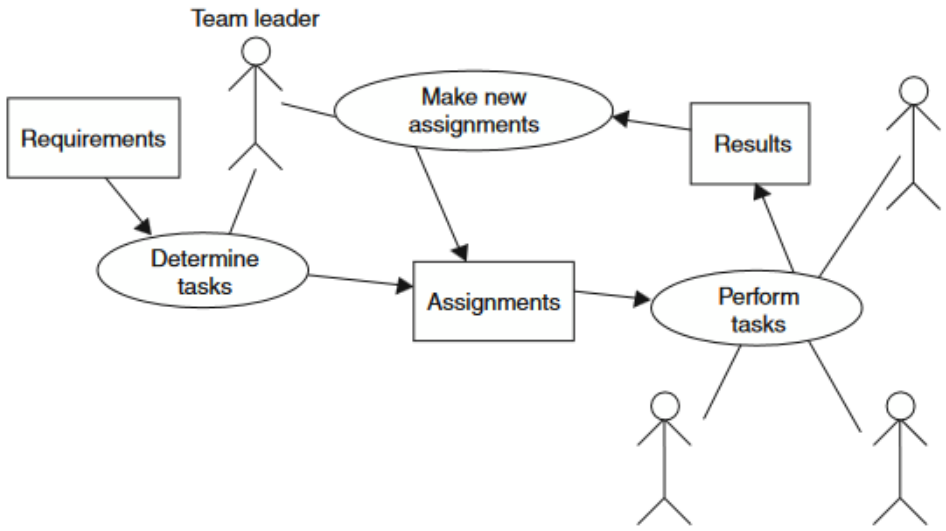
Beberapa tim terdiri dari orang-orang dengan keterampilan serupa. Tim-tim ini sering kali tetap bersama melalui banyak proyek. Tim lain terdiri dari orang-orang dengan keahlian berbeda yang dikelompokkan ke dalam tim berdasarkan kebutuhan akan keterampilan khusus untuk suatu proyek. Hal ini sering disebut organisasi matriks.

Kepala Tim Programmer

IBM mengembangkan konsep tim kepala pemrogram. Ini memberikan peran khusus kepada anggota tim. Kepala programmer adalah programmer terbaik dan memimpin tim. Nonprogrammer digunakan dalam tim untuk dokumentasi dan tugas administrasi. Pemrogram junior dimasukkan untuk dibimbing oleh kepala pemrogram.

Contoh 3.1

Gambarkan model proses tingkat tinggi untuk organisasi tim hierarkis. (Lihat Gambar 3-1.)



Gambar 3-1. Model proses tingkat tinggi untuk struktur tim hierarkis.

Contoh 3.2

Perusahaan WRT memiliki departemen TI dengan beberapa pengembang perangkat lunak berpengalaman dan banyak pemrogram baru. Manajer TI telah memutuskan untuk menggunakan tim yang sangat terstruktur dengan menggunakan pendekatan proses dalam pengelolaan. Setiap tim akan dipimpin oleh pengembang perangkat lunak yang berpengalaman. Setiap anggota tim akan diberikan serangkaian tugas setiap minggunya. Pemimpin tim akan terus meninjau kemajuan dan membuat tugas baru.

Praktek Kritis

Sebagian besar studi pengembangan perangkat lunak telah mengidentifikasi serangkaian praktik yang tampaknya penting untuk keberhasilan. 16 praktik keberhasilan penting berikut ini berasal dari Software Project Managers Network (www.spmn.com):

- Menerapkan manajemen risiko berkelanjutan.
- Perkirakan biaya dan jadwal secara empiris.
- Gunakan metrik untuk mengelola.
- Lacak nilai yang diperoleh.

- Melacak cacat terhadap target kualitas.
- Perlakukan manusia sebagai sumber daya yang paling penting.
- Mengadopsi manajemen konfigurasi siklus hidup.
- Kelola dan telusuri persyaratan.
- Gunakan desain perangkat lunak berbasis sistem.
- Memastikan interoperabilitas data dan database.
- Mendefinisikan dan mengontrol antarmuka.
- Desain dua kali, kode satu kali.
- Menilai risiko dan biaya penggunaan kembali.
- Periksa persyaratan dan desain.
- Kelola pengujian sebagai proses berkelanjutan.
- Kompilasi dan uji asap sesering mungkin.

Contoh 3.3

Manajer TI perusahaan WRT memerlukan proses perangkat lunak yang akan membantu pengembang perangkat lunaknya yang tidak berpengalaman dalam pengembangan perangkat lunak yang sukses. Manajer menggunakan daftar praktik terbaik untuk memastikan bahwa proses perangkat lunaknya akan mencakup aktivitas penting.

Latihan 1: Menerapkan manajemen risiko yang berkelanjutan. Manajer memasukkan langkah-langkah proses sepanjang siklus hidup di mana risiko yang mungkin terjadi diidentifikasi dan dievaluasi, dan tugas-tugas dimasukkan untuk memperbaiki risiko.

Latihan 2: Perkirakan biaya dan jadwal secara empiris. Manajer memasukkan langkah proses untuk memperkirakan biaya pada awal siklus hidup dan langkah-langkah untuk memperkirakan ulang biaya sepanjang siklus hidup. Langkah-langkahnya termasuk mengarsipkan data yang akan digunakan untuk estimasi masa depan.

Latihan 3: Gunakan metrik untuk mengelola. Manajer memilih metrik dan menyertakan langkah-langkah untuk pencatatan metrik dan langkah-langkah untuk mengevaluasi kemajuan berdasarkan metrik.

Latihan 4: Lacak nilai yang diperoleh. Manajer menyertakan langkah-langkah untuk menghitung nilai yang diperoleh (lihat di bawah) dan memposting perhitungannya.

Latihan 5: Melacak cacat terhadap target kualitas. Manajer menetapkan sasaran untuk jumlah laporan cacat yang diterima. Langkah-langkah proses untuk memposting jumlah laporan kerusakan disertakan.

Latihan 6: Perlakukan manusia sebagai sumber daya yang paling penting. Manajer meninjau seluruh proses perangkat lunak untuk mempertimbangkan dampaknya terhadap pemrogram.

Latihan 7: Gunakan manajemen konfigurasi siklus hidup. Manajer menyertakan dalam proses penggunaan alat manajemen konfigurasi untuk semua dokumen dan mencakup langkah-langkah proses untuk memasukkan semua dokumen dan perubahan ke dalam alat manajemen konfigurasi.

Latihan 8: Kelola dan lacak persyaratan. Manajer mencakup langkah-langkah proses untuk memperoleh persyaratan dari pengguna dan langkah-langkah untuk menelusuri setiap persyaratan ke fase pengembangan saat ini.

Latihan 9: Gunakan desain perangkat lunak berbasis sistem. Manajer mencakup langkah-langkah untuk memastikan desain berbasis sistem.
Latihan 10: Memastikan interoperabilitas data dan database. Manajer menyertakan langkah-langkah untuk memeriksa interoperabilitas antara data dan database.
Latihan 11: Mendefinisikan dan mengontrol antarmuka. Manajer mencakup langkah-langkah untuk mendefinisikan dan membuat dasar antarmuka.
Latihan 12: Desain dua kali, kode satu kali. Manajer menyertakan langkah-langkah peninjauan desain.
Latihan 13: Menilai risiko dan biaya penggunaan kembali. Manajer mencakup langkah-langkah untuk mengidentifikasi area yang berpotensi digunakan kembali dan langkah-langkah untuk menilai biaya dan risiko.
Latihan 14: Periksa persyaratan dan desain. Manajer mencakup langkah-langkah inspeksi dalam fase persyaratan dan desain.
Latihan 15: Kelola pengujian sebagai proses yang berkesinambungan. Manajer mencakup langkah-langkah pengujian di semua fase.
Latihan 16: Kompilasi dan lakukan tes asap sesering mungkin. Manajer sering menyertakan langkah-langkah pengujian dalam fase implementasi.

3.3 MODEL KEMATANGAN KEMAMPUAN

Institut Rekayasa Perangkat Lunak (www.sei.cmu.edu) telah mengembangkan Model Kematangan Kemampuan. Model Kematangan Kemampuan Rekayasa Perangkat Lunak (SE-CMM) digunakan untuk menilai proses pengembangan perangkat lunak suatu organisasi. Penilaian terhadap praktik, proses, dan organisasi organisasi digunakan untuk mengklasifikasikan organisasi pada salah satu tingkat berikut:

- ***Level 1: Awal—***Ini adalah level terendah dan biasanya ditandai dengan kekacauan.
- ***Level 2: Dapat Diulang—***Tingkat kemampuan pengembangan ini mencakup pelacakan biaya, jadwal, dan fungsionalitas proyek. Ada kemampuan untuk mengulangi kesuksesan sebelumnya.
- ***Level 3: Ditetapkan—***Tingkat ini memiliki proses perangkat lunak yang ditentukan dan didokumentasikan dan distandarisasi. Semua pengembangan dicapai dengan menggunakan proses standar.
- ***Level 4: Terkelola—***Tingkat ini mengelola proses dan produk secara kuantitatif.
- ***Level 5: Optimalisasi—***Tingkat ini menggunakan informasi kuantitatif untuk terus meningkatkan dan mengelola proses perangkat lunak.

Sebagian besar organisasi akan dinilai pada level 1 pada awalnya. Peningkatan ke tingkat yang lebih tinggi melibatkan upaya besar dalam organisasi dan manajemen proses. Level 5 hanya dicapai oleh beberapa organisasi.

Proses Perangkat Lunak Pribadi

Watts Humphrey telah mengembangkan Proses Perangkat Lunak Pribadi untuk meningkatkan keterampilan insinyur perangkat lunak individu. Pendekatannya membuat individu menyimpan catatan waktu pribadi untuk memantau dan mengukur keterampilan

individu. Salah satu hasilnya adalah mengukur produktivitas individu. Ukuran produktivitas yang umum adalah baris kode yang diproduksi per hari (LOC/hari). Selain itu, kesalahan dihitung waktunya dan dicatat. Hal ini memungkinkan seseorang untuk mempelajari di mana kesalahan terjadi dan menilai berbagai teknik untuk mengetahui pengaruhnya terhadap produktivitas dan tingkat kesalahan. Selain itu, produktivitas dapat digunakan untuk mengevaluasi kewajaran jadwal yang diusulkan.

Contoh 3.4

Programmer X mencatat catatan waktu ini.

Tanggal	Mulai	Berhenti	Istirahat	Delta	Tugas
1 Januari	09:00	15:30	30 Menit Makan Siang	360	Kode 50 LOC
1 Maret	09:00	14:00	30 Menit Makan Siang	270	Kode 60 LOC
1 April	09:00	11:30		150	Kode 50 LOC
	12:00	14:00		120	Pengujian

Pemrogram menghabiskan 360 + 270 + 150 + 120 = 900 menit untuk menulis dan menguji program 160 LOC. Dengan asumsi 5 jam per hari (300 menit/hari), X menghabiskan waktu efektif 3 hari untuk memprogram 160 LOC. Ini memberikan produktivitas 53 LOC/hari. Ketika manajer X menjadwalkan satu minggu untuk membuat kode proyek 1000 = LOC, X dapat memperkirakan bahwa proyek tersebut akan memakan waktu sekitar 4 minggu.

3.4 ANALISIS NILAI PEROLEHAN

Salah satu pendekatan untuk mengukur kemajuan dalam proyek perangkat lunak adalah dengan menghitung berapa banyak yang telah dicapai. Ini disebut analisis nilai yang diperoleh. Ini pada dasarnya adalah persentase perkiraan waktu yang telah diselesaikan. Tindakan tambahan dapat dihitung.

Meskipun hal ini didasarkan pada perkiraan upaya, hal ini dapat didasarkan pada kuantitas apa pun yang dapat diperkirakan dan berkaitan dengan kemajuan.

Tindakan Dasar

- *Biaya Pekerjaan yang Dianggarkan (BCW):* Perkiraan upaya untuk setiap tugas pekerjaan.
- *Anggaran Biaya Pekerjaan yang Dijadwalkan (BCWS):* Jumlah perkiraan upaya untuk setiap tugas pekerjaan yang dijadwalkan untuk diselesaikan pada waktu yang ditentukan.
- *Anggaran saat Penyelesaian (BAC):* Total BCWS dan perkiraan total upaya proyek.
- *Nilai yang Direncanakan (PV):* Persentase total perkiraan upaya yang diberikan pada tugas pekerjaan tertentu; PV = BCW/BAC.
- *Anggaran Biaya Pekerjaan yang Dilakukan (BCWP):* Jumlah perkiraan upaya untuk tugas pekerjaan yang telah diselesaikan pada waktu yang ditentukan.
- *Biaya Aktual Pekerjaan yang Dilakukan (ACWP)*: Jumlah upaya aktual untuk tugas pekerjaan yang telah diselesaikan.

Indikator Kemajuan

- Nilai Perolehan (EV) = BCWP/BAC
  = Jumlah PV untuk semua tugas pekerjaan yang diselesaikan
  = PC = Persentase selesai
- Indeks Kinerja Jadwal (SPI) = BCWP/BCWS
- Varians Jadwal (SV) = BCWP — BCWS
- Indeks Kinerja Biaya (CPI) = BCWP/ACWP
- Varians Biaya (CV) = BCWP — ACWP

Contoh 3.5

Perusahaan LMN sedang menyelesaikan proyeknya. Log pekerjaan di bawah menunjukkan status proyek saat ini.

Tugas	Perkiraan Pengerjaan (hari)	Upaya Aktual Sampai saat ini (Hari)	Perkiraan Tanggal Penyelesaian	Tanggal Sebenarnya selesai
1	5	10	25 Januari	1 Februari
2	25	20	15 Januari	15 Februari
3	120	80	15 Mei	
4	40	50	15 April	1 April
5	60	50	1 Juli	
6	80	70	1 September	

BAC adalah jumlah estimasi. BAC = 330 hari. BAC adalah perkiraan total pekerjaan. Pada 1/4/01, tugas 1,2, dan 4 telah selesai. BCWP adalah jumlah BCWS untuk tugas-tugas tersebut. Jadi BCWP adalah 70 hari. Nilai yang diperoleh (EV) adalah 70/330 atau 21,2 persen. Pada tanggal 4/1/01 tugas 1 dan 2 dijadwalkan untuk diselesaikan dan 1,2, dan 4 benar-benar selesai. Jadi BCWP adalah 70 hari dan BCWS adalah 30 hari. Dengan demikian, SPI-nya 70/30 atau 233 persen. SV = 70 hari —30 hari = +40 hari, atau 40 hari programmer ke depan. ACWP adalah jumlah upaya aktual untuk tugas 1, 2, dan 4. Jadi, ACWP adalah 80 hari programmer. CPI adalah 70/80 = 87,5 persen. CV = 70 hari programmer —80 hari programmer = —10 hari programmer, atau tertinggal 10 hari programmer.

Contoh 3.6

Pada 1/7/01, asumsikan bahwa tugas 3 juga telah diselesaikan dengan upaya aktual selama 140 hari, sehingga BCWP adalah 190 dan EV adalah 190/330, atau 57,5 persen. Pada 1/7/01, tugas 1, 2, 3, dan 4 sebenarnya telah selesai. Jadi BCWP adalah 190 hari dan BCWS adalah 250 hari. Jadi SPInya 190/250 = 76 persen. SV-nya adalah 190 hari pemrogram —250 hari pemrogram = —60 hari pemrogram, atau tertinggal 60 hari pemrogram. ACWP adalah jumlah upaya aktual untuk 1, 2, 3, dan 4. Jadi ACWP adalah 220 hari programmer.

Tugas 1 sampai 5 dijadwalkan selesai, namun hanya 1 sampai 4 yang benar-benar selesai. CPI adalah 190/220 = 86,3 persen, dan CV 190–220, atau tertinggal 30 hari programmer.

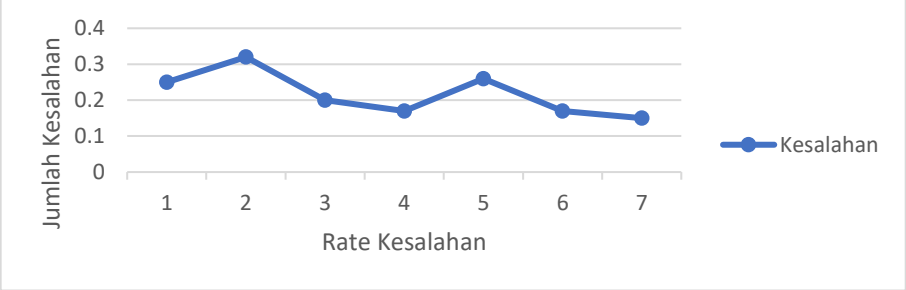
3.5 PELACAKAN KESALAHAN

Salah satu praktik manajemen yang sangat baik adalah pelacakan kesalahan, yaitu mencatat kesalahan yang telah terjadi dan waktu antar kesalahan (waktu antara terjadinya kesalahan). Hal ini dapat digunakan untuk membuat keputusan tentang kapan akan merilis perangkat lunak. Efek tambahan dari pelacakan dan publikasi tingkat kesalahan adalah membuat pengembang perangkat lunak sadar akan pentingnya kesalahan dan pengurangan kesalahan. Dampak perubahan pada proses perangkat lunak dapat dilihat pada data error. Selain itu, membuat kesalahan dan deteksi kesalahan terlihat mendorong penguji dan pengembang untuk menjadikan pengurangan kesalahan sebagai tujuan.

Tingkat kesalahan adalah kebalikan dari waktu antar kesalahan. Artinya, jika kesalahan terjadi setiap 2 hari sekali, maka tingkat kesalahan sesaat adalah 0,5 kesalahan/hari. Tingkat kesalahan sesaat saat ini merupakan perkiraan yang baik untuk tingkat kesalahan saat ini. Jika kesalahan yang menyebabkan kesalahan tidak dihilangkan ketika kesalahan ditemukan, maka tingkat kesalahan kumulatif (jumlah semua kesalahan yang ditemukan dibagi total waktu) merupakan perkiraan yang baik untuk tingkat kesalahan di masa depan. Biasanya, sebagian besar kesalahan diperbaiki (kesalahan dihilangkan), sehingga tingkat kesalahan akan turun dan waktu antar kesalahan akan meningkat. Plot data ini dapat menunjukkan tren tingkat kesalahan (kesalahan yang ditemukan per satuan waktu). Memasangkan garis lurus pada titik-titik adalah cara yang efektif untuk menampilkan tren. Tren ini dapat digunakan untuk memperkirakan tingkat kesalahan di masa depan. Ketika tren melintasi sumbu x, perkiraan tingkat kesalahan adalah nol, atau tidak ada lagi kesalahan. Jika sumbu x adalah jumlah kesalahan, maka nilai titik potong x dapat digunakan sebagai perkiraan jumlah kesalahan pada perangkat lunak. Jika sumbu x adalah waktu pengujian yang telah berlalu, intersep adalah perkiraan waktu pengujian yang diperlukan untuk menghilangkan semua kesalahan. Area di bawah garis terakhir ini merupakan perkiraan jumlah kesalahan pada perangkat lunak.

Contoh 3.7

Pertimbangkan data kesalahan berikut (diberikan sebagai waktu antar kesalahan): 4, 3, 5, 6, 4, 6, 7. Tingkat kesalahan sesaat adalah kebalikan dari waktu antar kesalahan: 0,25, 0,33, 0,20, 0,17, 0,25, 0,17, dan 0,14. Dengan memplot angka kesalahan tersebut akan menghasilkan kurva ke bawah, seperti yang ditunjukkan pada Gambar 3-2. Hal ini menunjukkan bahwa tingkat kesalahan sebenarnya menurun.



Gambar 3-2. Plot tingkat kesalahan.

Sebuah garis lurus yang melalui titik-titik tersebut akan memotong sumbu sekitar 11. Karena ini berarti bahwa tingkat kesalahan akan menjadi nol pada kesalahan kesebelas, perkiraan jumlah kesalahan dalam perangkat lunak ini adalah 11 kesalahan. Karena tujuh kesalahan telah ditemukan, hal ini menunjukkan bahwa mungkin ada empat kesalahan lagi dalam perangkat lunak.

3.6 POSTMORTEM

Salah satu aspek penting dalam pengembangan perangkat lunak adalah belajar dari kesalahan dan kesuksesan Anda. Dalam pengembangan perangkat lunak, ini disebut postmortem. Ini terdiri dari pengumpulan orang-orang penting dari kelompok pengembangan dan pengguna. Masalah terdiri dari kualitas, jadwal, dan proses perangkat lunak. Penting bagi setiap orang untuk bebas menyampaikan pendapat. Laporan formal perlu dibuat dan didistribusikan. Laporan tidak boleh disanitasi.

Contoh 3.8

Perusahaan JKL membuat laporan postmortem yang ditunjukkan:

<table>
<tr><td>Nama Proyek:
Proyek X</td><td>Tanggal Mulai:
5 September 2023</td><td>Tanggal Penyelesaian:
8 Desember 2023</td></tr>
<tr><td>Langkah Penyelesaian</td><td>Ukuran:<table><tr><th>Perkiraan</th><th>Sebenarnya</th></tr><tr><td>3000 LOC</td><td>5000 LOC</td></tr></table></td><td>Upaya:<table><tr><th>Perkiraan</th><th>Sebenarnya</th></tr><tr><td>12.000 Menit</td><td>10.000 Menit</td></tr></table></td></tr>
<tr><td>Komentar subjektif tentang estimasi</td><td>Baik:
Upaya Hampir Seluruhnya</td><td>Buruk:
Implikasi Buruk diremehkan</td></tr>
<tr><td>Komentar subjektif tentang Proses</td><td>Baik</td><td>Buruk:
Anggota tim tidak menyelesaikan tugas tepat waktu</td></tr>
<tr><td>Komentar subjektif tentang Jadwal</td><td>Baik</td><td>Buruk:
Kurangnya waktu untuk penyelesaian</td></tr>
<tr><td>Kualitas</td><td>Error ditemukan<table><tr><th>Req</th><th>Desain</th><th>Unit</th><th>Integ</th><th>Postdel</th></tr><tr><td></td><td></td><td></td><td>30</td><td></td></tr></table></td><td><table><tr><th></th><th>Ave</th><th>Maks</th></tr><tr><td>McCabe</td><td>4</td><td>30</td></tr><tr><td>Metode/Kelas</td><td>6</td><td>10</td></tr><tr><td>Atribut/Kelas</td><td>10</td><td>15</td></tr><tr><td>LOC/Kelas</td><td>150</td><td>500</td></tr></table></td></tr>
<tr><td>Komentar subjektif tentang Kualitas</td><td>Baik</td><td>Buruk
Sistem tidak diuji dengan baik</td></tr>
<tr><td>Masalah:
Pernyataan awal</td><td>Keterangan:
Format file masukan awal salah</td><td>Dampak:
2 Wock Terbuang</td></tr>
<tr><td>Masalah:</td><td>Keterangan:</td><td>Dampak:</td></tr>
<tr><td>Masalah:</td><td>Keterangan:</td><td>Dampak:</td></tr>
<tr><td>Masalah:</td><td>Keterangan:</td><td>Dampak:</td></tr>
</table>

LATIHAN SOAL

1. Apa yang dimaksud dengan visibilitas?
2. Apa perbedaan antara pendekatan proses dan pendekatan proyek?
3. Untuk proyek baru yang sangat berbeda dari proyek sebelumnya, apakah pendekatan proses atau manajemen proyek akan lebih baik?
4. Apa keuntungan membuat banyak tugas kecil yang hanya berukuran ''inci-kerikil''?
5. Ukuran kemajuan nilai perolehan apa yang dapat dikurangi selama suatu proyek?
6. Pengukuran kemajuan nilai yang diperoleh manakah yang nilainya lebih besar dari 1 barang?
7. Apa keuntungan menggunakan invers SPI dan CPI?

Latihan Soal Tambahan (Diskusi)

1. Gambarkan model proses untuk tim yang memiliki struktur lemah dan bergantung pada diskusi tim untuk menetapkan arah dan menyelesaikan masalah.
2. Dengan menggunakan log waktu berikut, hitung produktivitas pemrogram dalam LOC/hari. Asumsikan proyek 1 adalah 120 LOC dan proyek 2 adalah 80 LOC.

Tanggal	Mulai	Berhenti	Istirahat	Delta	Tugas
2 Januari 2023	08:30	16:30	60 Menit Makan Siang		Projek 1 Coding
2 Februari 2023	09:00	17:00	30 menit Makan Siang		Projek 1 Coding
2 Mei 2023	09:00	17:30	30 Menit Makan Siang, 60 menit Meeting		Projek 2 Coding
2 Juni 2023	07:30	12:00			Projek 2 Coding

3. Dengan menggunakan log pekerjaan berikut, hitung semua ukuran dasar dan indikator kemajuan. Apakah proyeknya sesuai jadwal? Asumsikan saat ini tanggal 5 Januari 2024.

Tugas	Perkiraan Pengerjaan (hari)	Upaya Aktual Sampai saat ini (Hari)	Perkiraan Tanggal Penyelesaian	Tanggal Sebenarnya selesai
1	50	70	15 Januari 2024	1 Februari 2024
2	35	20	15 Februari 2024	15 Februari 2024
3	20	40	25 Februari 2024	1 Maret 2024
4	40	40	15 April 2024	1 April 2024
5	60	10	1 Juni 2024	
6	80	20	1 Juli 2024	

4. Gunakan spreadsheet untuk menghitung PV dan indikator kemajuan proyek berikutnya dengan interval setengah bulan dari 1 Januari hingga 1 September.

Tugas	Perkiraan Pengerjaan (hari)	Upaya Aktual Sampai saat ini (Hari)	Perkiraan Tanggal Penyelesaian	Tanggal Sebenarnya selesai
1	30	37	1 Januari	1 Februari
2	25	24	2 Februari	15 Februari
3	30	41	1 Maret	15 Maret
4	50	47	15 April	1 April
5	60	63	1 Mei	15 April
6	35	31	15 Mei	1 Juni
7	55	58	1 Juni	1 Juni

8	30	28	15 Juni	15 Juni
9	45	43	1 Juli	15 Juli
10	25	29	1 Agustus	15 Agustus
11	45	49	15 Agustus	1 September

5. Seorang profesor mempunyai 40 pekerjaan rumah dan 40 ujian yang harus dinilai. Ujian biasanya memakan waktu 3 kali lebih lama untuk menilai tugas pekerjaan rumah. Hitung PV untuk setiap pekerjaan rumah dan untuk setiap ujian. Setelah 5 jam, jika profesor telah menyelesaikan setengah dari ujiannya, berapa lama waktu yang dibutuhkan untuk menyelesaikan penilaiannya?
6. Dengan mengetahui waktu antar-kesalahan berikut (yaitu, waktu antara terjadinya kesalahan), gunakan plot untuk memperkirakan jumlah total kesalahan awal dan waktu untuk menghilangkan seluruh kesalahan: 6, 4, 8, 5, 6, 9, 11, 14, 16, 19.
7. Proyek dimulai pada tanggal 1 Januari dan harus selesai pada tanggal 1 Juni. Sekarang tanggal 1 Maret. Lengkapi tabel berikut. Hitung EV, SPI, SV, dan CV. Tentukan apakah proyek tersebut tepat waktu. Benarkan jawaban Anda. Tunjukkan pekerjaan Anda.

Pekerjaan	Estimasi waktu	Waktu sebenarnya yang dibutuhkan	PV	Tenggat Waktu	Lengkap
1	30	10		1 Februari	
2	20	30		1 Maret	Ya
3	50	30		1 Mei	Ya
4	100	5		1 Juni	

BAB 4
PERENCANAAN PROYEK PERANGKAT LUNAK

4.1 PERENCANAAN PROYEK

Perencanaan sangat penting dan pengembangan perangkat lunak tidak terkecuali. Mencapai keberhasilan dalam pengembangan perangkat lunak memerlukan perencanaan. Perencanaan proyek perangkat lunak melibatkan penentuan tugas apa yang perlu dilakukan, bagaimana cara melakukan tugas tersebut, dan sumber daya apa yang diperlukan untuk menyelesaikan tugas tersebut.

WBS—Struktur Perincian Kerja

Salah satu tugas pertama adalah memecah tugas-tugas besar menjadi tugas-tugas kecil. Ini berarti menemukan bagian-bagian tugas yang dapat diidentifikasi. Hal ini juga berarti menemukan hasil dan pencapaian yang dapat digunakan untuk mengukur kemajuan.

Struktur rincian kerja (WBS) harus berupa struktur pohon. Perincian tingkat atas biasanya sesuai dengan model siklus hidup (LCM) yang digunakan dalam organisasi. Perincian tingkat berikutnya dapat disesuaikan dengan proses dalam proses organisasi model (PM). Tingkat selanjutnya digunakan untuk mempartisi tugas menjadi tugas-tugas yang lebih kecil dan lebih mudah dikelola.

Berikut ini adalah aturan untuk membangun struktur rincian kerja yang tepat:

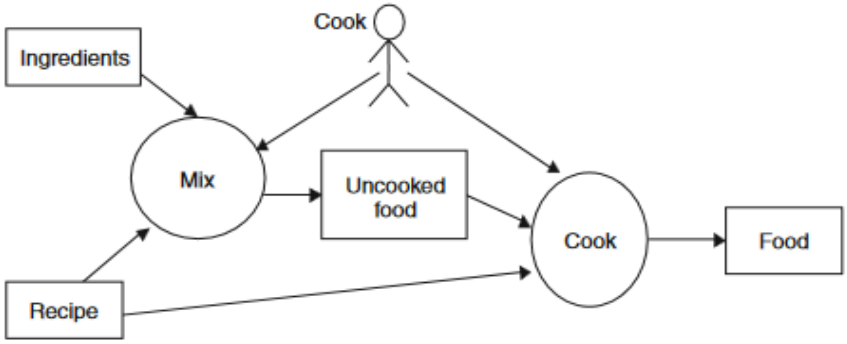
1. WBS harus berupa struktur pohon. Seharusnya tidak ada loop atau siklus di WBS. Tindakan berulang akan ditampilkan dalam model proses dan/atau model siklus hidup.
2. Setiap uraian tugas dan penyampaian harus dapat dimengerti dan tidak ambigu. Tujuan WBS adalah komunikasi dengan anggota tim. Jika anggota tim salah mengartikan tugas atau hasil yang seharusnya, akan timbul masalah.
3. Setiap tugas harus mempunyai kriteria penyelesaian (sering kali berupa hasil). Harus ada cara untuk memutuskan kapan suatu tugas selesai, karena subtugas yang tidak memiliki akhir yang pasti mendorong harapan kemajuan yang salah. Keputusan ini disebut kriteria penyelesaian. Hal ini dapat berupa penyampaian, misalnya, desain proyek yang lengkap, dan tinjauan sejawat dapat memutuskan apakah proyek tersebut sudah selesai.
4. Semua kiriman (artefak) harus diidentifikasi. Sebuah penyampaian harus dihasilkan oleh beberapa tugas atau tidak akan diproduksi.
5. Penyelesaian tugas yang positif harus berarti selesainya seluruh tugas. Tujuan dari jadwal rincian kerja adalah untuk mengidentifikasi subtugas yang diperlukan untuk menyelesaikan seluruh tugas. Jika tugas atau hasil penting hilang, keseluruhan tugas tidak akan terselesaikan.

Contoh 4.1

Dalam pembuatan sepotong roti, model siklus hidup memasaknya mungkin seperti ditunjukkan pada Gambar 4-1.

Gambar 4-1. Model siklus hidup untuk memasak.

Model proses memasak mungkin seperti yang ditunjukkan pada Gambar 4-2.



Gambar 4-2. Model proses memasak.

Subtugasnya mungkin sebagai berikut:

Pilih bahan, Periksa bahan, Kumpulkan bahan, Tambahkan cairan, Tambahkan ragi, Tambahkan sedikit tepung, Buat bolu (ragi dan cairan), Biarkan mengembang Gambar 4-2. Model proses memasak.

Subtugasnya mungkin sebagai berikut:

Pilih bahan, Periksa bahan, Kumpulkan bahan, Tambahkan cairan, Tambahkan ragi, Tambahkan sedikit tepung, Buat bolu (ragi dan cairan), Biarkan mengembang

Pilih makanan

Pilih bahan	Daftar bahan
Periksa bahan-bahannya	Daftar belanja

Kumpulkan bahan-bahannya

Kumpulkan bahan-bahannya	Bahan-bahan yang dirakit

Memasak makanan

Mencampur

Tambahkan cairan	Cairan dalam mangkuk
Meragikan	Cairan dengan ragi
Tambahkan sedikit tepung	Cairan dan tepung
Membuat bolu (ragi dan cairan)	Spons
Biarkan bangkit pertama kali	Spons yang terangkat
Tambahkan sisa tepung dan uleni	Adonan diuleni
Biarkan mengembang untuk kedua kalinya	Adonan yang sudah mengembang

Bentuk menjadi roti	Roti
Biarkan bangkit untuk ketiga kalinya	Roti yang sudah bangkit

Memasak

Memanggang	Roti

Makan

Mengiris	Potongan roti
Mentega	Irisan mentega
Makan	Selera yang baik

Membersihkan	
Membersihkan	Dapur bersih

Contoh 4.2

Tim XYZ ingin mengembangkan sistem pengenalan wajah untuk digunakan pada robot. Sistem ini dimaksudkan untuk menyambut pengunjung laboratorium robotika. Ia harus mengenali wajah-wajah yang pernah dilihatnya sebelumnya dengan keandalan yang wajar. Bagian pertama dari rincian pekerjaan mungkin mengenali subtugas berikut:

Kelayakan

Menentukan kelayakan visi.
Tentukan ketersediaan kamera dan perangkat lunak.
Jadwalkan akuisisi perangkat lunak kamera dan visi.

Analisis resiko

Tentukan risiko penglihatan.

Persyaratan

Tentukan persyaratan.

Desain

Prototipe desain.
Visi prototipe.

Penerapan

Kode pengambilan gambar.
Kode pemrosesan gambar.
Kode perbandingan gambar.
Integrasikan dengan perangkat lunak robot lainnya.

Pengujian

Uji pengambilan gambar.

Pengiriman

Dokumen.

Beberapa subtugas ini masih berada pada level yang sangat tinggi. Tugas-tugas ini mungkin tidak memiliki hasil yang jelas dan dapat diperiksa. Artinya, mungkin tidak mudah

untuk menentukan secara pasti kapan suatu subtugas telah diselesaikan. Subtugas tidak cocok untuk diselesaikan jika pengembang merasa telah selesai. Harus ada cara untuk menentukan secara obyektif kapan suatu subtugas telah diselesaikan dengan benar.

Contoh 4.3

Tim membagi subtugas "Kodekan pengambilan gambar" menjadi serangkaian subtugas yang lebih rinci, dengan setiap subtugas baru memiliki kriteria penyampaian dan penyelesaian yang lebih spesifik. Kumpulan subtugas dan kiriman adalah sebagai berikut:

Instal driver kamera komersial.	*Driver terinstal*
Uji driver dari windows dan simpan gambar ke file.	*File Gambar*
Tulis rutin untuk memanggil driver dari C++.	*Rutin*
Uji rutin C++ secara terpisah dan simpan gambar ke file.	*Gambar dari kode C++*
Uji rutin C++ dari perangkat lunak kontrol robot utama dan ambil gambar.	*Gambar dari utama*

4.2 PERT—TEKNIK EVALUASI DAN TINJAUAN PROGRAM

Teknik ini membuat grafik yang menunjukkan ketergantungan antar tugas. Setiap tugas mempunyai perkiraan waktu yang diperlukan untuk menyelesaikan tugas dan daftar tugas lain yang harus diselesaikan sebelum tugas ini dapat dimulai (ketergantungan). Grafik mungkin tidak selalu hanya memiliki satu subtugas awal atau hanya satu subtugas akhir. Keseluruhan tugas hanya selesai jika semua subtugas telah diselesaikan. Grafik tersebut dapat digunakan untuk menghitung waktu penyelesaian seluruh subtugas, waktu penyelesaian minimum seluruh tugas, dan jalur kritis subtugas.

ALGORITMA WAKTU PENYELESAIAN

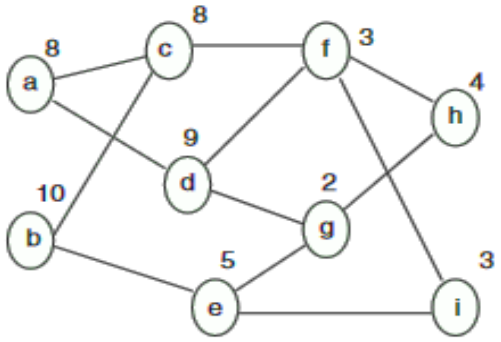
1. Untuk setiap node, lakukan langkah 1.1 (sampai waktu penyelesaian semua node dihitung)
   1.1 Jika pendahulunya telah selesai, ambil waktu penyelesaian terakhir dari pendahulunya dan tambahkan waktu yang diperlukan untuk simpul ini.
2. Node dengan waktu penyelesaian paling akhir menentukan waktu penyelesaian proyek yang paling awal.

CONTOH 4.4

Terapkan algoritma ini pada Tabel 4-1, yang menunjukkan contoh tugas dan dependensi. Ketergantungan yang sama ditunjukkan pada Gambar 4-3. Untuk menerapkan algoritma waktu penyelesaian, mulailah dengan subtugas a; ia tidak memiliki ketergantungan, sehingga dapat dimulai pada waktu awal (misalnya, 0). Subtugas ini dapat diselesaikan pada waktu 0 + 8 = 8. Demikian pula, subtugas b dapat diselesaikan pada waktu 0 + 10 = 10. Lihat Tabel 4-2. Perhatikan bahwa karena subtugas ini tidak bergantung satu sama lain atau pada hal lain, maka subtugas tersebut dapat dimulai pada waktu 0. Waktu penyelesaiannya dihitung tanpa memperhatikan kekurangan sumber daya. Artinya, untuk perhitungan penyelesaian ini, diasumsikan bahwa ada orang yang tersedia untuk melakukan kedua tugas tersebut pada waktu yang bersamaan.

Tabel 4.1 Subtugas

ID Subtask	Waktu yang dibutuhkan	Dependensi
a	8	
b	10	
c	8	a,b
d	9	a
e	5	b
f	3	c,d
g	2	d
h	4	f,g
i	3	e,f



Gambar 4-3. diagram PERT.

Karena waktu penyelesaian untuk subtugas a dan b kini telah dihitung, waktu penyelesaian untuk node c, d, dan e dapat dihitung. Karena pendahulu c selesai pada jam 8 dan 10, subtugas c dapat dimulai pada jam 10 dan selesai pada 10 + 8 = 16. Waktu mulai untuk d adalah 8 dan waktu penyelesaian dapat berupa 8 + 9 = 17, dan untuk e kali akan menjadi 10 dan 10 + 5 = 14.

Sekarang kita dapat memproses subtugas f dan g. Waktu mulai masing-masing bisa 17 dan 16. Waktu penyelesaiannya adalah 17 + 3 = 20 untuk f dan 16 + 2 = 18 untuk g. Subtugas h dan i sekarang dapat dihitung dengan memulai pada 21 dan h bersaing pada 25 dan i pada 24. Tabel 4-2 memuat semua waktu mulai dan selesai.

Tabel 4-2 Waktu Penyelesaian Sub Tugas

ID Subtask	Waktu mulai	Waktu Penuh	Jalur Kritis
a	0	8	
b	0	10	*
c	10	18	*
d	8	17	
e	10	15	
f	18	21	*
g	17	19	
h	21	25	*
i	21	24	

Jalur Kritis

Jalur kritis adalah kumpulan tugas yang menentukan waktu penyelesaian sesingkat mungkin. Waktu penyelesaian akan lebih lama jika sumber daya tidak mencukupi untuk melakukan seluruh aktivitas paralel. Namun, waktu penyelesaian tidak dapat dipersingkat dengan menambahkan lebih banyak sumber daya.

Algoritma Penandaan Jalur Kritis

1. Mulailah dengan node dengan waktu penyelesaian terakhir; tandai itu (mereka) sebagai kritis.
2. Pilih node pendahulu dari node kritis dengan waktu penyelesaian terakhir; tandai itu (mereka) sebagai kritis. Lanjutkan Langkah 2 hingga mencapai node awal.

Contoh 4.5

Pada Tabel 4-2 kita dapat melihat waktu penyelesaian seluruh subtugas. Subtugas h memiliki waktu penyelesaian paling lambat, 25. Oleh karena itu, kita menandai h sebagai bagian dari jalur kritis. Pendahulu dari h adalah f dan g. Subtugas f mempunyai waktu penyelesaian paling akhir dari kedua subtugas tersebut, sehingga f ditandai sebagai bagian dari jalur kritis.

Subtugas f memiliki c dan d sebagai pendahulunya. Karena c mempunyai waktu penyelesaian yang lebih lambat, c ditandai sebagai bagian dari jalur kritis. Subtugas c memiliki a dan b sebagai pendahulunya, dan karena b memiliki waktu berikutnya, maka subtugas tersebut merupakan bagian dari jalur kritis. Karena kita sekarang berada pada subtugas awal, jalur kritis telah selesai.

Waktu Slack

Subtugas yang berada pada jalur kritis harus dimulai sedini mungkin atau keseluruhan proyek akan tertunda. Namun, subtugas yang tidak berada pada jalur kritis mempunyai fleksibilitas pada saat dimulainya. Fleksibilitas ini disebut waktu Slack

Algoritma Untuk Waktu Slack

1. Pilih node nonkritis dengan waktu berakhir terakhir yang belum diproses. Jika subtugas tidak memiliki penerus, pilih waktu berakhir terakhir dari semua node. Jika subtugas memiliki penerus, pilih waktu mulai paling awal dari node penerus. Ini adalah waktu penyelesaian terakhir untuk subtugas ini. Buatlah waktu mulai paling akhir untuk subtugas ini agar mencerminkan waktu ini.
2. Ulangi Langkah 1 hingga semua subtugas jalur nonkritis telah diproses.

Contoh 4.6

Subtugas nonkritis yang waktu penyelesaiannya paling lambat adalah subtugas i. Karena tidak mempunyai penerus, waktu penyelesaian terakhir, 25, digunakan. Ini ditambahkan sebagai waktu penyelesaian terakhir untuk i. Karena 25 adalah 1 lebih lambat dari 24, waktu mulai diubah dari 21 menjadi 21,22. Sekarang subtugas non-kritis dan belum diproses terbaru adalah g. Karena h satu-satunya penerus g, dan h harus dimulai pada 21, g harus berakhir pada 21. Jadi waktu penyelesaian g menjadi 19,21 dan waktu mulai menjadi 17,19. Subtugas berikutnya yang akan diproses adalah d. Ia memiliki penerus f dan g. Subtugas f harus dimulai pada jam 18, sehingga penyelesaian d menjadi 17,18, dan permulaannya

menjadi 8,9. Subtugas berikutnya yang akan diproses adalah e. Subtugas e memiliki g dan i sebagai penerusnya. Waktu mulai terakhir subtugas g adalah 19 dan i adalah 22, sehingga subtugas e menjadi 10,14 untuk waktu mulai dan 15,19 untuk waktu penyelesaian. Subtugas terakhir yang diproses adalah a. Ia memiliki penerus c dan d. Subtugas a harus diselesaikan pada jam 9, sehingga waktu penyelesaiannya adalah 8,9, dan waktu mulainya adalah 0,1. Tabel 4-3 merangkum hasilnya.

Tabel 4-3 Subtugas dengan Waktu Slack

ID Subtask	Waktu mulai	Waktu Penuh	Jalur Kritis
a	0,1	8,9	
b	0	10	*
c	10	18	*
d	8,9	17,18	
e	10,14	15,19	
f	18	21	*
g	17,19	19,21	
h	21	25	*
i	21,22	24,25	

Contoh 4.7 Menggunakan Microsoft Project

Buka Proyek MS (versi 98). Pilih tampilan PERT di menu sebelah kiri. Di bawah menu tarik-turun Sisipkan, sisipkan tugas baru. Gunakan tombol kanan mouse untuk membuka informasi tugas. Ubah waktu pengerjaan tugas menjadi 5 hari. Seret dari tugas ini untuk membuat tugas baru dengan ketergantungan pada tugas pertama, atau buka menu Sisipkan untuk menyisipkan tugas baru lainnya. Buat tugas dan dependensi dari Contoh 4.5. Tugas di jalur kritis akan ditampilkan dengan warna merah. Gunakan bilah menu kiri untuk melihat tampilan bagan Gannt proyek ini.

4.3 ESTIMASI BIAYA PERANGKAT LUNAK

Tugas estimasi biaya perangkat lunak adalah menentukan berapa banyak sumber daya yang dibutuhkan untuk menyelesaikan proyek. Biasanya perkiraan ini dalam satuan programmer-bulan (PM).

Ada dua pendekatan yang sangat berbeda dalam estimasi biaya. Pendekatan yang lebih tua disebut estimasi LOC, karena didasarkan pada perkiraan awal jumlah baris kode yang perlu dikembangkan untuk proyek tersebut. Pendekatan yang lebih baru didasarkan pada penghitungan titik fungsi dalam deskripsi proyek.

ESTIMASI GARIS KODE (LOC)

Langkah pertama dalam estimasi berbasis LOC adalah memperkirakan jumlah baris kode dalam proyek yang telah selesai. Hal ini dapat dilakukan berdasarkan pengalaman, ukuran proyek sebelumnya, ukuran solusi pesaing, atau dengan memecah proyek menjadi bagian-bagian yang lebih kecil dan kemudian memperkirakan ukuran masing-masing bagian yang lebih kecil. Pendekatan standarnya adalah, untuk setiap potongan, memperkirakan

ukuran maksimum yang mungkin, max_i, ukuran minimum yang mungkin, min_i, dan ukuran ''tebakan terbaik'', besti. Estimasi keseluruhan proyek adalah 1/6 dari jumlah maksimum, minimum, dan 4 kali tebakan terbaik:

$$\text{Standart deviasi dari } S = (sd^2 + sd^2 + \ldots + sd^2)^{1/2}$$
$$\text{Standart deviasi dari E(i)} = (max - min)/6$$

Contoh 4.8

Tim WRT telah mengidentifikasi tujuh subbagian proyek mereka. Hal ini ditunjukkan pada Tabel 4-4 beserta perkiraan ukuran masing-masing subbagian.

Tabel 4-4 Perkiraan Ukuran Subbagian (dalam LOC)

Bagian	Max Size	Best Guess	Min Size
1	20	30	50
2	10	15	25
3	25	30	45
4	30	35	40
5	15	20	25
6	10	12	14
7	20	22	25

Perkiraan untuk setiap bagian adalah sebagai berikut:

$$\text{p1}(20 + 4 * 30 + 50)/6 = 190/6 = 31.6$$
$$\text{p2}(10 + 4 * 15 + 25)/6 = 95/6 = 15.8$$
$$\text{p3}(25 + 4 * 30 + 45)/6 = 190/6 = 31.6$$
$$\text{p4}(30 + 4 * 35 + 40)/6 = 220/6 = 36.7$$
$$\text{p5}(15 + 4 * 20 + 25)/6 = 120/6 = 20$$
$$\text{p6}(10 + 4 * 12 + 14)/6 = 72/6 = 12$$
$$\text{p7}(20 + 4 * 22 + 25)/6 = 133/6 = 22.17$$

Perkiraan untuk keseluruhan proyek adalah jumlah perkiraan untuk setiap bagian:

$$\text{Whole} = 31.6 + 15.8 + 31.6 + 36.7 + 20 + 12 + 22.17 = 170.07 \text{ LOC}$$

Estimasi standar deviasi estimasi tersebut adalah sebagai berikut:

$$Standard\ deviation$$
$$= ((50 - 20)^2 + (25 - 10)^2 + (45 - 25)^2 + (40 - 30)^2 + (25 - 15)^2 + (14 - 10)^2 + (25 - 20)^2)^{-5}$$
$$= (900 + 225 + 400 + 100 + 100 + 16 + 25)^{-5}$$
$$= 1766.5$$
$$= 42.03$$

Estimasi Biaya Berbasis Loc

Pendekatan dasar LOC adalah rumus yang mencocokkan data historis. Rumus dasarnya memiliki tiga parameter:

$$\text{Cost} = \alpha * \text{KLOC}^{\beta} + \gamma$$

Alpha, α, adalah biaya marjinal per KLOC (seribu baris kode). Ini adalah biaya tambahan untuk tambahan seribu baris kode. Parameter beta, β, adalah eksponen yang mencerminkan nonlinier hubungan. Nilai beta yang lebih besar dari 1 berarti biaya per KLOC meningkat seiring dengan bertambahnya ukuran proyek. Ini adalah skala disekonomis. Nilai beta kurang dari 1 mencerminkan skala ekonomi. Beberapa penelitian menemukan beta lebih besar dari 1, dan penelitian lain menemukan beta kurang dari 1. Parameter gamma, γ, mencerminkan biaya tetap dalam melaksanakan proyek apa pun. Penelitian telah menemukan gamma positif dan gamma nol.

Contoh 4.9

Perusahaan LMN telah mencatat data berikut dari proyek-proyek sebelumnya. Perkirakan parameter apa yang harus digunakan untuk rumus estimasi biaya dan berapa banyak upaya yang harus dilakukan untuk proyek baru sebesar 30 KLOC (lihat Tabel 4-5).

Tabel 4-5 Data Historis

ID Project	Ukuran (KLOC)	Usaha (PM)
1	50	120
2	80	192
3	40	96
4	10	24
5	20	48

Menganalisis atau memplot data ini akan menunjukkan hubungan linier antara ukuran dan usaha. Kemiringan garisnya adalah 2,4. Ini akan menjadi alpha, α, dalam rumus estimasi biaya berbasis LOC. Karena garisnya lurus (hubungan linier), beta, β, adalah 1. Nilai gamma, γ, akan menjadi nol.

Model Biaya Konstruktif (COCOMO)

COCOMO adalah rumus estimasi biaya LOC klasik. Itu diciptakan oleh Barry Boehm pada tahun 1970-an. Ia menggunakan seribu instruksi sumber yang disampaikan (KDSI) sebagai satuan ukurannya. KLOC setara. Unit usahanya adalah programmer-month (PM).
Boehm membagi data proyek historis menjadi tiga jenis proyek:

1. Aplikasi (terpisah, organik, misalnya pengolahan data, ilmiah)
2. Program utilitas (setengah terikat, misalnya kompiler, penghubung, penganalisis)
3. Program sistem (tertanam)

Dia menentukan nilai parameter model biaya untuk menentukan upaya:

1. Program aplikasi: PM = 2.4 * $(\text{KDSI})^{1.05}$
2. Program utilitas: PM = 3.0 * $(\text{KDSI})^{1.12}$

3. Program sistem: PM = 3.6 * $(\mathrm{KDSI})^{1.20}$

Contoh 4.10

Hitung upaya programmer untuk proyek dari 5 hingga 50 KDSI (lihat Tabel 4-6)

Tabel 4-6 Upaya COCOMO

Ukuran	Aplikasi	Util	Sistem
5K	13.0	18.2	24.8
10K	26.9	39.5	57.1
15K	41.2	62.2	92.8
20K	55.8	86.0	131.1
25K	70.5	110.4	171.3
30K	85.3	135.3	213.2
35K	100.3	160.8	256.6
40K	115.4	186.8	301.1
45K	130.6	213.2	346.9
50K	145.9	239.9	393.6

Boehm juga menetapkan bahwa dalam data proyeknya terdapat standar waktu pengembangan berdasarkan jenis proyek dan besarnya proyek. Berikut rumus waktu pengembangan (TDEV) dalam programmer-bulan:

1. Program aplikasi: TDEV = 2.5 * $(\mathrm{PM})^{0.38}$
2. Program utilitas: TDEV = 2.5 * $(\mathrm{PM})^{0.35}$
3. Program sistem: TDEV = 2.5 * $(\mathrm{PM})^{0.32}$

Contoh 4.11

Hitung TDEV standar menggunakan rumus COCOMO untuk proyek dari 5 hingga 50 KDSI (lihat Tabel 4-7).

Tabel 4-7 COCOMO
Waktu Pengembangan

Ukuran	Aplikasi	Util	Sistem
5K	6.63	6.90	6.99
10K	8.74	9.06	9.12
15K	10.27	10.62	10.66
20K	11.52	11.88	11.9
25K	12.60	12.97	12.96
30K	13.55	13.93	13.91
35K	14.40	14.8	14.75
40K	15.19	15.59	15.53
45K	15.92	16.33	16.25
50K	16.61	17.02	16.92

Analisis Titik Fungsi

Ide titik fungsi adalah untuk mengidentifikasi dan mengukur fungsionalitas yang diperlukan untuk proyek tersebut. Idenya adalah untuk menghitung hal-hal dalam perilaku

eksternal yang memerlukan pemrosesan. Item klasik yang perlu dihitung adalah sebagai berikut:

- *Inquiries* adalah pasangan permintaan-respons yang tidak mengubah data internal. Misalnya, permintaan alamat karyawan tertentu adalah sebuah penyelidikan. Seluruh rangkaian menanyakan, memberikan nama, dan mendapatkan alamat akan dihitung sebagai satu pertanyaan.
- *Input* adalah item data aplikasi yang disuplai ke program. Input logis biasanya dianggap sebagai satu item dan masing-masing kolom biasanya tidak dihitung secara terpisah. Misalnya, input data pribadi seorang karyawan mungkin dianggap sebagai satu input.
- *Outputnya* adalah tampilan data aplikasi. Ini bisa berupa laporan, tampilan layar, atau pesan kesalahan. Sekali lagi, masing-masing bidang biasanya tidak dianggap sebagai keluaran yang terpisah. Jika laporan memiliki beberapa baris, misalnya satu baris untuk setiap karyawan di departemen, semua baris ini akan dihitung sebagai satu keluaran. Namun, beberapa otoritas akan menghitung garis ringkasan sebagai keluaran terpisah.
- *File internal* adalah file logis yang dipahami pelanggan harus dipelihara oleh sistem. Jika sebuah file sebenarnya berisi 1000 entri data personel, mungkin akan dihitung sebagai satu file. Namun, jika file tersebut berisi data personalia, data ringkasan departemen, dan data departemen lainnya, maka mungkin akan dihitung sebagai tiga file terpisah untuk keperluan penghitungan titik fungsi.
- *Antarmuka eksternal* adalah data yang dibagikan dengan program lain. Misalnya, file personalia mungkin digunakan oleh sumber daya manusia untuk promosi dan penggajian. Dengan demikian, ini akan dianggap sebagai antarmuka di kedua sistem.

Menghitung Titik Fungsi yang Tidak Disesuaikan

Item titik fungsi individual diidentifikasi dan kemudian diklasifikasikan menjadi sederhana, rata-rata, atau kompleks. Bobot dari Tabel 4-8 kemudian diberikan pada tiap item dan totalnya dijumlahkan. Jumlah ini disebut titik fungsi yang belum disesuaikan.

Tidak ada standar untuk menghitung titik fungsi. Buku telah ditulis dengan aturan penghitungan yang berbeda. Hal penting untuk diingat adalah titik fungsi mencoba mengukur jumlah upaya yang diperlukan untuk mengembangkan perangkat lunak. Jadi, hal-hal yang berhubungan dengan usaha besar perlu menghasilkan lebih banyak titik fungsi dibandingkan hal-hal yang membutuhkan usaha kecil. Misalnya, salah satu perbedaan antara pendekatan penghitungan titik fungsi terkait dengan baris ringkasan di bagian bawah laporan. Beberapa insinyur perangkat lunak merasa bahwa baris ringkasan berarti keluaran lain harus dihitung, sementara yang lain hanya menghitung item utama dalam laporan. Jawabannya harus didasarkan pada seberapa besar upaya tambahan yang dibutuhkan garis ringkasan tersebut.

Aturan khusus tidak sepenting konsistensi dalam organisasi. Bekerja sama dan meninjau analisis titik fungsi lainnya akan membantu membangun konsistensi tersebut. Selain itu, peninjauan estimasi setelah penyelesaian proyek dapat membantu menentukan item mana yang memerlukan upaya lebih besar daripada yang ditunjukkan oleh analisis titik fungsi dan mungkin item mana yang dihitung terlalu tinggi dalam hal titik fungsi dan tidak memerlukan upaya sebanyak yang ditunjukkan.

Catatan: Cobalah untuk membuat titik fungsi Anda konsisten dengan upaya yang diperlukan untuk memproses setiap item.

Tabel 4-8 Bobot Titik Fungsi

	Simple	***Rata-Rata***	***Komplek***
Output	4	5	7
Inquiries	3	4	6
Input	3	4	6
File	7	10	15
Antarmuka	5	7	10

Contoh 4.12

Departemen menginginkan program yang menetapkan waktu dan ruangan untuk setiap bagian dan membuat jadwal lini untuk kursus. Departemen ini memiliki daftar bagian dengan nama profesor yang ditugaskan dan ukuran yang diharapkan. Departemen juga memiliki daftar ruangan dengan jumlah maksimum siswa yang dapat ditampung setiap ruangan. Ada juga rangkaian kelas yang tidak dapat diajarkan pada waktu yang bersamaan. Selain itu, profesor tidak dapat mengajar dua mata kuliah sekaligus.

Program ini jauh lebih sulit dibandingkan kompleksitas input dan outputnya. Ini memiliki dua input utama: file dengan daftar bagian, profesor yang ditugaskan, dan ukuran yang diantisipasi, dan file dengan daftar ruangan dengan ukuran maksimum.

Kedua file ini, meskipun mudah dibaca, akan sulit untuk diproses, sehingga dinilai rumit. Akan ada file tambahan dengan kumpulan kelas yang tidak dapat diajarkan secara bersamaan. Sekali lagi, struktur file ini sederhana tetapi akan sulit untuk diproses. Baris terakhir memiliki batasan yang bukan merupakan input atau output.

Ada outputnya, jadwal jalur. Ini adalah keluaran yang kompleks. Tidak ada pertanyaan atau antarmuka yang disebutkan, atau disebutkan tentang file yang dipelihara.

Produktivitas

Salah satu ukuran penting adalah produktivitas para pengembang perangkat lunak. Hal ini ditentukan dengan membagi ukuran total produk jadi dengan total usaha semua pemrogram. Ini memiliki satuan LOC/hari programmer. Alternatifnya adalah mengukur produktivitas dalam bentuk titik fungsi per hari programmer. Perhatikan bahwa produktivitas mencakup semua upaya yang dikeluarkan dalam semua fase siklus hidup perangkat lunak.

Contoh 4.13

Perusahaan XYZ menghabiskan upaya berikut untuk setiap fase siklus hidup proyek terbaru (lihat Tabel 4-9). Hitung upaya dalam LOC/hari pemrogram dan dalam titik fungsi/hari pemrogram. Perkiraan titik fungsi adalah 50 titik fungsi yang tidak disesuaikan. Proyek yang telah selesai mencakup 950 baris kode.

Tabel 4.9 Upaya Selama Tahapan

Fase	Pemrogramer (Hari)
Persiapan	20
Design	10
Implementasi	10
Pengujian	15
Dokumentasi	10

Total upaya adalah 65 hari programmer. Ini memberikan produktivitas 950/65 = 14,6 baris kode/hari programmer. Menggunakan titik fungsi yang tidak disesuaikan (fp), produktivitasnya adalah 50 fp/65 hari = 0,77 fp/programmer-hari.

Evaluasi Estimasi

Untuk mengevaluasi estimasi, suatu ukuran perlu dihitung. Tom DeMarco mengusulkan *faktor kualitas estimasi* (EQF). DeMarco mendefinisikan EQF sebagai luas di bawah kurva aktual dibagi luas antara estimasi dan nilai aktual. Ini adalah kebalikan dari persentase kesalahan atau kesalahan relatif rata-rata. Jadi, semakin tinggi EQF, semakin baik rangkaian estimasinya. DeMarco mengatakan nilai di atas 8 adalah wajar.

Contoh 4.11

Perkiraan berikut diberikan untuk proyek yang menelan biaya Rp. 3.5 Miliyar ketika selesai setelah 11,5 bulan:

Mulai	1.5 Bulan	5.5 Bulan	8 Bulan
Rp. 2.3 Miliyar	Rp. 3.1 Miliyar	Rp. 3.9 Miliyar	Rp. 3.5 Miliyar

Luas totalnya adalah 11,5 bulan dikalikan Rp. 3.5 Miliyar = Rp. 40.25 Miliyar bulan. Selisih antara kurva aktual dan perkiraan adalah |2.3 — 3:5| * 1:5 + |3.1 — 3.5| * 4 + |3.9 — 3.5| * 2.5 + |3.4 — 3.5| * 3.5 = Rp. 4.75 Miliyar bulan. Rasionya adalah 40.25/4.75 = 8.7.

Alat Estimasi Otomatis

Banyak alat tersedia di Internet yang akan menghitung COCOMO atau COCOMO2. Sebagian besar memiliki antarmuka yang sangat sederhana. Telusuri COCOMO menggunakan browser apa pun, dan ia akan menemukan banyak situs.

LATIHAN SOAL

1. Apa perbedaan antara WBS dan model proses?
2. Mengapa WBS harus berupa pohon?
3. Apa jadinya jika tidak ada kriteria penyelesaian suatu tugas di WBS?
4. Apa keuntungan menggunakan diagram PERT?
5. Mengapa menunda tugas di jalur kritis menunda keseluruhan proyek?
6. Apakah jalur kritis penting jika hanya satu orang yang mengerjakan suatu proyek?
7. Apa pentingnya waktu senggang?
8. Mengapa slack time didasarkan pada waktu paling awal dari waktu mulai terakhir tugas-tugas berikutnya?

9. Gambarlah diagram yang menunjukkan skala ekonomi dan skala disekonomis. Beri label pada diagram dan jelaskan yang mana.
10. Sangat umum menggunakan versi estimasi default. Pertimbangkan kapan terakhir kali seseorang meminta Anda memberikan perkiraan tentang sesuatu. Apakah estimasi yang Anda berikan merupakan definisi default estimasi atau definisi estimasi yang diusulkan DeMarco?
11. Mengapa parameter estimasi biaya harus ditentukan dari data perusahaan?

Latihan Tambahan (Diskusi)

1. Membuat WBS untuk tugas mengecat ruangan. Asumsikan bahwa model prosesnya adalah untuk proyek pekerjaan rumahan dengan aktivitas: merencanakan pekerjaan, membeli perlengkapan, melakukan pekerjaan, membersihkan.
2. Membuat WBS untuk pengembangan perangkat lunak untuk kantor gigi berikut:

   Tom memulai praktik dokter gigi di kota kecil. Dia akan memiliki asisten gigi, ahli kesehatan gigi, dan resepsionis. Dia menginginkan sistem untuk mengatur janji temu.
   Ketika seorang pasien meminta janji temu, resepsionis akan memeriksa kalender dan akan mencoba menjadwalkan pasien sedini mungkin untuk mengisi lowongan. Jika pasien puas dengan janji temu yang diusulkan, resepsionis akan memasukkan nama pasien dan tujuan janji temu. Sistem akan memverifikasi nama pasien dan memberikan rincian yang diperlukan dari catatan pasien, termasuk nomor ID pasien. Setelah setiap pemeriksaan atau pembersihan, ahli kesehatan atau asisten akan menandai janji temu telah selesai, menambahkan komentar, dan kemudian menjadwalkan pasien untuk kunjungan berikutnya jika diperlukan. Sistem akan menjawab pertanyaan berdasarkan nama pasien dan tanggal. Rincian pendukung dari catatan pasien ditampilkan bersama dengan informasi janji temu. Resepsionis dapat membatalkan janji temu. Resepsionis dapat mencetak daftar pemberitahuan untuk melakukan panggilan pengingat 2 hari sebelum janji temu. Sistem menyertakan nomor telepon pasien dari catatan pasien. Resepsionis juga dapat mencetak jadwal kerja harian dan mingguan dengan seluruh pasien.

3. Membuat WBS untuk pengembangan perangkat lunak untuk masalah B&B berikut:

   Tom dan Sue memulai penginapan dan sarapan di kota kecil di New England. Mereka akan memiliki tiga kamar tidur untuk para tamu. Mereka menginginkan sebuah sistem untuk mengelola pemesanan dan memantau pengeluaran dan keuntungan. Ketika calon pelanggan menelepon untuk melakukan reservasi, mereka akan memeriksa kalender, dan jika ada lowongan, mereka akan memasukkan nama pelanggan, alamat, nomor telepon, tanggal, harga yang disepakati, nomor kartu kredit, dan nomor kamar. . Reservasi harus dijamin dengan pembayaran 1 hari.
   Reservasi akan dilakukan tanpa jaminan untuk waktu yang telah disepakati. Jika tidak ada jaminan pada tanggal tersebut, reservasi akan dibatalkan.

4. Membuat WBS untuk pengembangan perangkat lunak untuk masalah dealer mobil berikut:

   Dealer mobil ingin mengotomatiskan inventarisnya. Itu dapat mencatat semua mobil yang dibeli pelanggan. Ini mencatat semua perbaikan. Ini mencatat semua pengiriman suku cadang perbaikan yang tiba. Dealer menginginkan laporan harian mengenai total perbaikan harian, penjualan harian, dan total inventaris. Laporan ini disebut ''laporan harian''. Dealer juga melacak semua pelanggan dan calon pelanggan yang mengunjungi

dealer. Dealer juga menginginkan laporan bulanan yang menunjukkan semua kunjungan dan pembelian pelanggan berdasarkan hari dalam sebulan. Dealer juga menginginkan kemampuan untuk menanyakan tentang pelanggan atau calon pelanggan mana pun.

5. Gambarlah diagram PERT untuk tugas Soal 1.
6. Gambarkan diagram PERT untuk kumpulan tugas dan ketergantungan tertentu. Lengkapi tabel yang menunjukkan jalur kritis dan waktu slack.

Simpul	Bagian	Waktu	Mulai	Selesai
a		10		
b	A	5		
c	A	2		
d	A	3		
e	b,c	7		
f	b,d	9		
g	c,d	5		
h	e,f,g	6		

7. Gambarkan diagram PERT untuk kumpulan tugas dan ketergantungan tertentu. Lengkapi tabel yang menunjukkan jalur kritis dan waktu slack.

Simpul	Dependensi	Waktu	Mulai	Selesai
a		10		
b	e	10		
c	d,f	10		
d	a,f,b	20		
e	a,f	8		
f	a	5		

8. Perkirakan parameter biaya dari kumpulan data yang diberikan:

Proyek	Ukuran (KLOC)	Biaya Programer (Bulan)
a	30	84
b	5	14
c	20	56
d	50	140
e	100	280
f	10	28

9. Perkirakan parameter biaya dari kumpulan data yang diberikan:

Proyek	Ukuran (KLOC)	Biaya Programer (Bulan)
a	30	95
b	5	80
c	20	65
d	50	155
e	100	305
f	10	35

10. Hitung upaya COCOMO, TDEV, staf rata-rata, dan produktivitas untuk proyek organik yang diperkirakan berjumlah 39.800 baris kode.
11. Hitung titik-titik fungsi yang belum disesuaikan untuk uraian soal pada Soal 2.

BAB 5
METRIK PERANGKAT LUNAK

5.1 PENDAHULUAN

Sains didasarkan pada pengukuran. Memperbaiki suatu proses memerlukan pemahaman tentang hubungan numerik. Ini memerlukan pengukuran. Pengukuran perangkat lunak adalah pemetaan simbol ke objek. Tujuannya adalah untuk mengukur beberapa atribut objek, misalnya untuk mengukur ukuran proyek perangkat lunak. Selain itu, tujuannya mungkin untuk memprediksi beberapa atribut lain yang saat ini tidak dapat diukur, seperti upaya yang diperlukan untuk mengembangkan proyek perangkat lunak.

Tidak semua pemetaan simbol ke objek berguna. Kekhawatiran penting adalah validasi metrik. Namun, validasi terkait dengan penggunaan metrik. Contohnya adalah tinggi badan seseorang. Tinggi badan berguna untuk memprediksi kemampuan seseorang melewati ambang pintu tanpa membenturkan kepalanya. Memiliki korelasi yang tinggi antara suatu ukuran dan suatu atribut saja tidak cukup untuk memvalidasi suatu ukuran. Misalnya, ukuran sepatu seseorang sangat berkorelasi dengan tinggi badan orang tersebut. Namun, ukuran sepatu biasanya tidak dapat diterima sebagai ukuran tinggi badan seseorang.

Berikut ini adalah kriteria untuk metrik yang valid:

1. Metrik harus memungkinkan entitas yang berbeda dapat dibedakan.
2. Metrik harus mematuhi kondisi representasi.
3. Setiap unit atribut harus memberikan kontribusi jumlah yang setara terhadap metrik.
4. Entitas yang berbeda dapat mempunyai nilai atribut yang sama.

Sering kali, atribut yang diminati tidak dapat diukur secara langsung. Dalam hal ini, ukuran tidak langsung digunakan. Ukuran tidak langsung melibatkan ukuran dan rumus prediksi. Misalnya, kepadatan bukanlah ukuran langsung. Ini dihitung dari massa dan kepadatan, yang keduanya merupakan ukuran langsung. Dalam ilmu komputer, banyak “kemampuan” (pemeliharaan, keterbacaan, pengujian, kualitas, kompleksitas, dll.) tidak dapat diukur secara langsung, dan pengukuran tidak langsung untuk atribut-atribut ini adalah tujuan dari banyak program metrik.

Berikut ini adalah kriteria metrik tidak langsung yang valid:

1. Model harus didefinisikan secara eksplisit.
2. Model harus konsisten secara dimensi.
3. Tidak boleh ada diskontinuitas yang tidak terduga.
4. Satuan dan jenis skala harus benar.

5.2 TEORI PENGUKURAN PERANGKAT LUNAK

Teori pengukuran representasional telah dipelajari selama lebih dari 100 tahun. Ini melibatkan sistem hubungan empiris, sistem hubungan numerik, dan pemetaan pelestarian hubungan antara kedua sistem.

Sistem hubungan empiris (E, R) terdiri dari dua bagian:

- Satu set entitas, E
- Satu set hubungan, R

Hubungannya biasanya ''kurang dari atau sama.'' Perhatikan bahwa tidak semuanya harus berhubungan. Artinya, himpunan R mungkin merupakan orde parsial. Sistem relasi numerik (N, P) juga terdiri dari dua bagian:

- Himpunan entitas, N. Disebut juga ''himpunan jawaban'', himpunan ini biasanya berupa bilangan— bilangan asli, bilangan bulat, atau real.
- Sekumpulan relasi, P. Himpunan ini biasanya sudah ada dan seringkali “kurang dari” atau “kurang dari atau sama dengan”.

Pemetaan pelestarian hubungan, M, memetakan (E, R) ke (N, P). Pembatasan penting pada pemetaan ini disebut kondisi representasi. Ada dua kemungkinan kondisi representasi. Versi yang paling ketat mengatakan bahwa jika dua entitas terkait di salah satu sistem, maka gambar (atau pra-gambar) di sistem lain akan terkait:

$$x \text{ rel } y \text{ jika } \mathrm{M}(x) rel \ \mathrm{M}(y)^{3}$$

Versi yang tidak terlalu membatasi mengatakan bahwa jika dua entitas terkait dalam sistem empiris, maka gambaran kedua entitas dalam sistem numerik tersebut terkait dengan cara yang sama:

$$\mathrm{M}(x) \text{ rel } \mathrm{M}(y) \text{ Jika } X \text{ rel } y$$

Penulis teori pengukuran klasik telah menggunakan kedua versi tersebut. Keuntungan dari versi kedua adalah bahwa tatanan parsial dalam sistem empiris dapat dipetakan ke bilangan bulat atau real yang keduanya terurut total.

Contoh 5.1 Tinggi Orang

Contoh klasik pemetaan sistem empiris ke sistem numerik adalah tinggi badan manusia. Dalam sistem empiris, ada hubungan tinggi badan yang dipahami dengan baik di antara manusia. Mengingat dua orang yang berdiri bersebelahan, semua orang pasti sepakat siapa yang lebih tinggi. Inilah sistem empirisnya: manusia adalah entitas dan hubungan yang dipahami dengan baik adalah “lebih pendek atau sama tingginya”.

Sistem numerik adalah sistem bilangan real (satuan metrik atau imperial) dengan hubungan standar kurang dari atau sama.

Pemetaannya hanyalah standar pengukuran tinggi badan orang. Ini biasanya diukur tanpa alas kaki, berdiri tegak di dinding.

Kondisi representasi (versi mana pun) terpenuhi, karena jika Fred lebih pendek atau sama dengan Bill, maka tinggi badan Fred yang diukur kurang dari atau sama dengan tinggi badan terukur Bill.

Contoh 5.2

Kembangkan ukuran, BESAR, untuk orang yang menggabungkan berat dan tinggi badan.

Secara empiris, jika dua orang memiliki tinggi badan yang sama, maka semakin berat maka semakin besar, dan jika dua orang memiliki berat yang sama, semakin tinggi maka semakin besar. Jika kita menggunakan gagasan ini, kita dapat memperoleh sebagian tatanan yang disetujui oleh kebanyakan orang. Satu-satunya pasangan orang yang tidak akan kami pesan dengan cara ini adalah jika yang satu lebih berat dan yang lainnya lebih tinggi.

Secara numerik, kita bisa menggunakan tupel, <tinggi, berat>. Setiap bagian dari tupel akan menjadi bilangan real. Dua tupel akan direlasikan jika kedua bagian direlasikan dalam arah yang sama. Artinya, jika x ; y adalah tupel, maka x lebih kecil atau sama dengan y dalam ''kebesaran'' jika $x_{tinggi} \leq y_{tinggi}$ dan $x_{berat} \leq y_{berat}$. Ini juga merupakan pesanan parsial, dan kedua versi kondisi representasi terpenuhi.

MONOTONISITAS

Karakteristik penting dari suatu ukuran adalah monotonisitas. Artinya nilai ukuran suatu atribut tidak berubah arah seiring bertambahnya atribut pada objek tersebut. Misalnya, jumlah baris kode tidak akan berkurang seiring bertambahnya kode.

Contoh5.3

Fungsi linier bersifat monotonik, karena selalu berjalan pada arah yang sama. Fungsi kuadrat biasanya tidak monotonik. Misalnya, $y = 5x - x^2$ tidak monotonik dalam rentang $x = 0$ hingga $x = 10$. Dari $x = 0$ hingga $x = 5, y$ bertambah. Dari $x = 5$ ke $x = 10, y$ berkurang.

TIMBANGAN PENGUKURAN

Ada lima jenis skala yang berbeda: nominal, ordinal, interval, rasio, dan absolut. Pengukuran yang paling tidak ketat adalah menggunakan jenis skala nominal. Tipe ini pada dasarnya memberikan angka atau simbol tanpa memperhatikan kuantitas apapun. Contoh klasik ukuran skala nominal adalah angka pada seragam olahraga. Kami tidak berpikir bahwa satu pemain lebih baik dari yang lain hanya karena nomor pada seragam yang satu lebih besar atau lebih kecil dari nomor pada seragam lainnya. Tidak ada rumus untuk mengkonversi dari satu skala nominal ke skala nominal lainnya.

Dalam ukuran skala ordinal, terdapat pengurutan tersirat dari entitas berdasarkan nomor yang diberikan kepada entitas tersebut. Contoh klasiknya adalah peringkat kelas. Jika seorang siswa mendapat peringkat pertama, prestasinya lebih baik daripada siswa yang menduduki peringkat kedua, atau ketiga, atau angka lain yang lebih besar dari 1. Namun, kami tidak pernah berasumsi bahwa perbedaan angka dalam peringkat itu signifikan. Artinya, kita tidak berasumsi bahwa selisih antara siswa pertama dan kedua sama dengan selisih antara siswa ke-100 dan ke-101. Rumus apa pun yang mengkonversi dari satu ukuran skala ordinal ke ukuran skala ordinal lainnya untuk entitas yang sama harus mempertahankan urutannya.

Dalam ukuran skala interval, besarnya selisihnya adalah konstan. Contohnya adalah suhu. Ada dua contoh pengukuran suhu yang umum digunakan, Fahrenheit dan Celsius. Rumus untuk mengubah skala Celsius ke skala Fahrenheit adalah $9/5 * x + 32$. Dengan dua skala interval untuk atribut yang sama, rumus konversinya harus berbentuk $ax + b$.

Dalam ukuran skala rasio, jumlah perbedaannya adalah konstan dan terdapat angka nol yang dapat digunakan oleh ukuran skala apa pun. Misalnya uang, panjang, dan tinggi

badan diukur dengan menggunakan skala rasio. Pengukuran ini memiliki pengertian nol yang dipahami dengan baik: nol uang, nol tinggi, dan nol panjang. Rumus apa pun untuk mengonversi satuan ke satuan lainnya—dari sentimeter ke inci, misalnya—hanya akan menggunakan konstanta perkalian.

Yang mutlak adalah ukuran skala penghitungan. Unit-unitnya jelas dan dipahami dengan baik. Menghitung kelereng merupakan salah satu contoh pengukuran skala absolut.

STATISTIK

Tidak semua statistik cocok untuk semua skala. Berikut ini menunjukkan metode statistik umum mana yang sesuai:

Skala nominal: Hanya modus, median, dan persentil

Skala ordinal: Korelasi di atas dan Spearman

Skala interval: Di atas dan mean, deviasi standar, dan korelasi Pearson

Skala rasio: Semua statistik

Skala absolut: Semua statistik

Contoh 5.4 Rata-rata

Suhu adalah ukuran skala interval. Oleh karena itu, masuk akal secara statistik untuk memberikan suhu rata-rata. Namun, angka pada seragam pemain baseball hanyalah ukuran skala nominal. Tidak masuk akal untuk memberikan rata-rata angka pada seragam tim. Begitu pula dengan rata-rata rangking siswa di suatu kelas atau rata-rata rangking siswa di sejumlah kelas juga tidak tepat.

5.3 METRIK PRODUK

Metrik produk adalah metrik yang dapat dihitung dari dokumen, terlepas dari cara pembuatannya. Umumnya, ini berkaitan dengan struktur kode sumber. Metrik produk dapat ditentukan untuk dokumen lain. Misalnya, jumlah paragraf dalam spesifikasi persyaratan akan menjadi metrik produk.

Contoh 5.5 Garis Kode

Metrik paling dasar untuk ukuran adalah baris kode metrik. Ada banyak cara berbeda untuk menghitung baris kode. Definisinya mungkin sederhana seperti jumlah karakter LINE BARU dalam file. Seringkali komentar dikecualikan dari hitungan baris. Terkadang garis kosong atau garis yang hanya memiliki pembatas dikecualikan. Terkadang pernyataan dihitung, bukan garis.

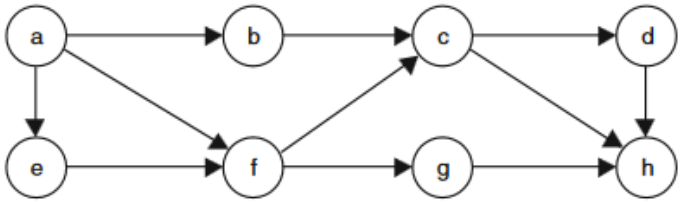
Nomor Siklomatik McCabe

Angka siklomatik McCabe, yang diperkenalkan pada tahun 1976, merupakan salah satu metrik yang paling umum digunakan dalam pengembangan perangkat lunak, setelah baris kode. Juga disebut ''ukuran kompleksitas McCabe'' dari judul artikel jurnal aslinya, hal ini didasarkan pada bilangan siklomatik teori graf. McCabe mencoba mengukur kompleksitas suatu program. Premisnya adalah bahwa kompleksitas berhubungan dengan aliran kendali program. Teori graf menggunakan rumus $C = e - n + 1$ untuk menghitung bilangan siklomatik. McCabe menggunakan rumus yang sedikit dimodifikasi:

$$C = e - n + 2p$$

Di mana:
e = Jumlah tepi
n = Jumlah node
p = Jumlah komponen yang terhubung kuat (biasanya 1)

Contoh 5.6
Tentukan bilangan siklomatik dari grafik aliran kendali yang ditunjukkan pada Gambar 5-1.



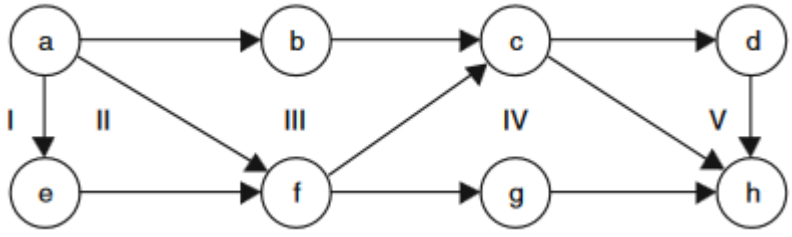
Gambar 5-1. Grafik aliran kontrol.

Ada 8 node, jadi $n = 8$. Ada 11 busur, jadi $e = 11$. Bilangan siklomatiknya adalah $C = 11 - 8 + 2 = 5$:

Graf planar adalah graf yang dapat digambar tanpa ada perpotongan garis. Matematikawan Swiss Leonhard Euler (1707–1783) membuktikan untuk graf planar bahwa $2 = n - e + r$, dengan r jumlah daerah, e = jumlah sisi, dan n = jumlah simpul. Daerah adalah daerah yang dikelilingi (atau dibatasi) oleh busur. Dengan menggunakan aljabar, bilangan ini dapat diubah menjadi $r = e - n + 2$. Oleh karena itu, jumlah daerah pada grafik planar sama dengan bilangan siklomatik.

Contoh 5.7

Beri label daerah pada grafik aliran kendali dari Contoh 5.6 dengan angka Romawi. Seperti ditunjukkan pada Gambar 5-2, ada lima wilayah. Wilayah I adalah bagian luar grafik.



Gambar 5-2. Kontrol grafik aliran dengan angka romawi.

Menghitung bilangan siklomatik dari grafik aliran kendali memakan waktu. Membangun grafik aliran kendali dari program besar akan sangat memakan waktu. McCabe menemukan metode yang lebih langsung dalam menghitung ukurannya. Ia menemukan bahwa jumlah daerah

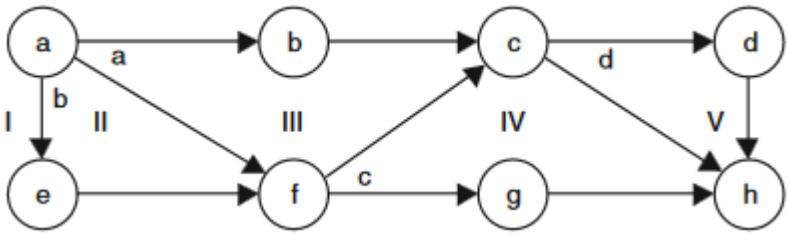
biasanya sama dengan satu lebih banyak dari jumlah keputusan dalam suatu program, $C = \pi + 1$, dimana π adalah jumlah keputusan.

Dalam kode sumber, pernyataan `IF`, perulangan `WHILE`, atau perulangan `FOR` dianggap sebagai satu keputusan. Pernyataan `CASE` atau beberapa cabang lainnya dihitung sebagai satu keputusan yang lebih sedikit dibandingkan dengan jumlah cabang yang mungkin.

Grafik aliran kontrol harus memiliki titik awal yang berbeda dan titik akhir yang berbeda. Jika hal ini dilanggar, maka jumlah keputusannya tidak akan berkurang satu pun dari jumlah daerah.

Contoh 5.8

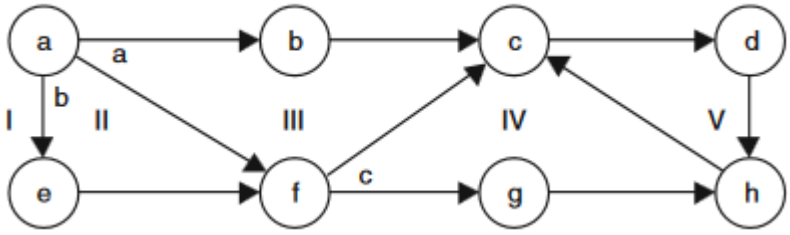
Beri label keputusan pada grafik alur kendali Contoh 5.6 dengan huruf kecil. Seperti ditunjukkan pada Gambar 5-3, dari simpul a terdapat tiga busur, sehingga harus ada dua keputusan yang diberi label a dan b. Dari node c dan f, ada dua busur dan masing-masing satu keputusan. Node lainnya mempunyai paling banyak satu pintu keluar sehingga tidak ada keputusan. Ada empat keputusan, jadi $C = 4 + 1 = 5$.



Gambar 5-3. Kontrol grafik aliran dengan huruf kecil.

Contoh 5.9

Hitung bilangan siklomatik menggunakan grafik aliran kendali tidak valid yang ditunjukkan pada Gambar 5-4.



Gambar 5-4. Grafik aliran kontrol tidak valid.

Cfg sama dengan contoh sebelumnya, hanya saja busur c-h telah digantikan oleh busur h-c. Ini tidak akan mengubah jumlah node, edge, atau wilayah. Jadi, dua metode penghitungan bilangan siklomatik pertama tidak akan berubah. Namun keputusan d telah dihilangkan sehingga cara ketiga tidak akan memberikan jawaban yang sama. Namun, ini bukan cfg yang valid, karena sekarang tidak ada node yang berhenti.

Nilai Ambang Batas

Aspek penting dari metrik adalah panduan tentang kapan nilai tersebut masuk akal dan kapan nilai tersebut tidak masuk akal. McCabe menganalisis sebuah proyek besar dan

menemukan bahwa untuk modul dengan bilangan siklomatik lebih dari 10, modul tersebut memiliki riwayat lebih banyak kesalahan dan lebih banyak kesulitan dalam pemeliharaan. Dengan demikian, 10 telah diterima sebagai nilai ambang batas bilangan siklomatik dalam suatu modul. Apabila bilangan siklomatis lebih besar dari 10, maka perlu dilakukan upaya untuk menurunkan nilainya atau membagi modulnya.

Ilmu Perangkat Lunak Halstead

Maurice Halstead adalah salah satu peneliti pertama di bidang metrik perangkat lunak. Dia melakukan pekerjaannya pada akhir tahun 1960an dan 1970an. Tujuannya adalah untuk mengidentifikasi apa yang berkontribusi terhadap kompleksitas perangkat lunak. Dia secara empiris mencari ukuran ukuran intrinsik. Setelah menemukan apa yang menurutnya merupakan ukuran dan rumus prediksi yang baik, ia mencoba mengembangkan teori yang koheren. Metrik sederhananya masih dianggap valid, sedangkan metrik dan rumus prediksinya yang lebih kompleks dianggap mencurigakan.

Entitas Dasar—Operator dan Operan

Pendekatan dasar yang memberikan hasil yang baik bagi Halstead adalah dengan mempertimbangkan program apa pun sebagai kumpulan token, yang ia klasifikasikan sebagai operator atau operan. Operan adalah token yang memiliki nilai. Biasanya, variabel dan konstanta adalah operan. Segala sesuatu yang lain dianggap sebagai operator. Jadi, koma, tanda kurung, operator aritmatika, tanda kurung, dan sebagainya semuanya dianggap sebagai operator.

Semua token yang selalu muncul berpasangan, tripel, dan seterusnya akan dihitung bersama sebagai satu token. Misalnya, tanda kurung kiri dan tanda kurung kanan akan dianggap sebagai salah satu kemunculan tanda kurung. Bahasa yang memiliki konstruksi if-then akan dianggap memiliki token if-then.

Halstead juga memperhatikan algoritma dan bukan tentang deklarasi, pernyataan i/o, dan sebagainya. Oleh karena itu, dia tidak menghitung deklarasi, pernyataan masukan atau keluaran, atau komentar. Namun, saat ini sebagian besar organisasi akan menghitung seluruh bagian dari suatu program.

Definisi Halstead tentang operator dan operan terbuka untuk banyak interpretasi. Tidak ada standar yang diterima untuk memutuskan situasi yang ambigu. Kabar baiknya adalah selama sebuah organisasi konsisten, hal itu tidak menjadi masalah. Kabar buruknya adalah orang-orang di satu organisasi tidak dapat membandingkan hasil mereka dengan hasil di organisasi lain.

Penulis merekomendasikan pendekatan berbasis sintaksis dimana semua operan adalah token yang ditentukan pengguna dan semua operator adalah token yang ditentukan oleh sintaks bahasa.

Tindakan Dasar—η_1 dan η_2

Jumlah operator unik dalam suatu program adalah η_1 (diucapkan ''eta one''), dan jumlah operan unik dalam suatu program adalah η_2 (diucapkan ''eta two'').

Jumlah total token unik adalah $\eta = \eta_1 + \eta_2$. Ini adalah ukuran dasar dari ukuran program.

Contoh 5.10

Identifikasi operator dan operan unik dalam kode berikut yang melakukan perkalian dengan penjumlahan berulang.

```
Z=0;
while X > 0
    Z= Z+Y;
    X= X-1;
end-while ;
print(Z) ;
```

operators

```
=; while-endwhile >+-print ()
```

operands

```
Z O X Y 1
```

thus, η_1= 8 and η_2 = 5

Operan Potensial, η_2*

Halstead ingin mempertimbangkan dan membandingkan implementasi algoritma yang berbeda. Dia mengembangkan konsep operan potensial yang mewakili kumpulan nilai minimal yang diperlukan untuk setiap implementasi algoritma yang diberikan. Hal ini biasanya dihitung dengan menghitung semua nilai yang awalnya tidak ditetapkan dalam algoritma. Ini akan mencakup nilai yang dibaca, parameter yang diteruskan, dan nilai global yang diakses dalam algoritme.

Panjang, N

Ukuran dasar berikutnya adalah jumlah total operator, N_1, dan jumlah total operan, N_2. Ini dijumlahkan untuk mendapatkan panjang program dalam bentuk token:

$$\mathbf{N = N_1 + N_2}$$

Contoh 5.11

Hitung panjang Halstead untuk kode Contoh 5.10.

Operator

```
=               3
;               5
while-endwhile  1
>               1
+               1
-               1
Print           1
```

()	1

Operand

z	4
0	2
x	3
y	2
1	1

Ada 14 kemunculan operator, jadi N_1 adalah 14. Demikian pula N_2 adalah 12.
$\mathrm{N} = \mathrm{N}_1 + \mathrm{N}_2 = 14 + 12 = 26.$

Perkiraan Panjang (est N atau N_hat)

Perkiraan panjang adalah rumus prediksi Halstead yang paling dasar. Berdasarkan perkiraan jumlah operator dan operan yang akan digunakan dalam suatu program, rumus ini memungkinkan perkiraan ukuran sebenarnya dari program dalam bentuk token:

$$\mathrm{est\ N} = \eta_1 * \log_2 \eta_1 + \eta_2 * \log_2 \eta_2$$

Contoh 5.12

Hitung perkiraan panjang kode Contoh 5.10.
$\log_2$ dari x adalah eksponen yang harus dipangkatkan 2 untuk mendapatkan hasil yang sama X. Jadi, $\log_2$ dari 2 adalah 1, $\log_2$ dari 4 adalah 2, dari 8 adalah 3, dari 16 adalah 4:

$$\log_2 of\ \eta_1 = \log_2 8 = 3$$
$$\log_2 of\ \eta_2 = \log_2 5 = 2.32$$
$$\mathrm{est\ N} = 8 * 3 + 5 * 2.32 = 24 + 11.6 = 35.6$$

sedangkan N sebenarnya adalah 26. Ini akan dianggap sebagai batas. Ini mungkin bukan perkiraan yang buruk untuk program sekecil itu.

Dari pengalaman, saya menemukan bahwa jika N dan est N tidak berada dalam jarak sekitar 30 persen satu sama lain, mungkin tidak masuk akal untuk menerapkan langkah-langkah ilmu perangkat lunak lainnya.

Volume, V

Halstead menganggap volume sebagai ukuran 3D, padahal volume tersebut benar-benar terkait dengan jumlah bit yang diperlukan untuk menyandikan program yang diukur. Dengan kata lain:

$$\mathrm{V} = \mathrm{N} * \log_2(\eta_1 + \eta_2)$$

Contoh 5.13

Hitung V untuk kode Contoh 5.10.

$$\mathrm{V} = 26 * \log_2 13 = 26 * 3.7 = 96.2$$

Volume memberikan jumlah bit yang diperlukan untuk mengkodekan banyak nilai yang berbeda. Angka ini sulit untuk ditafsirkan.

Volume Potensial, V*

Volume potensial adalah ukuran minimal dari solusi suatu masalah, yang diselesaikan dalam bahasa apa pun. Halstead berasumsi bahwa dalam implementasi minimal, hanya akan ada dua operator: nama fungsi dan operator pengelompokan. Jumlah minimal operan adalah $\eta_2{}^*$:

$$V^* = (2 + \eta_2{}^*) \log_2(2 + \eta_2{}^*)$$

Tingkat Implementasi, L

Karena kita mempunyai volume aktual dan volume minimal, maka wajar jika kita mengambil perbandingan. Halstead membagi volume potensial dengan volume aktual. Hal ini berkaitan dengan seberapa dekat implementasi saat ini dengan implementasi minimal yang diukur dengan potensi volume. Tingkat implementasinya tidak memiliki unit.

$$\mathbf{L = V^*/V}$$

Langkah-langkah dasar yang dijelaskan sejauh ini masuk akal. Banyak ide operan dan operator telah digunakan dalam banyak upaya metrik lainnya. Langkah-langkah yang tersisa diberikan untuk kepentingan sejarah dan tidak direkomendasikan karena berguna atau valid.

Usaha, E

Halstead ingin memperkirakan berapa banyak waktu (usaha) yang diperlukan untuk mengimplementasikan algoritma ini. Dia menggunakan gagasan diskriminasi mental dasar (emd).

$$\mathbf{E = V/L}$$

Satuannya adalah diskriminasi mental dasar (emd). Upaya Halstead tidak monoton—dengan kata lain, ada program yang jika Anda menambahkan pernyataan, upaya yang diperhitungkan akan berkurang.

Waktu, T

Selanjutnya, Halstead ingin memperkirakan waktu yang diperlukan untuk mengimplementasikan algoritma tersebut. Ia menggunakan beberapa karya yang dikembangkan oleh psikolog pada tahun 1950-an, John Stroud. Stroud telah mengukur seberapa cepat subjek dapat melihat benda yang lewat dengan cepat di depan wajahnya. S adalah bilangan Stroud (emd/detik) yang diambil dari percobaan tersebut. Halstead menggunakan 18 emd/detik sebagai nilai S.

$$\mathbf{T = E/S}$$

Aliran Informasi Henry–Kafura

Sallie Henry dan Dennis Kafura mengembangkan metrik untuk mengukur kompleksitas antar modul kode sumber. Kompleksitasnya didasarkan pada aliran informasi masuk dan keluar dari modul. Untuk setiap modul, penghitungan dilakukan terhadap seluruh aliran informasi yang masuk ke dalam modul, ini, dan semua informasi yang mengalir keluar dari modul, keluar. Arus informasi ini mencakup penerusan parameter, variabel global, serta input dan output. Mereka juga menggunakan ukuran masing-masing modul sebagai faktor perkalian. LOC dan ukuran kompleksitas telah digunakan sebagai bobot ini.

$$\mathrm{HK_i} = \mathrm{weight_i} * (\mathrm{out_i} * \mathrm{in_i})^2$$

Besaran totalnya adalah penjumlahan $\mathrm{HK_i}$ dari masing-masing modul.

Contoh 5.14

Hitung metrik aliran informasi HK dari informasi berikut. Asumsikan bobot setiap modul adalah 1.

Mod #	a	b	c	d	e	f	g	h
$\mathrm{in_i}$	4	3	1	5	2	5	6	1
$\mathrm{out_i}$	3	3	4	3	4	4	2	6

Mod	a	b	c	d	e	f	g	h
$\mathrm{HK_i}$	144	81	16	255	64	400	144	36

HK untuk keseluruhan program adalah 1110.

5.4 METRIK PROSES

Produktifitas

Produktivitas adalah salah satu metrik proses dasar. Hal ini dihitung dengan membagi total jalur sumber yang dikirimkan dengan hari programmer yang dikaitkan dengan proyek tersebut. Unit biasanya LOC/hari programmer. Di banyak proyek pada tahun 1960an, produktivitasnya adalah 1 LOC/hari programmer. Dalam proyek besar, produktivitas biasanya berkisar antara 2 hingga 20 LOC/hari programmer. Dalam proyek-proyek kecil dan individual, produktivitasnya bisa jauh lebih tinggi.

Contoh 5.15

Proyek ini berjumlah 100 KLOC. Dua puluh programmer mengerjakan proyek tersebut selama satu tahun. Tahun ini mencakup seluruh upaya untuk tahap persyaratan, desain, implementasi, pengujian, dan pengiriman. Asumsikan ada sekitar 240 hari kerja dalam setahun (20 hari sebulan selama 12 bulan, tidak ada hari libur). Produktivitasnya 100.000 LOC / 20 * 240 hari = 20,8 LOC/hari programmer.

5.5 PENDEKATAN GQM

Vic Basili dan Dieter Rombach mengembangkan pendekatan ini di Universitas Maryland. GQM adalah singkatan dari tujuan, pertanyaan, dan metrik. Idenya adalah untuk terlebih dahulu mengidentifikasi tujuan dari pendekatan ini. Selanjutnya dikembangkan pertanyaan-pertanyaan yang berkaitan dengan tujuan tersebut. Terakhir, metrik dikembangkan untuk mengukur atribut yang terkait dengan pertanyaan.

Contoh 5.16

Gunakan pendekatan GQM untuk masalah kepuasan pelanggan.

***Sasaran*—**Kepuasan pelanggan

***Pertanyaan*—**Apakah pelanggan tidak puas ketika ditemukan masalah?

***Metrik*—**Jumlah laporan kerusakan pelanggan

LATIHAN SOAL

1. Jelaskan mengapa contoh tinggi badan memenuhi kriteria metrik yang valid.
2. Sebuah penelitian terhadap anak-anak sekolah dasar menemukan korelasi yang tinggi antara ukuran sepatu dan kemampuan membaca. Apakah ini berarti ukuran sepatu merupakan ukuran kecerdasan yang baik?
3. Jelaskan mengapa uang merupakan ukuran skala rasio dan bukan sekedar skala interval.
4. Jelaskan mengapa IPK tidak sesuai dengan teori pengukuran.
5. Mengapa kompleksitas tidak mudah diukur?
6. Apa keuntungan memiliki tatanan parsial pada sistem hubungan empiris?
7. Mengapa jumlah keputusan ditambah 1 merupakan metode penting untuk menghitung bilangan siklomatik McCabe?
8. Mengapa monotonisitas merupakan karakteristik penting dari metrik ukuran atau upaya seperti metrik upaya Halstead?

Latihan Soal Tambahan (Diskusi)

1. Identifikasi skala yang tepat untuk setiap ukuran berikut:
   LOC
   Bilangan siklomatik McCabe
   Rata-rata kedalaman sarang
   Kedalaman sarang maksimum
2. Tunjukkan bahwa skala suhu Celcius dan Fahrenheit merupakan skala interval yang menggunakan suhu Celcius 20, 30, dan 40 derajat.
3. Tunjukkan bahwa bilangan siklomatik McCabe memenuhi teori pengukuran representasional.
4. Tunjukkan bahwa bilangan siklomatik McCabe merupakan suatu ukuran skala interval.
5. Hitung bilangan siklomatik McCabe pada kode sumber berikut. Gambarlah grafik aliran kendali. Labeli daerah tersebut dengan angka Romawi.

```
read x,y,z;
type=`` scalene'';
if (x == y or x == z or y == z) type="isosceles'';
if (x == y and x == z) type =''equilateral'';
if (x >= y+z or y >= x+z or z >= x+y) type =''not a triangle''
if (x <= 0 or y <= 0 or| z <= 0) type =''bad inputs'';
print type;
```

BAB 6
ANALISIS DAN MANAJEMEN RISIKO

6.1 PENDAHULUAN

Risiko adalah kemungkinan terjadinya peristiwa yang tidak diinginkan (disebut peristiwa risiko). Risiko melibatkan ketidakpastian (peristiwa yang dijamin akan terjadi bukanlah risiko) dan kerugian (peristiwa yang tidak berdampak negatif terhadap proyek bukanlah risiko). Manajemen risiko proaktif adalah proses upaya meminimalkan kemungkinan dampak buruk dari peristiwa risiko yang terjadi.

Terdapat perbedaan pendapat mengenai risiko apa yang harus dikelola. Beberapa ahli menyarankan bahwa hanya risiko yang unik pada proyek saat ini yang harus dipertimbangkan dalam analisis dan manajemen risiko. Pandangan mereka adalah bahwa pengelolaan risiko yang umum terjadi pada sebagian besar proyek harus dimasukkan ke dalam proses perangkat lunak.

6.2 IDENTIFIKASI RISIKO

Ini adalah proses mengidentifikasi kemungkinan risiko. Risiko dapat diklasifikasikan menjadi ff yang memengaruhi rencana proyek (risiko proyek), yang memengaruhi kualitas (risiko teknis), atau yang memengaruhi kelayakan produk (risiko bisnis). Beberapa ahli mengecualikan peristiwa yang umum terjadi pada semua proyek dari pertimbangan manajemen risiko. Para ahli ini menganggap peristiwa umum tersebut sebagai bagian dari perencanaan proyek standar.

Contoh 6.1

Pertimbangkan sebuah proyek yang melibatkan upaya untuk mengembangkan perangkat lunak penting keselamatan pada perangkat keras mutakhir. Buat daftar risiko dan klasifikasikan masing-masing risiko sebagai proyek, teknis, atau bisnis dan sebagai risiko umum untuk semua proyek atau khusus untuk proyek ini.

Resiko	Proyek	Teknis	Bisnis	Umum	Spesial
Perangkat keras tidak tersedia		X			X
Persyaratan tidak lengkap	X			X	
Penggunaan metodologi khusus		X			
Masalah dalam mencapai keandalan yang dibutuhkan		X			
Retensi orang-orang penting	X			X	
Meremehkan upaya yang diperlukan	X			X	
Satu-satunya pelanggan potensial bangkrut			X		X

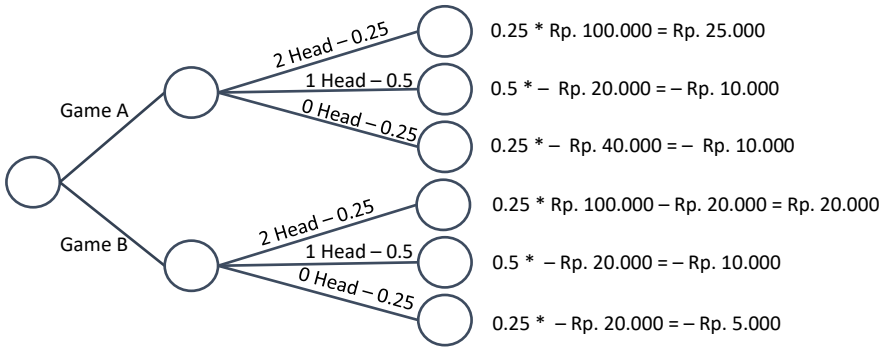
6.3 ESTIMASI RISIKO

Estimasi risiko melibatkan dua tugas dalam menilai suatu risiko. Tugas pertama adalah memperkirakan kemungkinan terjadinya suatu risiko, yang disebut probabilitas risiko, dan tugas kedua adalah memperkirakan biaya terjadinya peristiwa risiko, yang sering disebut

dampak risiko. Memperkirakan kemungkinan risiko akan sulit. Risiko yang diketahui jauh lebih mudah untuk dikelola, dan risiko tersebut menjadi bagian dari proses perangkat lunak. Risiko-risiko baru yang unik pada proyek saat ini adalah risiko-risiko yang paling penting untuk dikelola. Biaya risiko mungkin lebih mudah ditentukan berdasarkan pengalaman kegagalan proyek sebelumnya.

6.4 EKSPOSUR RISIKO

Eksposur risiko adalah nilai yang diharapkan dari peristiwa risiko. Hal ini dihitung dengan mengalikan probabilitas risiko dengan biaya kejadian risiko.



6.5 MITIGASI RISIKO

Mitigasi risiko adalah strategi proaktif dalam mencoba menemukan cara untuk mengurangi kemungkinan terjadinya peristiwa risiko atau dampak terjadinya peristiwa tersebut. Meskipun tidak ada cara ajaib untuk mengurangi risiko, pendekatan yang umum dilakukan adalah mencoba menyelesaikan risiko terkait ketidakpastian sejak dini. Misalnya, jika ada kekhawatiran mengenai sistem tertentu yang perlu digunakan, lebih baik menyelidiki sistem tersebut sejak dini. Seringkali, sebuah prototipe dapat dibangun yang akan mengidentifikasi masalah sejak dini.

Contoh 6.4

Pertimbangkan risiko yang diidentifikasi dalam masalah identifikasi risiko. Sarankan pendekatan untuk mengurangi kemungkinan atau mengurangi dampak atau keduanya.

Resiko	Kurangi Probabilitas	Kurangi Dampak
Perangkat keras tidak tersedia	Mempercepat pengembangan. perangkat keras	Bangun simulator
Persyaratan tidak lengkap	Tingkatkan peninjauan persyaratan	
Penggunaan metodologi khusus	Tingkatkan pelatihan staf, rekrut tenaga ahli	
Masalah dalam mencapai keandalan yang dibutuhkan	Desain untuk keandalan	
Retensi orang-orang penting	Bayar lebih	Pekerjakan personel tambahan
Meremehkan upaya yang diperlukan	Sewa estimator eksternal	Bangun di waktu senggang, sering-seringlah memperkirakan ulang
Satu-satunya pelanggan potensial bangkrut	Melakukan evaluasi eksternal terhadap area tersebut	Identifikasi klien potensial lainnya

6.6 RENCANA MANAJEMEN RISIKO

Rencana manajemen risiko harus mencakup pengidentifikasi, deskripsi risiko, perkiraan kemungkinan risiko, perkiraan dampak risiko, daftar strategi mitigasi, rencana darurat, pemicu risiko (untuk menentukan kapan rencana darurat harus diaktifkan) , dan individu yang bertanggung jawab. Bidang tambahan mungkin mencakup status metrik terkait saat ini dan/atau di masa lalu.

CONTOH 6.5

Kembangkan formulir untuk manajemen risiko dan masukkan data sampel.

ID Risiko: 1-010-77	**Masalah: 10 persen**	**Dampak: sangat tinggi**
Deskripsi: Perangkat keras khusus mungkin tidak tersedia.		
Strategi mitigasi: Membangun simulator, mempercepat pengembangan perangkat keras.		
Pemicu risiko: Perangkat keras terlambat 1 minggu atau lebih dari jadwal.		
Rencana darurat: Pengembangan perangkat keras outsourcing sebagai cadangan, kirimkan sistem pada simulator.		
Status/tanggal/orang yang bertanggung jawab: Dibuat 1,01 Jan – Fred Jones Sim. selesai 10 Februari 01 - Bill Olson		

LATIHAN SOAL

1. Mengapa manajemen risiko itu penting?
2. Pertimbangkan berkendara ke bandara untuk naik pesawat dari maskapai penerbangan yang belum pernah Anda gunakan sebelumnya. Resiko apa yang mungkin unik dalam perjalanan ke bandara ini, dan risiko apa saja yang dapat dikelola sebagai bagian dari perjalanan normal ke bandara?

Latihan Tambahan (Diskusi)

1. Analisis potensi risiko masalah klinik gigi di Bab 4, Masalah 2. Klasifikasikan risiko sebagai normal atau unik untuk proyek ini.
2. Pertimbangkan sebuah proyek yang mempunyai kemungkinan 0,5 persen untuk kesalahan yang tidak terdeteksi sehingga perusahaan harus membayar denda sebesar Rp. 1.000.000.000. Hitung eksposur risiko.
3. Pertimbangkan penggunaan tinjauan tambahan untuk Soal 2 yang memerlukan biaya Rp. 1.000.000 namun menghilangkan kesalahan tersebut sebanyak 50 persen. Hitung eksposur risiko baru ini dengan menggunakan tinjauan tambahan. Apakah pendekatan tinjauan tambahan lebih baik?
4. Apa yang akan berubah pada Soal 3 jika tinjauan tambahan hanya efektif 10 persen saja?
5. Buatlah pohon keputusan untuk permasalahan pada Contoh 6.3 jika pada Game A, bayarannya adalah
   Rp. 50.000 dan jika biaya bermain di Game B adalah Rp. 40.000. Haruskah Anda memainkan salah satu game tersebut?
6. Perusahaan X mempunyai data historis yang menunjukkan tingkat kesalahan normal sebesar 0,0036 kesalahan per KLOC. Sebuah studi tentang teknik tinjauan baru

menunjukkan bahwa biayanya Rp. 10.000.000 per 100 KLOC dan mengurangi jumlah kesalahan sebesar 50 persen. Asumsikan bahwa setiap kesalahan merugikan perusahaan rata-rata sebesar Rp. 10.000.000. Proyek saat ini diperkirakan berukuran 50 KLOC. Hitung eksposur risiko untuk setiap pendekatan. Apakah teknik review baru ini layak dilakukan?

BAB 7
JAMINAN KUALITAS PERANGKAT LUNAK

7.1 PENDAHULUAN

Ada banyak cara untuk mendefinisikan kualitas. Tidak ada yang sempurna. Ini seperti pepatah lama, ''Saya mengetahuinya ketika saya melihatnya.''

Salah satu definisinya adalah bahwa ''kualitas adalah totalitas fitur dan karakteristik suatu produk atau layanan yang mempengaruhi kemampuannya untuk memuaskan kebutuhan tertentu'' (British Standards Institution).

Definisi lainnya adalah bahwa perangkat lunak berkualitas adalah perangkat lunak yang melakukan apa yang seharusnya dilakukan. Kurangnya kualitas lebih mudah untuk didefinisikan; itu adalah ketidakpuasan pelanggan. Ukuran yang biasa dilakukan adalah laporan cacat.

Teknik utama untuk mencapai kualitas adalah tinjauan atau penelusuran perangkat lunak. Tujuan dari inspeksi adalah untuk menemukan kesalahan. Pendekatan formal terbukti lebih berhasil dibandingkan pendekatan informal. Metrik yang paling sering digunakan untuk mengevaluasi inspeksi adalah penemuan kesalahan/KLOC. Efisiensi dapat diukur dalam bentuk kesalahan yang ditemukan/jam yang dihabiskan. Banyak eksperimen telah dilakukan mengenai berapa lama waktu persiapan yang optimal. Beberapa pekerjaan juga telah dilakukan mengenai berapa lama pertemuan inspeksi harus berlangsung.

7.2 INSPEKSI FORMAL DAN TINJAUAN TEKNIS

Inspeksi formal adalah aktivitas formal dan terjadwal di mana seorang desainer menyajikan materi tentang suatu desain dan sekelompok rekan terpilih mengevaluasi aspek teknis dari desain tersebut.

Rincian mengenai cara melakukan inspeksi formal atau tinjauan teknis dapat sangat bervariasi. Aspek-aspek berikut biasanya diterima sebagai pembeda inspeksi formal dengan tinjauan lainnya:

Rekan-rekan yang berpengetahuan digunakan.
Produser adalah peserta aktif.
Produk yang eksplisit dan lengkap diperiksa.
Tujuan utamanya adalah untuk menemukan cacat.
Inspeksi formal digunakan secara rutin dalam pengembangan perangkat lunak.
Peran khusus ditetapkan.
Inspeksi menggunakan langkah-langkah spesifik dari inspeksi formal.
Setidaknya ada tiga orang yang terlibat dalam pemeriksaan tersebut.

Peran Inspeksi

Meskipun terdapat variasi, berikut adalah peran dasar yang digunakan sebagian besar inspeksi:

Moderator—Moderator memilih tim, melakukan inspeksi, dan melaporkan hasilnya.

Pembaca—Pembaca sering kali bukanlah produsen produk; namun, pembaca akan memandu tim melalui produk kerja selama rapat inspeksi.

Perekam—Pencatat menyimpan catatan inspeksi dan melaporkan setiap cacat secara akurat.

Produser—Produser adalah orang yang pertama kali memproduksi produk. Perannya adalah menjawab pertanyaan selama pemeriksaan. Produser juga bertanggung jawab untuk memperbaiki setiap masalah yang teridentifikasi dalam inspeksi. Dia kemudian melaporkan koreksi tersebut kepada moderator.

Langkah Inspeksi

Berikut langkah-langkah dasar dalam pemeriksaan:

1. ***Ikhtisar—***Ketika produsen memenuhi kriteria masuk, inspeksi dijadwalkan. Produser kemudian melakukan tinjauan. Ini memperkenalkan anggota tim inspeksi lainnya dengan produk yang akan diperiksa.
2. ***Persiapan—***Anggota tim inspeksi mempelajari produk. Waktu yang dihabiskan dalam persiapan dikontrol berdasarkan ukuran produk di KLOC. Para anggota dapat menggunakan daftar periksa untuk fokus pada isu-isu penting.
3. ***Rapat inspeksi—***Moderator mengawasi rapat inspeksi. Beberapa pendekatan menggunakan pembaca selain produser untuk benar-benar melakukan inspeksi. Perekam membuat catatan lengkap tentang permasalahan yang diangkat. Semua anggota tim inspeksi menandatangani laporan. Setiap anggota tim dapat membuat laporan minoritas jika ada perbedaan pendapat.
4. ***Pengerjaan Ulang—***Produser meninjau laporan dan mengoreksi produk.
5. ***Tindak lanjut—***Moderator meninjau laporan dan koreksinya. Jika memenuhi kriteria keluar, pemeriksaan selesai. Jika tidak, moderator dapat meminta produsen mengerjakan ulang produknya atau menjadwalkan inspeksi ulang.

Daftar Periksa

Daftar periksa adalah daftar item yang harus diperiksa selama peninjauan. Terkadang item-item tersebut diungkapkan sebagai pertanyaan yang harus dijawab. Nilai dari daftar periksa adalah memusatkan perhatian pengulas pada potensi masalah. Setiap kesalahan yang ditemukan harus dianalisis untuk melihat apakah hal tersebut memerlukan item daftar periksa untuk fokus pada masalah tersebut. (Saya ingat proses debugging yang panjang yang disebabkan oleh titik koma tepat setelah kondisi dalam pernyataan IF di C. Sekarang daftar periksa apa pun untuk program C yang saya tulis mencakup pemeriksaan titik koma di akhir kondisi keputusan.). Item daftar periksa yang tidak efektif dalam menemukan kesalahan selama inspeksi harus dipertimbangkan untuk dihapus. Terlalu banyak item daftar periksa akan mengurangi efektivitas pemeriksaan.

7.3 KEANDALAN PERANGKAT LUNAK

Keandalan adalah kemungkinan tidak terjadinya kegagalan dalam jangka waktu tertentu. Hal ini biasanya dilambangkan dengan $R(n)$, dimana n adalah banyaknya satuan

waktu. Jika satuan waktunya hari, maka $R(1)$ adalah peluang tidak gagalnya 1 hari. Probabilitas kegagalan dalam jangka waktu tertentu adalah 1 dikurangi reliabilitas untuk jangka waktu tersebut $(F(n) = 1 - R(n))$.

Keandalan perangkat lunak merupakan ukuran seberapa sering perangkat lunak menghadapi masukan data atau kondisi lain yang tidak diproses dengan benar untuk menghasilkan jawaban yang benar. Keandalan perangkat lunak tidak berkaitan dengan keausan perangkat lunak. Analogi yang lebih baik untuk kegagalan perangkat lunak adalah mengambil kelereng dari tas atau melempar anak panah dengan mata tertutup ke balon di dinding.

Tingkat Kesalahan

Jika kesalahan terjadi setiap 2 hari, maka tingkat kesalahan sesaat adalah 0,5 kesalahan per hari. Tingkat kesalahan merupakan kebalikan dari waktu antar kesalahan (inter-error time). Tingkat kesalahan dapat digunakan sebagai perkiraan kemungkinan kegagalan, F(1). Kecuali kita mengetahui suatu tren, perkiraan terbaik mengenai perilaku jangka pendek di masa depan adalah perilaku saat ini. Jadi jika kita menemukan 20 kesalahan dalam satu hari, perkiraan terbaik kita untuk hari berikutnya adalah 20 kesalahan.

Contoh 7.1

Jika error terjadi setelah 2 hari, berapa peluang sistem tidak mengalami kegagalan dalam 1, 2, 3, dan 4 hari?

Jika kesalahan terjadi setiap 2 hari, kita dapat menggunakan 0,5 sebagai tingkat kesalahan sesaat. Ini juga dapat digunakan untuk memperkirakan probabilitas kegagalan selama 1 hari. Jadi, F(1) = 0,5. Lalu, R(1) = 1 — F(1) = 0:5. R(2) = 0:25. R(3) = 0:125. R(4) = 0:0625.

Jika kita dapat melihat tren tingkat kesalahan, maka kita dapat memperkirakan tingkat kesalahan dengan lebih baik. Daripada menggunakan persamaan untuk menyesuaikan data, plot tingkat kegagalan dapat digunakan untuk memvisualisasikan perilaku.

Jika x adalah waktu antar kegagalan, 1/x adalah laju kegagalan sesaat. Plot tingkat kegagalan sesaat versus angka kegagalan atau waktu kegagalan yang telah berlalu. Cobalah untuk membuat garis lurus pada titik-titik tersebut. Nilai garis pada waktu saat ini dapat digunakan untuk tingkat kesalahan.

Perpotongan garis ini dengan sumbu horizontal menunjukkan jumlah kesalahan dimana tingkat kegagalan menjadi nol, atau jumlah waktu yang diperlukan untuk menghilangkan semua kesalahan. Jika sumbu x adalah waktu yang berlalu, maka luas di bawah garis lurus (satuannya adalah waktu kegagalan/waktu) mewakili jumlah gangguan.

Dengan demikian, data empiris tentang seberapa sering perangkat lunak gagal selama pengujian atau observasi digunakan untuk memperkirakan laju saat ini. Ide teoretis akan digunakan untuk menyempurnakan prediksi untuk jangka waktu yang lebih lama.

Teori Probabilitas

$\mathrm{F}(1)$ adalah probabilitas gagal pada eksekusi berikutnya. Hal ini sama dengan theta, yaitu persentase kasus uji yang gagal. Probabilitas kegagalan dapat diperkirakan dengan tingkat kesalahan sesaat saat ini atau dengan perkiraan tingkat kesalahan dari plot tingkat kesalahan.

Jika kita mengetahui $\mathrm{R}(1)$, maka peluang kita dapat mengeksekusi n kasus uji tanpa kegagalan adalah $\mathrm{R}(n) = \mathrm{R}(1)^n$.
Perhatikan bahwa $\mathrm{F}(n)$ bukan $\mathrm{F}(1)^n$. $\mathrm{F}(n) = 1 - (1 - \mathrm{F}(1))^n$.

7.4 PENJAMINAN MUTU STATISTIK

Jaminan kualitas statistik (SQA) adalah penggunaan statistik untuk memperkirakan kualitas perangkat lunak. Mengeksekusi kode dengan sekumpulan kecil kasus uji yang dipilih secara acak akan memberikan hasil yang dapat digunakan untuk memperkirakan kualitas. Hal ini terkadang disebut pemeriksaan perangkat lunak. Tingkat kesalahan pada sampel yang dipilih secara acak dapat digunakan sebagai perkiraan tingkat kesalahan dalam proyek yang telah selesai.

Jika persentase pelaksanaan yang benar tinggi, maka pembangunan berjalan baik. Jika persentase pelaksanaan yang benar rendah, maka tindakan perbaikan mungkin tepat untuk proses pembangunan.

7.5 STANDAR IEEE UNTUK RENCANA SQA

Bagian penting dari pencapaian kualitas adalah merencanakan kualitas, yaitu merencanakan aktivitas-aktivitas yang akan membantu mencapai kualitas. Asosiasi Standar IEEE telah mengembangkan standar (Std 730-1989) untuk rencana jaminan kualitas perangkat lunak. Berikut ini adalah bagian dari bagian yang ditentukan dalam IEEE Std 730-1989:

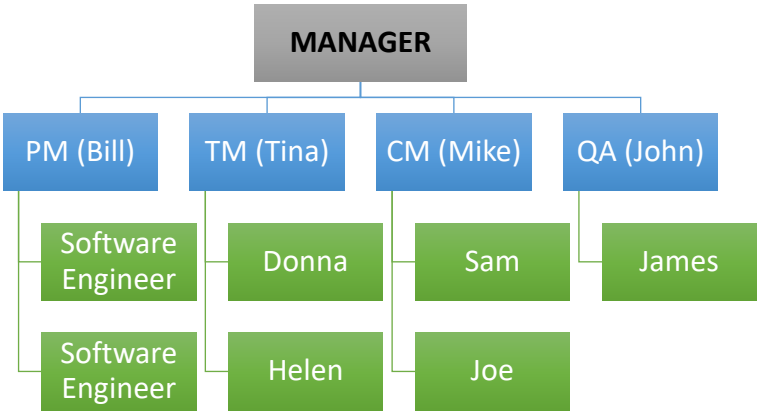
1. ***Tujuan***—Bagian ini berisi daftar perangkat lunak yang tercakup dan bagian siklus hidup perangkat lunak yang tercakup.
2. ***Dokumen Referensi***—Bagian ini berisi daftar semua dokumen yang direferensikan dalam rencana.
3. ***Manajemen***
   - 3.1 *Organisasi*—Bagian ini menjelaskan struktur organisasi dan tanggung jawabnya, dan biasanya mencakup bagan organisasi.
   - 3.2 *Tugas*—Bagian ini harus mencantumkan semua tugas yang harus dilakukan, hubungan antara tugas dan pos pemeriksaan, dan urutan tugas.
   - 3.3 *Tanggung Jawab*—Bagian ini mencantumkan tanggung jawab setiap unit organisasi.
4. ***Dokumentasi***
   - 4.1 *Tujuan*—Bagian ini berisi daftar semua dokumen yang diperlukan dan menyatakan bagaimana dokumen akan dievaluasi.
   - 4.2 *Dokumen minimum*—Bagian ini menjelaskan dokumentasi minimum yang diperlukan, biasanya mencakup hal-hal berikut:
     SRS—Spesifikasi Persyaratan Perangkat Lunak
     SDD—Deskripsi Desain Perangkat Lunak
     SVVP—Rencana Verifikasi dan Validasi Perangkat Lunak
     SVVR—Laporan Verifikasi dan Validasi Perangkat Lunak
     Dokumentasi pengguna—Manual, panduan

SCMP—Rencana Manajemen Konfigurasi Perangkat Lunak

5. ***Standar, Praktik, Konvensi, dan Metrik***
   Bagian ini harus mengidentifikasi S, P, C, dan M yang akan diterapkan dan bagaimana kepatuhan harus dipantau dan dijamin. Isi minimal harus mencakup standar dokumentasi, standar struktur logika, standar pengkodean, standar pengujian, produk SQA yang dipilih, dan metrik proses.
6. ***Peninjauan dan Audit—***Bagian ini akan menjelaskan peninjauan/audit apa yang akan dilakukan, bagaimana peninjauan/audit tersebut akan dilakukan, dan tindakan lebih lanjut apa yang diperlukan.
7. ***Pengujian—***Bagian ini mencakup semua pengujian yang tidak termasuk dalam SVVP.
8. ***Pelaporan Masalah—***Bagian ini harus menjelaskan praktik dan prosedur untuk melaporkan, melacak, dan menyelesaikan masalah, termasuk tanggung jawab organisasi.
9. ***Alat, Teknik, dan Metodologi—***Bagian ini akan mengidentifikasi alat, teknik, dan metodologi perangkat lunak khusus serta menjelaskan penggunaannya.
10. ***Pengendalian Kode—***Bagian ini harus menjelaskan metode dan fasilitas untuk memelihara versi perangkat lunak yang terkontrol.
11. ***Pengendalian Media—***Bagian ini harus menjelaskan metode dan fasilitas untuk mengidentifikasi, menyimpan, dan melindungi media fisik.
12. ***Pengendalian Pemasok (untuk outsourcing)—***Bagian ini menyatakan ketentuan untuk memastikan bahwa perangkat lunak yang disediakan oleh pemasok memenuhi standar.
13. ***Catatan—***Bagian ini harus mengidentifikasi dokumentasi yang harus disimpan dan metode pengumpulan, pemeliharaan, dan pengamanan dokumentasi.
14. ***Pelatihan—***Bagian ini akan mengidentifikasi kegiatan pelatihan yang diperlukan.
15. ***Manajemen Risiko—***Bagian ini menjelaskan metode dan prosedur manajemen risiko.

Contoh 7.2

Mengembangkan Bagian 3 dan Bagian 8 dari rencana SQA untuk proyek pengembangan perangkat lunak. Asumsikan seorang manajer proyek bernama Bill; tim penguji eksternal yang terdiri dari Tina, pemimpin, Donna, dan Helen; grup manajemen konfigurasi (CM) terpisah yang terdiri dari Mike, Sam, dan Joe; dan tim penjaminan mutu (QA) eksternal terpisah yang terdiri dari John dan James.
Lihat Gambar 7-1.



Gambar 7-1. Bagian 3 Rencana SQA.

Bagian 3

3.1 Organisasi

3.2 Tugas

Semua dokumen akan ditinjau. Alat manajemen konfigurasi akan mengelola semua dokumen dan modul kode sumber. Semua rencana pengujian akan dilakukan selama fase persyaratan dan mencakup jumlah kasus pengujian yang memadai. Inspeksi formal akan dilakukan pada akhir setiap fase.

7.6 TANGGUNG JAWAB

Tim proyek bertanggung jawab atas semua pengembangan, termasuk persyaratan, desain, dan implementasi. Tim proyek menghasilkan rencana pengujian sebagai bagian dari persyaratan. Mereka juga bertanggung jawab atas semua dokumentasi, termasuk panduan pengguna dan dokumen pelatihan.

Tim penguji bertanggung jawab untuk menguji versi dasar kode sumber. Tim penguji akan menggunakan rencana pengujian yang dikembangkan selama persyaratan. Kasus uji tambahan akan dikembangkan untuk memenuhi cakupan setiap pernyataan kode. Setiap perbedaan dalam rencana pengujian dan/atau persyaratan atau pengujian akan dilaporkan kepada manajer keseluruhan. Tim manajemen konfigurasi akan bertanggung jawab untuk menerima item konfigurasi perangkat lunak dan menetapkan nomor versi. Tim penjaminan mutu akan bertanggung jawab mengawasi semua tinjauan, penelusuran, dan inspeksi. Tim QA akan melacak semua laporan masalah.

Bagian 8

Semua masalah yang teridentifikasi di luar unit pengembangan harus dilaporkan ke tim QA untuk diberi nomor laporan masalah. Manajer masing-masing tim akan menyetujui koreksi laporan masalah yang ditugaskan kepada tim tersebut. Tim QA akan bertanggung jawab untuk melacak semua masalah dan melaporkan mingguan kepada manajer keseluruhan.

LATIHAN SOAL

1. Apa perbedaan antara tinjauan dan tinjauan teknis formal?
2. Bagaimana cara mengevaluasi daftar periksa?

3. Revisi checklist apa saja yang harus dilakukan?
4. Faktor-faktor apa yang mempengaruhi efektivitas tinjauan teknis formal?
5. Bagaimana efektivitas tinjauan teknis formal diukur?
6. Apa yang terjadi jika produsen tidak dapat menyelesaikan suatu permasalahan?
7. Apa yang terjadi jika satu atau lebih anggota tim inspeksi tidak setuju dengan mayoritas?
8. Jika sebuah dadu jujur dilempar sebanyak 5 kali, berapa peluang munculnya angka 6 pada salah satu pelemparan?
9. Jika sebuah dadu jujur dilempar 5 kali, berapakah peluang munculnya angka 6 pada paling sedikit salah satu pelemparan?

Latihan Tambahan (Diskusi)

1. Buatlah daftar periksa untuk meninjau kode C++.
2. Buatlah daftar periksa untuk meninjau desain perangkat lunak.
3. Gambarkan model proses untuk inspeksi formal.
4. Dengan asumsi bahwa pengujian tersebut mewakili situasi operasional, hitung keandalan sistem perangkat lunak yang memiliki 10 kesalahan dalam 200 kasus pengujian.
5. Asumsikan teknik FTR A membutuhkan 2 jam/persiapan KLOC dan memberikan waktu 1 jam/waktu review KLOC dan teknik FTR B memerlukan 1 jam/waktu persiapan KLOC dan 4 jam/waktu review KLOC. Asumsikan juga bahwa dalam eksperimen terkontrol dengan kode sumber yang sama, teknik A menemukan 12 kesalahan/KLOC dan B menemukan 14 kesalahan/KLOC. Bandingkan kedua teknik untuk efektivitasnya.
6. Jika perangkat lunak mengalami 5 kegagalan dalam 100 pengujian selama 10 hari pengujian, berapa perkiraan yang baik mengenai keandalan perangkat lunak pada hari berikutnya? Pekan?
7. Mengembangkan rencana SQA untuk masalah B&B (Bab 4, Masalah 3).
8. Kesalahan 1 terjadi setelah 4 hari, dan kesalahan 2 terjadi 5 hari kemudian. Plot grafik tingkat kesalahan versus angka kesalahan dan grafik tingkat kesalahan versus waktu, dan perkirakan jumlah kesalahan dalam sistem dan waktu untuk menghilangkan semua kesalahan.

BAB 8
PERSYARATAN

8.1 PENDAHULUAN

Tujuan dari fase persyaratan adalah untuk memperoleh persyaratan dari pengguna. Hal ini biasanya dicapai melalui pengembangan diagram dan spesifikasi kebutuhan setelah berdiskusi dengan pengguna. Pengguna kemudian meninjau diagram dan spesifikasi untuk menentukan apakah pengembang perangkat lunak telah memahami persyaratannya. Oleh karena itu, diagram dan spesifikasi harus dikomunikasikan kembali kepada pengguna tentang aspek-aspek penting yang diperlukan dari perangkat lunak yang akan diproduksi. Bagian berikut menjelaskan diagram dan spesifikasi kebutuhan yang berguna dalam mencapai komunikasi ini.

8.2 MODEL OBJEK

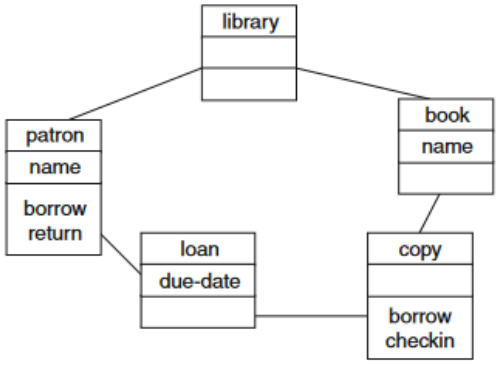
Pendekatan dasar dalam metodologi berorientasi objek (OO) adalah mengembangkan model objek (lihat Bagian 2.4) yang menggambarkan subset dunia nyata yang merupakan domain masalah. Tujuannya adalah memodelkan domain masalah dan bukan merancang implementasi. Dengan demikian, entitas-entitas yang penting untuk memahami masalah akan dilibatkan meskipun mereka tidak akan dilibatkan dalam solusi. Atribut dan metode yang disertakan dalam model objek juga diperlukan untuk memahami masalah dan bukan hanya penting untuk solusi.

Berikut ini adalah aturan model objek untuk persyaratan:

1. Semua entitas dunia nyata yang penting untuk memahami domain masalah harus disertakan.
2. Semua metode dan atribut yang penting untuk memahami domain masalah harus disertakan.
3. Objek, atribut, dan metode yang hanya signifikan bagi implementasi tidak boleh dicantumkan.

Contoh 8.1

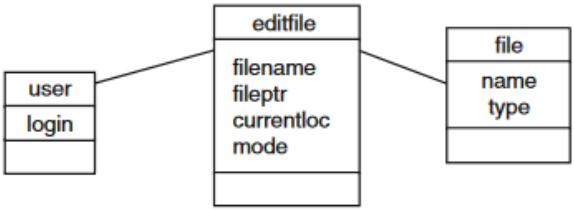
Gambarlah model objek untuk masalah perpustakaan. Lihat Gambar 8-1.



Gambar 8.1. Model objek untuk masalah perpustakaan.

Contoh 8.2

Gambarlah model objek untuk editor mirip vi yang disederhanakan. Lihat Gambar 8-2.



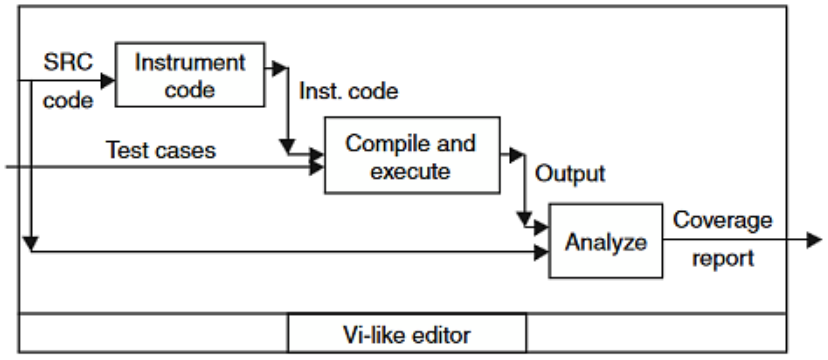
Gambar 8-2. Model objek untuk editor mirip vi yang disederhanakan.

8.3 PEMODELAN ALIRAN DATA

Meskipun tidak banyak digunakan dalam pengembangan OO, diagram aliran data (lihat Bagian 2.2) merupakan bagian penting dari pengembangan perangkat lunak sebelum OO. Diagram aliran data (DFD) masih memiliki peran penting dalam spesifikasi banyak sistem. Pentingnya diagram aliran data adalah dalam menentukan data apa yang tersedia untuk suatu komponen. Mengetahui data yang tersedia sering kali membantu dalam memahami apa yang diharapkan dilakukan oleh suatu komponen dan bagaimana komponen tersebut akan menyelesaikan tugasnya.

Contoh 8.3

Gambarlah DFD untuk Unix sederhana, editor mirip vi. Lihat Gambar 8-3.



Gambar 8-3. Diagram aliran data untuk Unix, editor mirip vi.

8.4 PEMODELAN PERILAKU

Pemodelan perilaku mengacu pada perilaku sistem, biasanya dari sudut pandang pengguna. Diagram ini digunakan untuk menentukan aspek sistem yang diusulkan. Penting agar diagram menangkap aspek-aspek penting dari sistem dan mampu mengkomunikasikan aspek-aspek tersebut kepada pengembang dan pengguna untuk konfirmasi bahwa ini adalah sistem yang diinginkannya.

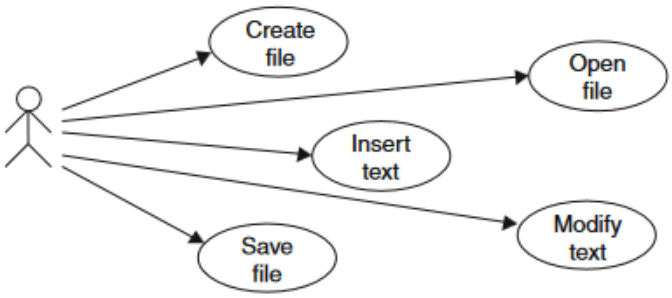
Kasus Penggunaan

Diagram use case mewakili fungsionalitas sistem dari sudut pandang pengguna (lihat Bagian 2.5). Semua fungsi penting harus disebutkan. Namun, fungsi rutin yang tersirat dalam

frasa tingkat tinggi tidak harus disebutkan secara spesifik (bahaya miskomunikasi harus diimbangi dengan kejelasan). Persyaratan tekstual akan merinci fungsi-fungsi individual ini.

Contoh 8.4

Gambarlah diagram kasus penggunaan untuk editor yang mirip dengan editor mirip Unix vi yang disederhanakan. Lihat Gambar 8-4.



Gambar 8-4. Gunakan diagram kasus untuk editor.

Fungsi penting dalam diagram ini adalah manipulasi file (membuat, menyimpan, dan membuka). Sisipkan dan modifikasi dimaksudkan sebagai frasa tingkat tinggi yang mencakup fungsi pengeditan teks pada umumnya. Perhatikan bahwa beberapa kemampuan seperti pencarian, penyalinan, dan pemindahan mungkin diabaikan, karena tidak disebutkan secara eksplisit.

Skenario

Skenario adalah serangkaian tindakan yang menyelesaikan tugas pengguna. Urutan alternatif hanya ditampilkan dengan memiliki skenario tersendiri untuk setiap alternatif. Skenario digunakan untuk menggambarkan kemampuan penting atau usulan penggunaan sistem.

Dalam UML, diagram interaksi (lihat Bab 2) digunakan untuk menentukan skenario. Skenario juga dapat ditentukan dengan membuat daftar urutan tindakan.

Contoh 8.5

Tulis skenario untuk editor vi yang disederhanakan menggunakan setiap kasus penggunaan pada Contoh 8.4. Gunakan titik koma untuk memisahkan tindakan. Gunakan tanda kurung untuk memuat komentar atau ketentuan.

```
Create file
      vi filename (file does not already exist)
Open file
      vi filename (file already exists)
Insert text
      I ; <desired text> ; <esc>
      i ; <desired text> ; <esc>
      0 ; <desired text> ; <esc>
      o ; <desired text> ; <esc>
      A ; <desired text> ; <esc>
      a ; <desired text> ; <esc>
Modify text
```

```
        cw ; <new text> ; <esc>
        dw
        dd
        x
Save file
        ZZ
```

Catatan: Tidak semua urutan ditampilkan. Agar singkatnya, tidak semua operasi ditampilkan. Dalam spesifikasi aktual, upaya harus dilakukan untuk menunjukkan seluruh operasi dan urutan operasi yang signifikan. Dalam contoh ini, setiap skenario hanya mewakili sebagian penggunaan. Alternatifnya, setiap skenario dapat dijalankan dari file terbuka hingga file tertutup.

Diagram State

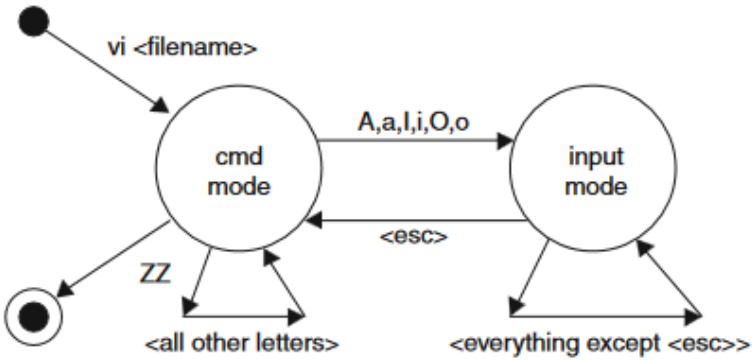
Rincian diagram keadaan dibahas dalam Bab 2. Ketika digunakan sebagai bagian dari spesifikasi persyaratan, penting bahwa negara mencerminkan kondisi domain yang dapat dimengerti oleh pengguna. Negara-negara yang hanya signifikan terhadap implementasi harus digabungkan menjadi negara-negara domain yang signifikan. Selain itu, transisi yang diizinkan harus mencakup semua transisi yang diizinkan dari skenario. Urutan yang tidak dimaksudkan dalam sistem yang diusulkan tidak boleh diizinkan dalam diagram keadaan. Hal ini mungkin sulit dilakukan, karena adanya transisi dalam suatu skenario tidak menghalangi transisi lainnya.

Berikut ini adalah aturan untuk menggunakan diagram keadaan dalam spesifikasi kebutuhan:

1. Semua negara bagian harus merupakan domain signifikan.
2. Semua urutan dari skenario harus diperbolehkan.
3. Semua skenario yang dilarang tidak boleh dibiarkan.

Contoh 8.6

Gambarlah diagram keadaan untuk editor mirip vi yang sederhana. Lihat Gambar 8-5.



Gambar 8-5. Diagram status untuk editor sederhana seperti vi.

Diagram keadaan ini dibangun dengan menggunakan pemahaman program untuk menggabungkan negara-negara menjadi negara-negara yang mempunyai relevansi domain. Diagram ini mewakili dengan baik perilaku editor mirip vi sederhana yang diusulkan kepada pengguna.

8.5 KAMUS DATA

Kamus data adalah tabel informasi tentang setiap elemen data dalam sistem. Awalnya, pada fase persyaratan kamus data akan menjadi item data dari domain masalah. Entri tipikal akan menyertakan nama item, di kelas mana item tersebut berada, tipe item data, dan semantik item.

CONTOH 8.7

Bangun kamus data untuk masalah perpustakaan pada Contoh 2.6.

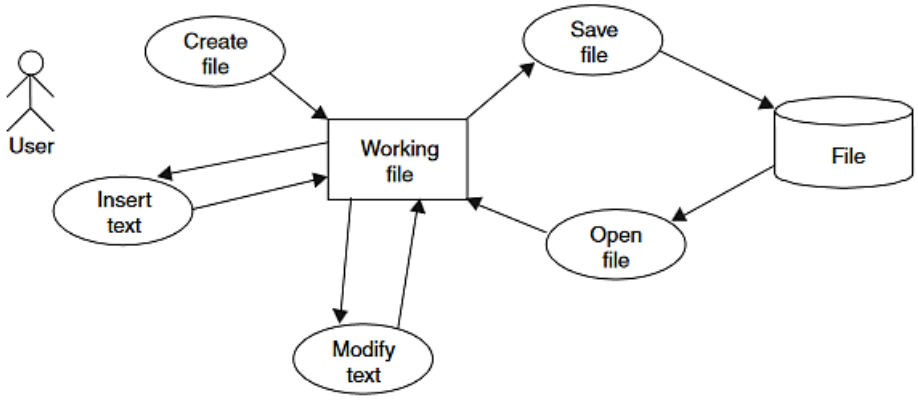
Name	Class	Type	Size	Semantics
Author Book	Book	String	< 40 char	Last name, first name (may be truncated)
Book	Book	Object		Abstract concept of the book
Book IDE	Copy	Key		Key to info about the book
Borrower	Loan	Key		Key to patron who made this loan
Copy	Copy	Object		Library's physical copy of a book
Copy IDE	Copy	Key		Key to physical copy being borrowed
ISBN	Book	String	10-20 char	International Standard Book Number
Loan	Loan	Object		A borrowing that is still active
Name	Patron	String	< 40 char	Last name, first name (may be truncated)
Patron	Patron	Object		Registered holder of library card
Title	Book	String	< 50 char	First 50 char of title from title page

8.6 DIAGRAM SISTEM

Diagram sistem adalah diagram yang didefinisikan secara nonformal yang digunakan untuk memberikan gambaran umum tentang sistem yang diusulkan. Hal ini sering digunakan ketika diagram yang didefinisikan secara formal terlalu terbatas untuk mengungkapkan gambaran umum yang diperlukan. Diagram sistem biasanya menggabungkan aspek aliran data dan diagram use case. Mereka biasanya memiliki oval yang mewakili bagian pemrosesan sistem, objek data yang mewakili file dan/atau database, kotak yang mewakili data, dan gambar tempel yang mewakili orang. Busur digunakan untuk menunjukkan aliran masuk dan keluar fungsi. Tantangan dalam diagram sistem adalah menjaga konsistensi dalam penggunaan simbol dan memberikan rincian yang memadai.

Contoh 8.8

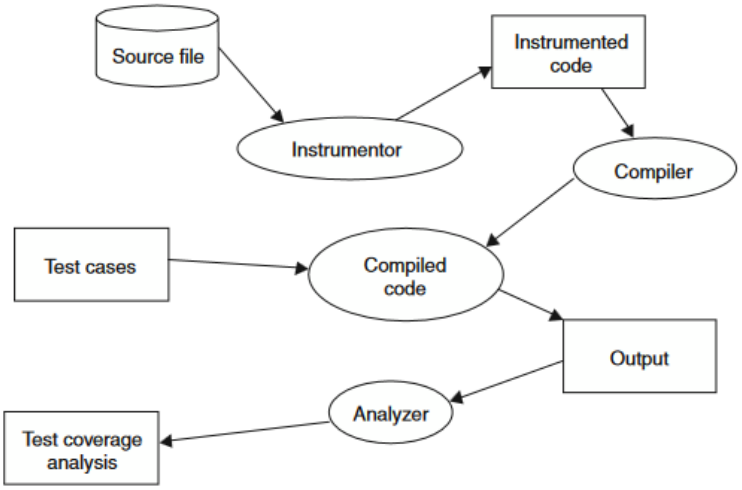
Gambarlah diagram sistem untuk editor mirip vi yang sederhana. Lihat Gambar 8-6.



Gambar 8-6. Diagram sistem untuk editor sederhana seperti vi.

CONTOH 8.9

Gambarkan diagram sistem untuk alat pengujian yang menginstrumentasikan kode sumber, mengkompilasi kode yang diinstrumentasi, mengeksekusi kode tersebut dengan kasus pengujian, dan kemudian menganalisis hasilnya. Lihat Gambar 8-7. Perhatikan bahwa keluaran dari proses kompiler adalah proses lain.



Gambar 8-7. Diagram sistem untuk alat pengujian.

8.7 STANDAR IEEE UNTUK SPESIFIKASI PERSYARATAN PERANGKAT LUNAK

Garis besar SRS berikut ini berdasarkan IEEE 830-1993:

1. ***Pendahuluan***—Bagian ini dimaksudkan untuk memberikan gambaran umum tentang spesifikasi lainnya.
   - 1.1 *Tujuan*—Bagian ini harus menjelaskan tujuan SRS dan audiens yang dituju.
   - 1.2 *Ruang Lingkup*—Bagian ini harus mengidentifikasi produk, menjelaskan apa yang akan dan tidak akan dilakukan produk, dan menjelaskan penerapan perangkat lunak, termasuk manfaat, sasaran, dan sasaran.
   - 1.3 *Definisi*—Bagian ini harus mengidentifikasi semua istilah, akronim, dan singkatan yang digunakan dalam spesifikasi.
   - 1.4 *Referensi*—Bagian ini harus mengidentifikasi semua dokumen yang dirujuk di tempat lain dalam spesifikasi.
   - 1.5 *Ikhtisar*—Bagian ini harus menjelaskan isi dokumen lainnya.
2. ***Deskripsi Keseluruhan***—Bagian ini dimaksudkan untuk memberikan latar belakang untuk memahami persyaratan lainnya.
   - 2.1 *Perspektif Produk*—Bagian ini harus menempatkan produk dalam perspektif dengan produk lain. Biasanya mencakup diagram blok dari sistem yang lebih besar. Ini harus menentukan batasan (misalnya, antarmuka sistem dengan perangkat lunak lain), antarmuka pengguna (misalnya, format layar, pengaturan waktu), antarmuka perangkat keras, antarmuka perangkat lunak (misalnya, versi perangkat lunak yang dihubungkan), batasan memori, operasi (misalnya, mode operasi). tions), dan kendala adaptasi lokasi.

2.2 *Fungsi Produk*—Bagian ini harus mencakup ringkasan fungsi utama produk.
2.3 *Karakteristik Pengguna*—Bagian ini harus mencakup tingkat pendidikan, pengalaman, dan keahlian teknis pengguna.
2.4 *Kendala*—Bagian ini harus mencakup batasan lain (misalnya peraturan) yang tidak tercakup dalam Bagian 2.1.
2.5 *Asumsi dan Ketergantungan*—Bagian ini harus mencakup asumsi apa pun yang, jika tidak benar, memerlukan perubahan pada persyaratan.
2.6 *Pembagian Persyaratan*—Bagian ini harus mengidentifikasi persyaratan yang mungkin tertunda pada versi produk mendatang.

3. ***Persyaratan Spesifik***—Menurut Standar IEEE 830: ''Bagian SRS ini harus berisi semua persyaratan perangkat lunak hingga tingkat detail yang cukup untuk memungkinkan perancang merancang sistem untuk memenuhi persyaratan tersebut, dan penguji dapat menguji apakah sistem tersebut memenuhi persyaratan tersebut. sistem memenuhi persyaratan tersebut.'' Ini adalah kriteria penting untuk diingat: SRS harus cukup rinci sehingga desain dan pengujian dapat dibuat langsung dari SRS. Selain itu, menurut Standar IEEE 830: ''Persyaratan ini harus mencakup minimal deskripsi setiap masukan (stimulus) ke dalam sistem, setiap keluaran (respons) dari sistem dan semua fungsi yang dilakukan oleh sistem sebagai respons terhadap suatu kondisi. masukan atau untuk mendukung keluaran."
   3.1 *Persyaratan Antarmuka Eksternal*—Bagian ini harus menjelaskan semua masukan dan keluaran sistem. Ini merinci informasi dari Bagian 2.1
   3.2 *Fungsi*—Bagian ini harus menjelaskan semua fungsi sistem. Hal ini harus mencakup pemeriksaan validitas masukan, respons terhadap situasi abnormal, pengaruh parameter, dan hubungan keluaran dengan masukan.
   3.3 *Persyaratan Kinerja*—Bagian ini harus menjelaskan persyaratan statis dan dinamis.
   3.4 *Kendala Desain*—Bagian ini harus menjelaskan segala kendala pada desain.

LATIHAN SOAL

1. Apa kelebihan DFD dibandingkan diagram lainnya?
2. Apa tujuan spesifikasi perilaku dan diagram dalam spesifikasi kebutuhan?
3. Fungsi apa yang penting untuk disertakan dalam diagram use case?
4. Kriteria apa yang harus digunakan untuk mengevaluasi skenario?
5. Kriteria apa yang harus digunakan untuk mengevaluasi diagram keadaan?
6. Apa keuntungan dari diagram sistem?
7. Apa masalah utama pada diagram sistem?

Latihan Tambahan (Diskusi)

1. Gambarkan model objek untuk permasalahan B&B (lihat Soal 4.3).
2. Gambarkan model objek untuk permasalahan kantor gigi (lihat Soal 4.2).
3. Gambarkan DFD untuk sistem B&B (lihat Soal 4.3).
4. Gambarlah DFD untuk sistem kantor gigi (lihat Soal 4.2)

5. Gambarkan diagram use case untuk masalah B&B (lihat Soal 4.3).
6. Gambarkan diagram use case untuk permasalahan klinik gigi (lihat Soal 4.2).
7. Tuliskan skenario untuk masalah B&B (lihat Soal 4.3).
8. Tuliskan skenario untuk permasalahan klinik gigi (lihat Soal 4.2).
9. Gambarlah diagram keadaan untuk permasalahan B&B secara keseluruhan (lihat Soal 4.3).
10. Gambarkan diagram keadaan untuk reservasi item data pada permasalahan B&B (lihat Soal 4.3).
11. Gambarlah diagram keadaan untuk permasalahan kantor gigi (lihat Soal 4.2).
12. Gambarlah diagram sistem untuk sistem B&B (lihat Soal 4.3).
13. Gambarlah diagram sistem untuk sistem kantor gigi (lihat Soal 4.2).

BAB 9
DESAIN PERANGKAT LUNAK

9.1 PENDAHULUAN

Desain adalah ''proses penerapan berbagai teknik dan prinsip untuk tujuan mendefinisikan perangkat, proses, atau sistem dengan detail yang cukup untuk memungkinkan realisasi fisiknya.'' Desain juga merupakan bagian yang paling artistik atau kreatif dari sebuah karya. proses pengembangan perangkat lunak. Hanya sedikit aturan yang dapat ditulis untuk memandu desain.

Proses desain mengubah “apa” dari persyaratan menjadi “bagaimana” dalam desain. Hasil dari tahap desain harus berupa dokumen yang memiliki detail yang cukup untuk memungkinkan sistem diimplementasikan tanpa interaksi lebih lanjut dengan pembuat spesifikasi atau pengguna.

Proses desain juga mengubah terminologi dari ruang masalah persyaratan menjadi ruang solusi implementasi. Beberapa penulis berbicara tentang objek analisis berorientasi objek (OOA), yang berada dalam ruang masalah/domain, dan objek desain berorientasi objek (OOD), yang berada dalam ruang solusi/implementasi. Misalnya saja, dalam ruang masalah kita dapat berbicara tentang objek dunia nyata seperti seseorang; di ruang solusi kita dapat berbicara tentang kelas C yang disebut person.

Gunter dkk. menulis tentang fenomena di lingkungan (dunia) dan fenomena di implementasi (mesin). Suatu fenomena dapat terlihat atau tersembunyi. Persyaratan yang berorientasi pada pengguna dapat dinyatakan dalam bentuk fenomena, tersembunyi atau terlihat, dari lingkungan. Namun, spesifikasi yang akan digunakan sebagai dasar pengembangan harus berada di antara lingkungan dan implementasi dan harus dinyatakan dalam fenomena yang terlihat dari masing-masing hal. Spesifikasi ini merupakan titik awal untuk desain dan akan disebut spesifikasi pengembangan dalam buku ini.

Contoh 9.1

Robot diperlukan untuk menemukan merek kaleng pop tertentu menggunakan kamera hitam putih dan mengembalikan kaleng tersebut ke lokasi daur ulang. Pernyataan seperti itu dapat menjadi persyaratan yang berorientasi pada pengguna dan terdiri dari fenomena dari lingkungan. Namun, kaleng pop merupakan fenomena tersembunyi di lingkungan. Artinya, implementasinya tidak akan mengenal pop kaleng; ia akan mengetahui tentang gambar hitam-putih kaleng pop. Inilah fenomena yang terlihat. Ketika spesifikasi yang akan digunakan sebagai titik awal desain ditulis, perlu dibicarakan dalam kerangka gambar-gambar tersebut. Akan diasumsikan (dan mungkin perlu diverifikasi) bahwa hanya kaleng pop asli yang akan memberikan gambar tersebut. Misalnya, permasalahan akan jauh lebih rumit jika dinding lingkungan dipenuhi iklan yang memuat gambar kaleng pop.

Contoh 9.2

Identifikasi fenomena mana yang ada di lingkungan dan mana yang di implementasi dalam sistem perpustakaan. Buku fisik adalah fenomena yang tersembunyi di lingkungan.

Sistem tidak pernah tahu tentang buku itu. Ketika pustakawan memindai buku, dia sebenarnya memindai kode batang. Kode batang ini bukan ISBN tetapi harus mencerminkan kemungkinan banyak salinan dari satu buku. Kode batang ini dapat dilihat oleh lingkungan. Implementasinya mungkin menggunakan pengidentifikasi atau penunjuk berbeda untuk data buku. Pengidentifikasi internal ini disembunyikan dalam implementasi.

Spesifikasi pengembangan perlu ditulis dalam bentuk bar code di buku. Baik buku fisik maupun pengenal internal tidak boleh disebutkan dalam spesifikasi pengembangan.

9.2 TAHAPAN PROSES DESAIN

Berikut tahapan dalam desain:

Desain data—Fase ini menghasilkan struktur data.

Desain arsitektur—Fase ini menghasilkan unit struktural (kelas).

Desain antarmuka—Fase ini menentukan antarmuka antar unit.

Desain prosedural—Fase ini menentukan algoritma masing-masing metode.

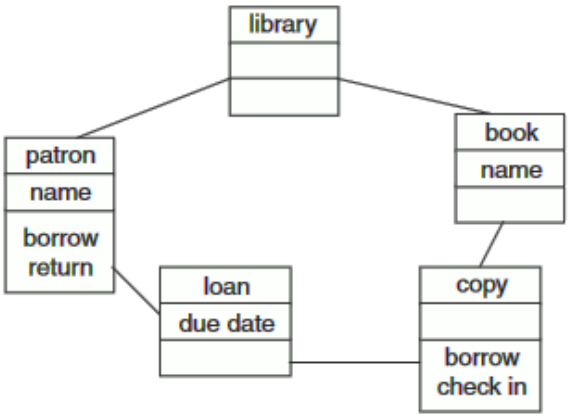
CONTOH 9.3

Rancang kelas perpustakaan/struktur data dari item data dalam model objek yang ditunjukkan pada Gambar 9-1 untuk masalah perpustakaan (lihat Contoh 8.1 dan 2.6).

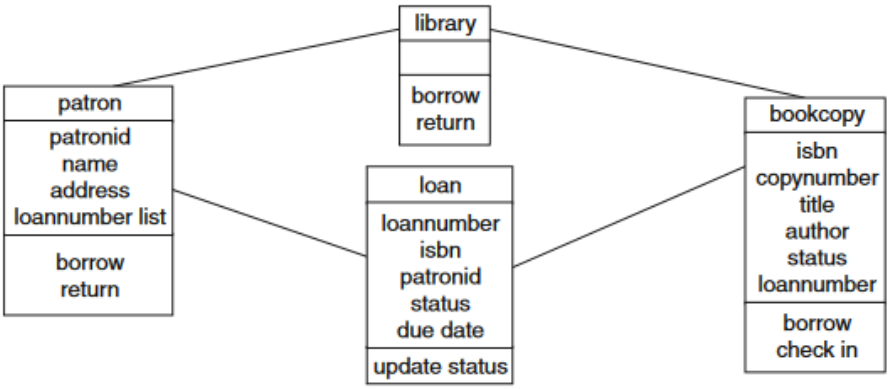
Desain data dan fase arsitektur telah digabungkan dalam contoh ini. Konsentrasi dalam contoh ini adalah pada fungsi peminjaman dan pembayaran, dengan sedikit memperhatikan tugas-tugas lain yang diperlukan, seperti administrasi, pembuatan katalog, penetapan denda yang telah jatuh tempo, penghentian buku, dan pemeliharaan patron.

Entitas domain ''buku'' mungkin tidak akan melanjutkan desainnya. Ini akan digabungkan dengan ''salinan'' ke dalam kelas/struktur data yang menyimpan semua informasi tentang salinan. Mungkin akan menggunakan ISBN dan nomor salinan sebagai pengidentifikasi unik. Informasi pelindung akan disimpan dalam struktur data kedua. Setiap catatan mungkin diidentifikasi dengan nomor ID pelindung yang unik. Informasi pinjaman mungkin merupakan struktur data terpisah atau tidak. Jika informasi peminjaman perlu disimpan selain pengembalian buku, maka sebaiknya informasi tersebut dibuat kelas/struktur data yang terpisah. Jika tidak, ID pelindung dapat menjadi bagian dari kelas salinan/struktur data beserta tanggal jatuh tempo buku.

Perhatikan pada Gambar 9-2 bahwa banyak item data telah ditambahkan yang lebih banyak berada di ruang implementasi/solusi dibandingkan di ruang masalah/domain. Dapat dikatakan bahwa “ISBN” adalah bagian dari ruang masalah dan bukan ruang solusi, namun banyak sistem perpustakaan tidak mengizinkan pengguna normal untuk mencari berdasarkan ISBN.



Gambar 9-1. Model objek untuk perpustakaan.



Gambar 9-2. Diagram kelas untuk masalah perpustakaan.

Antarmuka

Spesifikasi antarmuka adalah spesifikasi perilaku eksternal suatu modul. Ini harus cukup lengkap sehingga modul pemanggil mengetahui secara pasti apa yang akan dilakukan modul yang dipanggil dalam keadaan apa pun. Informasi tersebut juga harus cukup lengkap sehingga pelaksana mengetahui secara pasti informasi apa yang harus diberikan.

Spesifikasi antarmuka dalam model OO seringkali merupakan tanda tangan dari metode publik dan semantik yang terkait dengan metode tersebut. Antarmuka juga dapat ditentukan sebagai bagian dari spesifikasi formal dari perilaku keseluruhan sistem.
Antarmuka juga dapat berupa invarian, prakondisi, dan pascakondisi untuk suatu metode.

Contoh 9.4

Rancang antarmuka untuk fungsi peminjaman dari masalah perpustakaan menggunakan diagram kelas yang dihasilkan pada Contoh 9.3. Baik patron maupun penyalin buku mempunyai metode ''meminjam''. Agaknya, memanggil salah satu dari dua metode ini menciptakan contoh pinjaman. Tidak jelas dari diagram kelas metode mana yang membuat instance tersebut. Namun, mungkin akan jelas jika parameter dan tipe kembalian dari masing-masing metode ini ditentukan.

```
method patron::borrow
    input parameters - isbn
```

```
    return type - int
        0 if book is not available
        1 if book is available and loan instance created successfully
        -1 if error condition
method bookcopy: : borrow
    input parameter - loannumber
    return type - int
        0 if bookcopy is not available
        1 if bookcopy updated successfully
```

9.3 KONSEP DESAIN

Dua pendekatan terhadap desain dikenal sebagai penyempurnaan dan modularitas:

Penyempurnaan—Pendekatan desain ini mengembangkan desain dengan menyempurnakan tingkat detail secara berturut-turut. Kadang-kadang ini disebut desain "top-down".

Modularitas—Ini adalah pendekatan penataan yang membagi perangkat lunak menjadi bagian-bagian yang lebih kecil. Semua bagian dapat diintegrasikan untuk mencapai persyaratan masalah.

Contoh 9.5

Sempurnakan fungsi pinjam buku dari masalah perpustakaan. Tingkat atas dimulai dengan fungsi pinjam buku dengan dua parameter yaitu judul buku dan nama patron.

Penyempurnaan selanjutnya menambahkan pengertian entitas pinjaman. Ini mungkin memiliki bagian-bagian berikut: menemukan buku berdasarkan judul buku, menemukan patron berdasarkan nama patron, dan membuat instance peminjaman berdasarkan IDS buku dan patron.

Penyempurnaan selanjutnya memperluas setiap bagian. Temukan buku, kembalikan ISBN jika buku ditemukan dan tersedia, kembalikan nol jika buku tidak ditemukan, dan kembalikan –1 jika buku sedang digunakan. Temukan patron, kembalikan ID patron jika patron ditemukan dan bereputasi baik, kembalikan nol jika patron tidak ditemukan, dan kembalikan –1 jika patron tidak memenuhi syarat untuk meminjam buku. Buat pengembalian pinjaman 1 jika berhasil dibuat.

Atribut Desain

Tiga atribut desain adalah sebagai berikut:

Abstraksi—Sebuah objek dikatakan abstrak jika detail yang tidak perlu dihilangkan. Demikian pula, abstraksi dalam seni mencoba menyampaikan suatu gambar hanya dengan sedikit detail.

Abstraksi dalam desain perangkat lunak mencoba membiarkan perancang fokus pada isu-isu penting tanpa memperhatikan detail tingkat rendah yang tidak perlu. Abstraksi yang baik menyembunyikan detail yang tidak perlu.

Kohesi—Suatu bahan dikatakan kohesif jika bahan-bahan tersebut saling menempel. Suatu prosedur disebut kohesif jika semua pernyataan dalam prosedur tersebut berkaitan dengan setiap keluaran. Suatu kelas disebut kohesif jika semua atribut dalam kelas tersebut digunakan oleh setiap metode. Artinya, keterpaduan dalam suatu modul

tercapai bila segala sesuatunya saling berkaitan. Kohesi yang tinggi umumnya dianggap diinginkan.

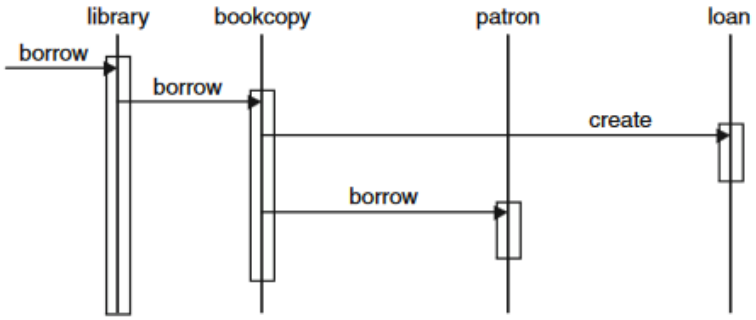
Awalnya, kohesi didefinisikan dalam bentuk jenis-jenis kohesi. Jenis-jenisnya mencakup kebetulan, logis, temporal, prosedural, komunikasi, sekuensial, dan fungsional. Kohesi temporal adalah ketika semua fungsi dikelompokkan bersama karena harus dilakukan pada waktu yang sama. Kohesi logis adalah ketika fungsi-fungsi tersebut secara logis dimiliki bersama.

Kopling—Kopling adalah ukuran keterhubungan modul-modul. Dua modul digabungkan jika perubahan pada variabel di satu modul mungkin memerlukan perubahan di modul lainnya. Biasanya kopling terendah yang diinginkan.

Contoh 9.6

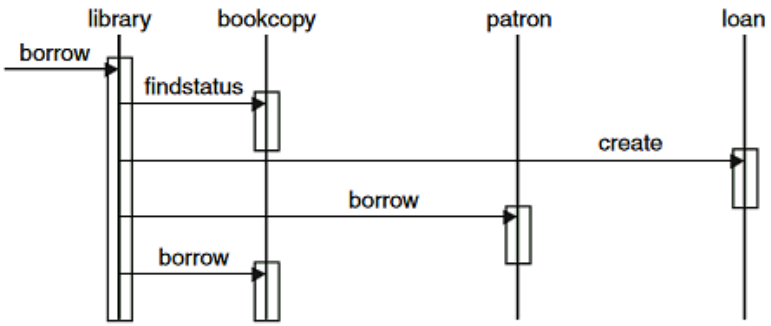
Evaluasi abstraksi dalam fungsionalitas peminjaman dalam masalah perpustakaan. Fungsi `borrow` muncul dalam tiga kelas: `library`, `patron`, dan `bookcopy`. Abstraksi terbaik adalah jika fungsi `borrow` di perpustakaan mengetahui sesedikit mungkin detail tentang fungsi `patron` dan `bookcopy`. Misalnya, apakah fungsi peminjaman perlu mengetahui tentang kelas `loan`?

Seperti ditunjukkan pada Gambar 9-3, jika fungsi `borrow` di `library` hanya memanggil fungsi `borrow` di salah satu level yang lebih rendah, maka fungsi tersebut memiliki abstraksi yang baik. Kelas yang lebih rendah tersebut akan menangani detail pembuatan instance `loan` dan meneruskan pointer ke kelas tingkat yang lebih rendah lainnya.



Gambar 9-3 DFD peminjaman buku

Namun, jika fungsi `borrow` di `library` mengetahui tentang kelas `loan`, ia dapat memeriksa ketersediaan `book`, membuat instance `borrow`, dan memanggil kedua fungsi `borrow` tingkat rendah untuk menetapkan nilai instance `loan`. (Lihat Gambar 9-4.)



Gambar 9-5. Pinjam diagram interaksi—versi 1.

9.4 MENGUKUR KOHESI

Irisan PROGRAM

Nilai variabel dalam suatu program bergantung pada nilai variabel lainnya. Ada dua ketergantungan dasar: ketergantungan data, dimana nilai x mempengaruhi nilai y melalui pasangan definisi dan penggunaan, dan ketergantungan kontrol, dimana nilai x menentukan apakah kode yang berisi definisi y dieksekusi.

Contoh 9.7 Perkalikan Dengan Penambahan Berulang

Kode berikut menghitung produk dari x dan y. Variabel keluaran z memiliki ketergantungan data pada variabel x, karena x ditambahkan ke z. Output z memiliki ketergantungan kontrol pada variabel y, karena y mengontrol berapa kali x ditambahkan ke z.

```
z = 0;
while x > 0 do
     z = z + y;
     x = x-1;
end-while
```

Irisan program dapat dihitung dari kedua arah. Irisan keluaran menemukan setiap pernyataan yang mempengaruhi nilai keluaran yang ditentukan. Irisan masukan menemukan setiap pernyataan yang dipengaruhi oleh nilai masukan yang ditentukan.

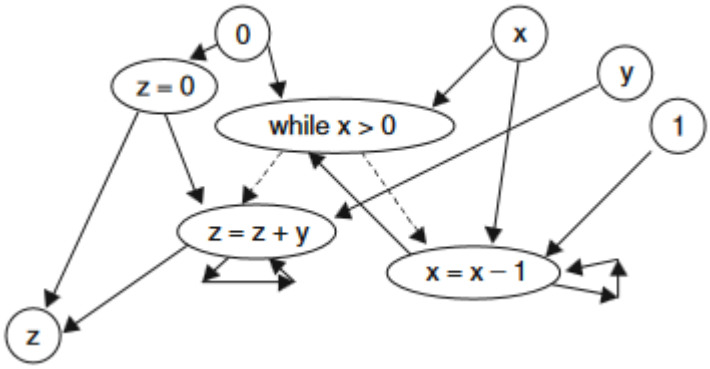
Lebih mudah untuk menghitung irisan program dari grafik berarah yang memiliki sekumpulan node, n, di mana setiap node merupakan masukan, keluaran, atau pernyataan dalam kode. Busur, e, adalah dependensi.

James Bieman dan Linda Ott telah menggunakan definisi variabel dan referensi sebagai unit dasar, bukan pernyataan program. Definisi dan referensi ini disebut token. Jadi, setiap referensi konstan, referensi variabel, dan definisi variabel adalah token terpisah.

Contoh 9.8

Gambarlah grafik berarah yang menunjukkan ketergantungan antar variabel dalam kode pada Contoh 9.7. Gunakan garis padat untuk ketergantungan data dan garis putus-putus untuk ketergantungan kontrol.

Dari grafik pada Gambar 9-6, kita dapat melihat bahwa potongan keluaran akan dimulai dari satu-satunya keluaran, z. Token z, z, y, z, dan 0 dari pernyataan $z = z + y$ dan $z = 0$ ditambahkan ke irisan. Selanjutnya token x dan 0 ditambahkan dari pernyataan sedangkan $x > 0$. Selanjutnya token x , x , dan 1 dari pernyataan $x = x + 1$ ditambahkan. Ini menghabiskan pernyataan, jadi semua yang ada di program ini ada di potongan keluaran untuk variabel z.



Gambar 9-6 persamaan rumus *z*

Irisan masukan dapat dimulai dengan variabel masukan x. Token x , $0, x$, x, dan 1 dari pernyataan sementara $x > 0$ dan $x = x - 1$ ditambahkan ke irisan. Selanjutnya, token z, z, dan y dari pernyataan $z = z + y$ ditambahkan. Tidak ada token lain yang dapat ditambahkan. Jadi, irisan masukan adalah segalanya kecuali $z = 0$.

Irisan masukan untuk variabel y hanya akan berisi token y awal dan token z dan y dari pernyataan $z = z + y$.

Token LEM

Bieman dan Ott juga mendefinisikan beberapa metrik kohesi menggunakan potongan keluaran. Definisinya didasarkan pada token lem, yaitu token (bagian kode) yang ada di lebih dari satu irisan, dan token lem super, yang ada di semua irisan. Kelengketan suatu token adalah persentase potongan keluaran dalam suatu prosedur yang berisi token tersebut.

Ada tiga ukuran kohesi fungsional:

***Kohesi fungsional lemah (WFC)*—**Rasio token lem terhadap total token

Kohesi fungsional kuat (SFC)—Rasio token superglue terhadap total toke

***Daya rekat (A)*—**Rata-rata daya rekat semua token

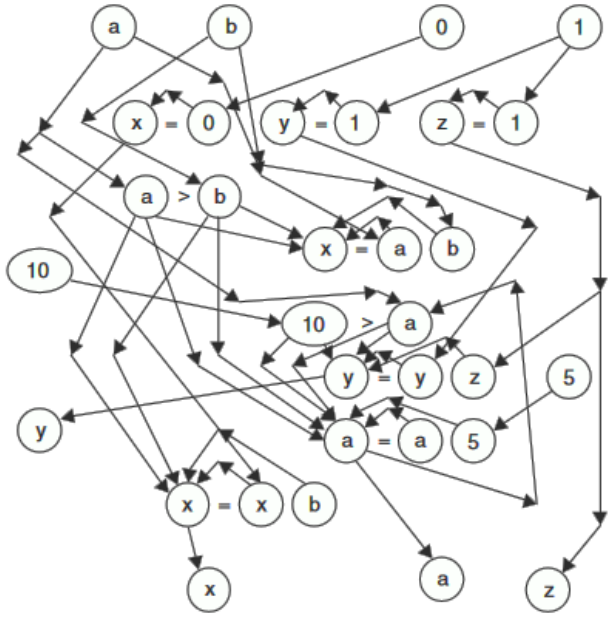
CONTOH 9.9

Hitung ukuran kohesi fungsional untuk fragmen kode berikut.

```
cin >> a >> b;
int x,y, z;
x=0; y=1; z=1;
if (a >b) {
     x = a*b;
     while (10 > a) {
          y=y+z;
          a=a+5;
```

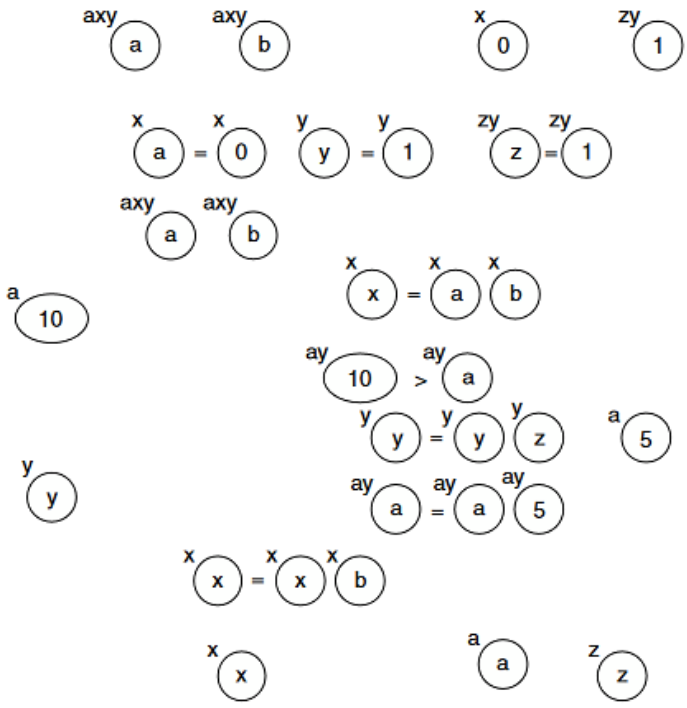
```
}
else {
     x=x+b;
)
```

Gambar 9-7 menunjukkan setiap token dari kode. Busur diambil dari setiap token ke semua token yang langsung dipengaruhi oleh nilai token tersebut.



Gambar 9-7. Grafik berarah menunjukkan semua dependensi.

Lihat Gambar 9-8.



Gambar 9-8. Token beranotasi menunjukkan irisan tempat token tersebut muncul.

Tidak ada token lem super, sehingga kohesi fungsional kuat (SFC) sama dengan nol. Dari 31 token terdapat 12 token lem, sehingga kohesi fungsional lemahnya adalah 12/31 atau 0,387.

Ada empat potong. Nol token memiliki daya rekat 100 persen. Empat token ada dalam tiga irisan, sehingga memiliki daya rekat 75 persen. Delapan token ada dalam dua irisan, sehingga memiliki daya rekat 50 persen. Sisanya, 19, hanya dalam satu irisan, sehingga memiliki daya rekat 25 persen.

Daya rekat adalah daya rekat rata-rata semua token, jadi

(4*0.75+8*0.50+ 19*0.25)/31 = 11.25/31 = 0.363.

9.5 MENGUKUR KOPLING

Kopling adalah ukuran seberapa erat ikatan dua atau lebih modul atau kelas. Secara khusus, metrik penggandengan harus menunjukkan seberapa besar kemungkinan perubahan pada modul lain akan mempengaruhi modul ini. Banyak metrik penggandengan telah diusulkan. Bentuk dasar metrik penggandengan adalah menetapkan daftar item yang menyebabkan satu modul dikaitkan dengan cara kerja internal modul lain.

Kopling Modul Dharma

Dharma mengusulkan metrik dengan daftar situasi berikut untuk dihitung:

di = Jumlah parameter data masukan

ci = Jumlah parameter kontrol masukan

do = Jumlah parameter data keluaran

co = Jumlah parameter kontrol keluaran

gd = Jumlah variabel global yang digunakan sebagai data

gc = Jumlah variabel global yang digunakan sebagai kontrol

w = Jumlah modul yang dipanggil (fan-out)

r = Jumlah modul yang memanggil modul ini (fan-in)

Indikator penggandengan modul Dharma adalah kebalikan dari jumlah item sebelumnya dikalikan konstanta proporsionalitas:

$$mc = k/(di + 2 * ci + do + 2 * co + gd + 2 * gc + w + r)$$

Ada dua kesulitan dengan metrik ini. Salah satunya adalah invers yang berarti semakin besar jumlah situasi yang dihitung, semakin besar penggabungan modul ini dengan modul lain dan semakin kecil nilai mc. Permasalahan lainnya adalah parameter dan jumlah pemanggilan menawarkan potensi masalah namun tidak menjamin bahwa modul ini terhubung dengan cara kerja modul lain. Penggunaan variabel global hampir menjamin bahwa modul ini terikat dengan modul lain yang mengakses variabel global yang sama.

9.6 PERSYARATAN PENELUSURAN

Keterlacakan persyaratan mencoba menghubungkan setiap persyaratan dengan elemen desain yang memenuhi persyaratan. Persyaratan harus mempengaruhi desain. Jika

suatu persyaratan tidak memiliki bagian desain yang sesuai atau bagian desain tidak memiliki bagian yang sesuai dalam persyaratan, maka terdapat potensi masalah. Tentu saja, beberapa persyaratan tidak memiliki bagian spesifik dari desain yang mencerminkan persyaratan tersebut, dan beberapa bagian dari desain mungkin sangat umum sehingga tidak ada bagian dari persyaratan yang memerlukan bagian tersebut.

Salah satu pendekatan untuk memeriksa ketertelusuran adalah dengan menggambar matriks. Di satu sumbu akan dicantumkan semua item persyaratan, dan di sumbu lainnya akan ada daftar item desain. Tanda akan ditempatkan di persimpangan ketika item desain memenuhi suatu persyaratan.

Contoh 9.10

Gambarlah sebuah matriks yang menunjukkan penelusuran persyaratan dalam deskripsi masalah B&B dan desainnya berikut ini.

Persyaratan:

> *Tom dan Sue memulai penginapan dan sarapan di kota kecil di New England. [1] Mereka akan memiliki tiga kamar tidur untuk tamu. [2] Mereka menginginkan sistem untuk mengelola [2.1] reservasi dan memantau [2.2] pengeluaran dan keuntungan. Ketika calon pelanggan menelepon untuk reservasi [3], mereka akan memeriksa kalender [4], dan jika ada lowongan [5], mereka akan memasukkan [6.1] nama pelanggan, [6.2] alamat, [6.3] telepon nomor, [6.4] tanggal, [6.5] harga yang disepakati, [6.6] nomor kartu kredit, dan [6.7] nomor kamar. Reservasi harus [7] dijamin dengan [7.1] pembayaran 1 hari. Reservasi akan ditahan tanpa jaminan untuk [7.2] waktu yang disepakati. Jika tidak dijamin pada tanggal tersebut, reservasi akan dibatalkan.*

Desain:

```
Class [A] B&B attributes: [A.1] day* daylist [DAYMAX] ;
     [A.2] reservation* reslist [MAX]; [A.3] transaction*
     translist [TRANSMAX]
     methods: [A.4] display calendar by week
          [A.5] display reservations by customer
          [A.6] display calendar by month
Class [B] day attributes: [B.1] date thisdate
     [B.2] reservation* rooms [NUMBEROFROOMS]
     methods: [B.3] create(), [B.4] addreservation(),
     [B.5] deletereservation ()
Class [C] reservation attributes: [C.1]string name
     [C.2] string address [C.3] string creditcardnumber
     [C.4] date arrival [C.5] date guaranteeby
     [C.6] int numberofdays [C.7]int roomnumber
     methods: [C.8] create() [C.9] guarantee ()
     [C.10] delete()
Class [D] transaction attributes: [D.1] string name
     [D.2] date postingdate [D.3] float amount
     [D.4] string comments
```

	A	A.1	A.2	A.3	A.4	A.5	A.6	B	B.1	B.2	B.3	B.4	B.5
1										X			
2	X												
2.1			X							X	X	X	X
2.2				X									
3			X										
4					X	X	X						
5													
6.1													
6.2													
6.3													
6.4													
6.5													
6.6													
6.7													
7													
7.1													
7.2													
7.3													

	C	C.1	C.2	C.3	C.4	C.5	C.6	C.7	C.8	C.9	C.10	D	D.1	D.2	D.3	D.4
1																
2																
2.1	X															
2.2												X				
3																
4																
5		X														
6.1		X														
6.2																
6.3																
6.4					X		X									
6.5																
6.6																
6.7									X							
7											X					
7.1																
7.2							X									
7.3												X				

Seperti terlihat pada tabel di atas, terdapat sejumlah baris kosong dan kolom kosong. Persyaratan 5 terkait dengan lowongan. Tidak ada penanganan yang tegas terhadap lowongan, padahal yang dimaksud dengan lowongan adalah tidak adanya reservasi pada tanggal tertentu. Persyaratan 6.3 adalah nomor telepon pelanggan, dan tidak ada. Persyaratan 6.5 adalah harga yang disepakati, yang tidak ada dalam informasi reservasi. Persyaratan 7.1 menyebutkan pembayaran 1 hari, yang juga tidak ada dalam atribut.

Kolom A.1 merupakan daftar harian yang disertakan untuk membantu pencarian lowongan. B dan B.1 diperlukan namun tidak spesifik untuk suatu persyaratan. C.8 adalah konstruktor. D.1 sampai dengan D.4 merupakan rincian transaksi yang diabaikan dalam persyaratan.

LATIHAN SOAL

1. Dengan adanya desain berikut, tunjukkan ide penggandengan, kohesi, dan abstraksi berikut apakah diinginkan memiliki nilai tinggi atau rendah dan bagaimana desain ini mencontohkan istilah tersebut.

```
Class college
     student* stulist [MAX]
     course* courselist [MAX]
public:
     addStudent (char* studentname)
     addStudentToCourse (char* studentname, char*
     coursename)
     void displayStudent (char* studentname)
     void displayStudentsInCourse (char* coursename)
Class course
     student* classroll [MAX]
public:
     void displayStudents ()
Class student
     char* name
public:
     void displayname ()
```

2. Mengapa Gunter membatasi istilah/kejadian yang dapat digunakan dalam suatu spesifikasi? Apa perbedaan antara kebutuhan pengguna dan spesifikasi?
3. Sistem yang diusulkan adalah pengenalan wajah berbasis pengolahan citra. Sistem tersebut akan memiliki kamera dan dimaksudkan untuk mencegah non-karyawan memasuki fasilitas rahasia perusahaan dengan mengontrol kunci pintu. Ketika seseorang mencoba memutar pegangan pintu, sistem mengambil gambar dan membandingkannya dengan sekumpulan gambar karyawan saat ini.
   Klasifikasikan masing-masing kejadian berikut, apakah kejadian tersebut berada di lingkungan atau di dalam sistem dan apakah kejadian tersebut tersembunyi atau terlihat:
   1. Seseorang mencoba memutar gagang pintu.
   2. Pintu tidak terkunci oleh sistem.
   3. Seorang karyawan membiarkan seorang non-karyawan melewati pintu.
   4. Seorang karyawan mempunyai saudara kembar identik.
   5. Suatu gambar memiliki jumlah kesamaan minimal untuk algoritma pencocokan.

Latihan Soal Tambahan (Diskusi)

1. Gambarkan skenario interaksi antara pelanggan yang mencoba membeli CD musik tertentu dengan uang tunai dan petugas di toko musik. Pastikan untuk mencakup semua kemungkinan. Gunakan model mesin negara dengan kejadian sebagai busur.
2. Hitung metrik kohesi fungsional Bieman dan Ott untuk segmen kode berikut. Gambarlah grafik berarah dan tunjukkan alirannya.

```
cin >> a >> b;
int x,y,z;
x=0; y=1; z=1;
while (a > 0) {
    x=x+b;
    z=z*b;
    if (a>b) {
        y=y*a;
    }
    a=a-1;
}
cout << x << a << z << y;
}
```

BAB 10
PENGUJIAN PERANGKAT LUNAK

10.1 PENDAHULUAN

Pengujian perangkat lunak adalah pelaksanaan perangkat lunak dengan data pengujian sebenarnya. Kadang-kadang disebut pengujian perangkat lunak dinamis untuk membedakannya dari analisis statis, yang kadang-kadang disebut pengujian statis. Analisis statis melibatkan analisis kode sumber untuk mengidentifikasi masalah. Meskipun teknik lain sangat berguna dalam memvalidasi perangkat lunak, eksekusi perangkat lunak yang sebenarnya dengan data pengujian yang nyata sangatlah penting.

10.2 DASAR-DASAR PENGUJIAN PERANGKAT LUNAK

Pengujian menyeluruh adalah pelaksanaan setiap kasus uji yang mungkin. Jarang sekali kita bisa melakukan pengujian menyeluruh. Bahkan sistem yang sederhana pun mempunyai terlalu banyak kemungkinan kasus uji. Misalnya, sebuah program dengan dua masukan bilangan bulat pada mesin dengan kata 32-bit akan memiliki 264 kemungkinan kasus uji (lihat Pertanyaan Tinjauan 10.1). Oleh karena itu, pengujian selalu mengeksekusi persentase yang sangat kecil dari kemungkinan kasus pengujian.

Dua hal mendasar dalam pengujian perangkat lunak adalah (1) kasus uji apa yang akan digunakan (pemilihan kasus uji) dan (2) berapa banyak kasus uji yang diperlukan (kriteria penghentian). Pemilihan kasus uji dapat didasarkan pada spesifikasi (fungsional), struktur kode (struktural), aliran data (data flow), atau pemilihan kasus uji secara acak. Pemilihan kasus uji dapat dilihat sebagai upaya untuk menempatkan kasus uji di seluruh ruang masukan. Beberapa area dalam domain mungkin sangat rawan kesalahan dan mungkin memerlukan perhatian ekstra. Kriteria penghentian dapat didasarkan pada kriteria cakupan, seperti mengeksekusi n kasus uji di setiap subdomain, atau kriteria penghentian dapat didasarkan pada kriteria perilaku, seperti pengujian hingga tingkat kesalahan kurang dari ambang batas x.

Suatu program dapat dianggap sebagai pemetaan dari ruang domain ke ruang atau rentang jawaban. Diberikan sebuah masukan, yang merupakan sebuah titik dalam ruang domain, program menghasilkan keluaran, yang merupakan sebuah titik dalam rentang tersebut. Demikian pula spesifikasi programnya adalah peta dari ruang domain ke ruang jawaban.

Spesifikasi penting untuk pengujian perangkat lunak. Kebenaran dalam perangkat lunak didefinisikan sebagai pemetaan program yang sama dengan pemetaan spesifikasi. Pepatah yang baik untuk diingat adalah ''sebuah program tanpa spesifikasi selalu benar.'' Sebuah program tanpa spesifikasi tidak dapat diuji terhadap suatu spesifikasi, dan program melakukan apa yang dilakukannya dan tidak melanggar spesifikasinya.

Sebuah kasus uji harus selalu menyertakan keluaran yang diharapkan. Terlalu mudah untuk melihat keluaran dari komputer dan menganggapnya benar. Jika keluaran yang

diharapkan berbeda dengan keluaran sebenarnya, maka penguji dan/atau pengguna dapat memutuskan mana yang benar.

10.3 KRITERIA CAKUPAN TES

Kriteria cakupan tes adalah aturan tentang cara memilih tes dan kapan harus menghentikan tes. Salah satu permasalahan mendasar dalam penelitian pengujian adalah bagaimana membandingkan efektivitas kriteria cakupan tes yang berbeda. Pendekatan standarnya adalah dengan menggunakan hubungan subsum.

Subsume

Kriteria pengujian A menggolongkan kriteria cakupan pengujian B jika ada set pengujian yang memenuhi kriteria A juga memenuhi kriteria B. Ini berarti bahwa kriteria cakupan pengujian A mencakup kriteria B. Misalnya, jika satu kriteria cakupan pengujian mengharuskan setiap pernyataan untuk dieksekusi dan kriteria lain mengharuskan setiap pernyataan dieksekusi dan beberapa pengujian tambahan, maka kriteria kedua akan menggabungkan kriteria pertama.

Para peneliti telah mengidentifikasi hubungan subsum di antara sebagian besar kriteria konvensional. Namun, meskipun subsum adalah karakteristik yang digunakan untuk membandingkan kriteria tes, subsum tidak mengukur efektivitas relatif dari dua kriteria. Hal ini karena sebagian besar kriteria menentukan bagaimana serangkaian kasus uji akan dipilih. Memilih kumpulan kasus uji yang minimal untuk memenuhi suatu kriteria tidaklah seefektif memilih kasus uji yang baik sampai kriteria tersebut terpenuhi. Oleh karena itu, rangkaian kasus uji yang baik dan memenuhi kriteria yang “lebih lemah” mungkin jauh lebih baik daripada rangkaian kasus uji yang dipilih dengan buruk namun memenuhi kriteria yang “lebih kuat”.

Pengujian Fungsional

Dalam pengujian fungsional, spesifikasi perangkat lunak digunakan untuk mengidentifikasi subdomain yang harus diuji. Salah satu langkah pertama adalah membuat kasus uji untuk setiap jenis keluaran program yang berbeda. Misalnya, setiap pesan kesalahan harus dibuat. Selanjutnya, semua kasus khusus harus memiliki test case. Situasi sulit harus diuji. Kesalahan umum dan kesalahpahaman harus diuji. Hasilnya harus berupa serangkaian kasus uji yang akan menguji program secara menyeluruh ketika diimplementasikan. Serangkaian kasus uji ini juga dapat membantu memperjelas kepada pengembang beberapa perilaku yang diharapkan dari perangkat lunak yang diusulkan.

Dalam buku klasiknya, Glenford Myers mengajukan masalah pengujian fungsional berikut: Kembangkan serangkaian kasus uji yang baik untuk program yang menerima tiga angka, a, b, dan c, interpretasikan angka-angka tersebut sebagai panjang sisi sebuah segitiga, dan menampilkan jenis segitiga. Myers melaporkan bahwa berdasarkan pengalamannya, sebagian besar pengembang perangkat lunak tidak akan merespons dengan rangkaian pengujian yang baik. Saya menemukan pengalaman yang sama dalam menggunakan contoh ini di kelas rekayasa perangkat lunak. Beberapa kelas bahkan gagal memasukkan segitiga valid ke dalam set tes.

CONTOH 10.1

Untuk soal segitiga klasik ini, kita dapat membagi ruang domain menjadi tiga subdomain, satu untuk setiap jenis segitiga berbeda yang akan kita pertimbangkan: sisi tak sama panjang (tidak ada sisi yang sama), sama kaki (dua sisi sama panjang), dan sama sisi (semua sisi sama besar). Kami juga dapat mengidentifikasi dua situasi kesalahan: subdomain dengan masukan yang buruk dan subdomain yang sisi-sisinya tidak membentuk segitiga. Selain itu, karena urutan sisinya tidak ditentukan, semua kombinasi harus dicoba.
Terakhir, setiap kasus uji perlu menentukan nilai keluarannya.

Subdomain	***Example Test Case***
Scalene:	
Increasing size	(3,4,5 scalene)
Decreasing size	(5,4,3 scalene)
Largest as second	(4,5,3 scalene)
Isosceles:	
a =b & other side larger	(5,5,8-isosceles)
a=c & other side larger	(5,8,5 isosceles)
b=c & other side larger	(8,5,5 isosceles)
a=b & other side smaller	(8,8,5-isosceles)
a=c & other side smaller	(8,5,8 isosceles)
b=c & other side smaller	(5,8,8-isosceles)
Equilateral:	
All sides equal	(5,5,5 equilateral)
Not a triangle:	
Largest first	(6,4,2 not a triangle)
Largest second	(4,6,2-not a triangle)
Largest third	(1,2,3 not a triangle)
Bad inputs:	
One bad input	(-1,2,4 bad inputs)
Two bad inputs	(3,-2,-5 bad inputs)
Three bad inputs	(0,0,0 - bad inputs)

Daftar subdomain ini dapat ditambah untuk membedakan subdomain lain yang mungkin dianggap signifikan. Misalnya, dalam subdomain tak sama panjang, sebenarnya ada enam urutan berbeda, namun penempatan yang terbesar mungkin yang paling signifikan berdasarkan kemungkinan kesalahan dalam pemrograman. Perhatikan bahwa satu kasus uji di setiap subdomain biasanya dianggap minimal namun dapat diterima.

Matrik Uji

Cara untuk memformalkan identifikasi subdomain ini adalah dengan membangun matriks menggunakan kondisi yang dapat kita identifikasi dari spesifikasi dan kemudian secara sistematis mengidentifikasi semua kombinasi kondisi ini sebagai benar atau salah.

Contoh 10.2

Syarat pada soal segitiga tersebut adalah $(1)\ a = b$ atau $a = c$ atau $b = c, (2)\ a = b$ dan $b = c, (3)\ a < b + c$ dan $b < a + c$ dan $c < a + b$, dan $(4)\ a > 0$ dan $b > 0$ dan $c > 0$. Keempat kondisi tersebut dapat diletakkan pada baris-baris suatu matriks. Kolom matriks masing-masing akan menjadi subdomain. Untuk setiap subdomain, tanda T akan ditempatkan di setiap baris yang kondisinya benar dan F jika kondisinya salah. Semua kombinasi T dan F yang valid akan digunakan. Jika ada tiga kondisi, mungkin ada 23 = 8 subdomain (kolom). Baris tambahan akan digunakan untuk nilai a, b, dan c dan untuk keluaran yang diharapkan untuk setiap subdomain.

Kondisi	1	2	3	4	5	6	7	8
$a = b$ atau $a = c$ atau $b = c$	*T*	*T*	*T*	*T*	*T*	*F*	*F*	*F*
$a = b$ atau $b = c$	*T*	*T*	*F*	*F*	*F*	*F*	*F*	*F*
$a \leq b + c$ atau $b \leq a + c$ atau $c \leq a + b$	*T*	*F*	*T*	*T*	*F*	*T*	*T*	*F*
$a \geq 0$ atau $b \geq 0$ atau $c \geq 0$	*T*	*F*	*T*	*F*	*F*	*T*	*F*	*F*
Kasus Tes Sederhana	0,0,0	3,3,3	0,4,0	3,8,3	5,8,5	0,5,6	3,4,8	3,4,5
Output yang diharapkan	Bad Input	Equateral	Bad Input	Not Triangle	Isosceles	Bad Input	Not triangle	Scalene

Pengujian Struktural

Pengujian struktural didasarkan pada struktur kode sumber. Kriteria pengujian struktural yang paling sederhana adalah cakupan setiap pernyataan, sering disebut cakupan C0.

C0—Setiap Cakupan Pernyataan

Kriteria ini adalah bahwa setiap pernyataan kode sumber harus dieksekusi oleh beberapa kasus uji. Pendekatan normal untuk mencapai cakupan C0 adalah dengan memilih kasus uji hingga alat cakupan menunjukkan bahwa semua pernyataan dalam kode telah dieksekusi.

Contoh 10.3

Pseudocode berikut mengimplementasikan masalah segitiga. Matriks menunjukkan baris mana yang dieksekusi oleh kasus uji mana. Perhatikan bahwa tiga pernyataan pertama (A, B, dan C) dapat dianggap sebagai bagian dari node yang sama.

Node	Line Sumber	3,4,5	3,5,3	0,1,0	4,4,4,
A	`read a,b,c`	*	*	*	*
B	`type=''scalene''`	*	*	*	*
C	`if(a == b||b == c||a == c)`	*	*	*	*
D	`type=''isosceles''`		*	*	*
E	`if(a == b&&b == C)`	*	*	*	*
F	`type=''equilateral''`				*
G	`if(a>=b+c||b>=a+c||c>=a+b)`	*	*	*	*
H	`type=''not a triangle''`			*	
I	`if(a <= 0||b <= 0||c <= 0)`	*	*	*	*
J	`type=''bad inputs''`			*	
K	`print type`	*	*	*	*

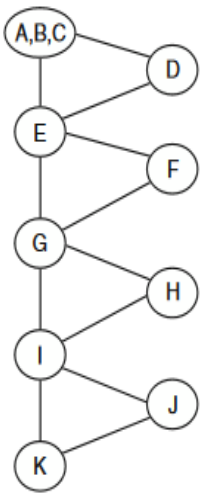
Pada uji kasus keempat, setiap pernyataan telah dieksekusi. Kumpulan kasus uji ini bukanlah kumpulan terkecil yang dapat mencakup setiap pernyataan. Namun, menemukan set pengujian terkecil sering kali tidak menghasilkan set pengujian yang baik.

C1—Pengujian Setiap Cabang

Kriteria pengujian yang lebih menyeluruh adalah pengujian setiap cabang, yang sering disebut cakupan pengujian C1. Dalam kriteria ini, tujuannya adalah untuk mengambil keputusan dua arah.

Contoh 10.4

Jika kita memodelkan program Contoh 10.3 sebagai grafik aliran kendali (lihat Bab 2), kriteria cakupan ini memerlukan cakupan setiap busur dalam diagram aliran kendali. Lihat Gambar 10-1.



Bagian	3,4,5	3,5,3	0,1,0	4,4,4
ABC-D		*	*	*
ABC-E	*			
D-E		*	*	*
E-F				*
E-G	*	*	*	
F-G				*
G-H			*	
G-I	*	*		*
H-I			*	
I-J			*	
I-K	*	*		*
J-K			*	

Gambar 10-1. Grafik aliran kontrol misalnya 10.3.

Pengujian Setiap Jalur

Yang lebih menyeluruh lagi adalah kriteria pengujian setiap jalur. Jalur adalah urutan unik dari node program yang dieksekusi oleh kasus uji. Pada matriks pengujian (Contoh 10.2) di atas terdapat delapan subdomain. Masing-masingnya kebetulan merupakan sebuah jalan. Dalam contoh tersebut, terdapat enam belas kombinasi T dan F yang berbeda. Namun, delapan kombinasi tersebut merupakan jalur yang tidak layak. Artinya, tidak ada kasus uji yang dapat memiliki kombinasi T dan F tersebut untuk keputusan dalam program. Akan sangat sulit untuk menentukan apakah suatu jalur tidak layak atau hanya sulit menemukan kasus uji yang mengeksekusi jalur tersebut.

Kebanyakan program dengan perulangan mempunyai jumlah jalur yang tidak terhingga. Secara umum, pengujian setiap jalur tidak masuk akal.

Contoh 10.5

Tabel berikut menunjukkan delapan jalur yang layak dalam kodesemu segitiga dari Contoh 10.3.

Path	T/F	Test Case	Output
ABCEGIK	FFFF	3,4,5	Scalene
ABCEGHIK	FFTF	3.4,8	Not a triangle
ABCEGHIJK	FFTT	0,5,6	Bad inputs
ABCDEGIK	TFFF	5,8,5	Isosceles
ABCDEGHIK	TFTF	3,8,3	Not a triangle
ABCDEGHIJK	TFTT	0,4,0	Bad inputs
ABCDEFGIK	TTFF	3,3,3	Equilateral
ABCDEFGHIJK	TTTT	0,0,0	Bad inputs

Cakupan Berbagai Kondisi

Kriteria pengujian kondisi ganda mengharuskan setiap kondisi relasi primitif dievaluasi benar dan salah. Selain itu, semua kombinasi T/F untuk relasi primitif dalam suatu kondisi harus dicoba. Perhatikan bahwa evaluasi ekspresi3 yang malas akan menghilangkan beberapa kombinasi. Misalnya, dalam “dan” dari dua relasi primitif, relasi kedua tidak akan dievaluasi jika relasi pertama salah.

Contoh 10.6

Dalam pseudocode pada Contoh 10.3, terdapat beberapa kondisi di setiap pernyataan keputusan. Primitif yang tidak dieksekusi karena evaluasi yang malas ditunjukkan dengan ''X''.

```
if(a==b||b==c||a==c)
```

Kombinasi	Kasus Kemungkinan Uji	Cabang
TXX	3,3,4	ABC-D
FTX	4,3,3	ABC-D
FFT	3,4,3	ABC-D
FFF	3,4,5	ABC-E

```
if(a==b&&b==c)
```

Kombinasi	Kasus Kemungkinan Uji	Cabang
TT	3,3,4	E-F
TF	3,3,4	E-G
FX	4,3,3	E-G

```
if(a>=b+c||b>=a+c||c>=a+b)
```

Kombinasi	Kasus Kemungkinan Uji	Cabang
TXX	8,4,3	G-H
FTX	4,8,3	G-H
FFT	4,3,8	G-H
FFF	3,3,3	G-I

```
if(a<=0||b<=0||c<=0)
```

Kombinasi	Kasus Kemungkinan Uji	Cabang
TXX	0,4,5	I-J
FTX	4,-2,-2	I-J
FFT	5,4,-3	I-J
FFF	3,3,3	I-K

Pengujian Subdomain

Pengujian subdomain adalah gagasan untuk mempartisi domain masukan menjadi subdomain yang saling eksklusif dan memerlukan jumlah kasus pengujian yang sama dari setiap subdomain. Ini pada dasarnya adalah ide di balik matriks pengujian. Pengujian subdomain lebih umum karena tidak membatasi cara pemilihan subdomain. Secara umum, jika ada alasan bagus untuk memilih subdomain, maka subdomain tersebut mungkin berguna untuk pengujian. Selain itu, subdomain dari pendekatan lain mungkin dibagi lagi menjadi subdomain yang lebih kecil. Pekerjaan teoretis telah menunjukkan bahwa pengelompokan subdomain hanya efektif jika cenderung mengisolasi potensi kesalahan ke dalam subdomain individual.

Cakupan setiap pernyataan dan cakupan setiap cabang bukanlah pengujian subdomain. Tidak ada subdomain yang saling eksklusif terkait dengan eksekusi pernyataan atau cabang yang berbeda. Cakupan setiap jalur adalah cakupan subdomain, karena subdomain dari kasus uji yang menjalankan jalur tertentu melalui suatu program saling eksklusif dengan subdomain untuk jalur lainnya.

Contoh 10.7

Untuk soal segitiga, kita mungkin memulai dengan subdomain untuk setiap keluaran. Ini mungkin dibagi lagi menjadi subdomain baru berdasarkan apakah elemen terbesar atau terburuk ada di posisi pertama, posisi kedua, atau posisi ketiga (jika sesuai).

Subdomain	Possible Test Case
Equilateral	3.3,3
Isos - first	8,5,5

Subdomain	Possible Test Case
Not triangle - first	8,3,3
Not triangle - sec	3,8,4

Isos - sec	5.8,5
Isos - third	5,5,8
Scalene - first	5.4,3
Scalene - sec	4,5,3
Scalene - third	3,4,5

Not triangle - third	4,3,8
Bad input - first	0,3,4
Bad input - sec	3,0,4
Bad input - third	3,4,0

C1 Menggabungkan C0

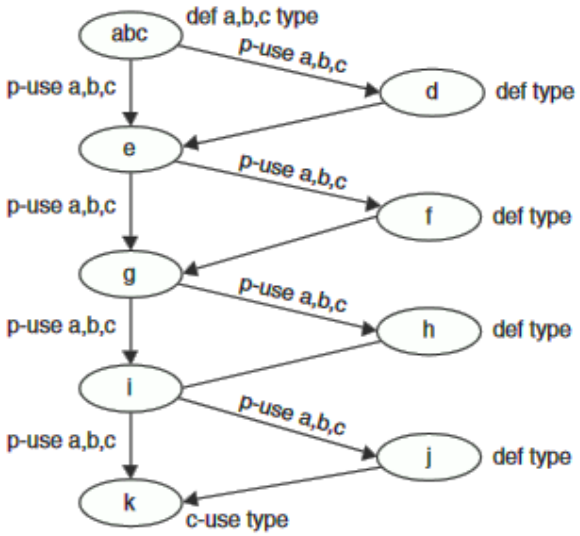
Contoh 10.8—C1 Menggabungkan C0

Untuk soal segitiga, pada Contoh 10.3 kami memilih kasus uji yang baik hingga kami mencapai cakupan C0. Kasus ujinya adalah (3,4,5—scalene), (3,5,3— sama kaki), (0,1,0—input buruk), dan (4,4,4—sama sisi). Pengujian ini juga mencakup empat dari lima kemungkinan keluaran. Namun, kita dapat mencapai cakupan C1 dengan dua kasus uji: (3,4,5—scalene) dan (0,0,0—input buruk). Tes ini mungkin tidak sebaik set tes pertama. Namun mencapai cakupan C1 dan juga mencapai cakupan C0.

10.4 PENGUJIAN ALIRAN DATA

Pengujian aliran data adalah pengujian berdasarkan aliran data melalui suatu program. Data mengalir dari tempat ia didefinisikan ke tempat ia digunakan. Definisi data, atau def, adalah ketika suatu nilai ditetapkan ke suatu variabel. Dua jenis penggunaan yang berbeda telah diidentifikasi. Penggunaan komputasi, atau penggunaan c, adalah ketika variabel muncul di sisi kanan pernyataan penugasan. C-use dikatakan terjadi pada pernyataan penugasan. Penggunaan predikat, atau p-use, adalah ketika variabel digunakan dalam kondisi pernyataan keputusan. Penggunaan p ditugaskan ke kedua cabang dari pernyataan keputusan. Jalur bebas definisi, atau bebas def, adalah jalur dari definisi suatu variabel ke penggunaan variabel tersebut tanpa menyertakan definisi variabel lainnya.

Contoh 10.9—Grafik Aliran Kendali dari Soal Segitiga (Contoh 10.3) Grafik aliran kendali pada Gambar 10-2 dianotasi dengan definisi dan penggunaan variabel a,b, dan c.



Gambar 10-2. Grafik aliran kontrol masalah segitiga.

Ada banyak kriteria pengujian aliran data. Kriteria dasarnya mencakup dcu, yang memerlukan jalur bebas def dari setiap definisi ke penggunaan c; dpu, yang memerlukan jalur bebas def dari setiap definisi ke penggunaan p; dan du, yang memerlukan jalur bebas def dari setiap definisi hingga setiap kemungkinan penggunaan. Kriteria yang paling luas adalah all-du-paths, yang mensyaratkan semua jalur bebas def dari setiap definisi hingga setiap kemungkinan penggunaan.

Contoh 10.10—Pengujian Aliran Data pada Masalah Segitiga

dcu-The only c-use is for variable type in node k (the output statement).

From def type in node abc to node k	Path abc,e,g,i,k
From def type in node d to node k	Path d,e,g,i,k
From def type in node f to node k	Path f,g,i,k
From def type in node h to node k	Path h,i,k
From def type in node j to node k	Path j,k

dpu-The only p-use is for variables a,b,c and the only def of a,b,c is node abc.

From node abc to arc abc-d
From node abc to arc abc-e
From node abc to arc e-f
From node abc to arc e-g
From node abc to arc g-h
From node abc to arc g-i
From node abc to arc i-j
From node abc to arc i-k

du-All defs to all uses.
All test cases of dcu and dpu combined.

all-du-paths-All def-free paths from all defs to all uses.
Same as du tests.

10.5 PENGUJIAN ACAK

Pengujian acak dilakukan dengan memilih kasus uji secara acak. Keuntungan pendekatan ini adalah cepat dan juga menghilangkan bias penguji. Selain itu, inferensi statistik lebih mudah bila tes dipilih secara acak. Seringkali pengujian dipilih secara acak dari profil operasional.

Contoh 10.11

Untuk soal segitiga, kita dapat menggunakan generator bilangan acak dan mengelompokkan setiap rangkaian tiga bilangan yang berurutan sebagai rangkaian pengujian.

Kami akan memiliki pekerjaan tambahan untuk menentukan keluaran yang diharapkan. Salah satu permasalahannya adalah peluang untuk menghasilkan kasus uji sama sisi akan sangat kecil. Jika itu benar-benar terjadi, kami mungkin akan mulai mempertanyakan pembuat nomor pseudorandom kami.

Profil Operasional

Pengujian di lingkungan pengembangan seringkali sangat berbeda dengan eksekusi di lingkungan operasional. Salah satu cara untuk membuat keduanya lebih mirip adalah dengan menentukan spesifikasi tipe dan probabilitas tipe tersebut akan ditemui dalam operasi normal. Spesifikasi ini disebut profil operasional. Dengan menggambar kasus uji dari profil operasional, penguji akan lebih yakin bahwa perilaku program selama pengujian lebih dapat memprediksi perilaku program selama operasi.

Contoh 10.12

Profil operasional yang mungkin untuk soal segitiga adalah sebagai berikut:

#	Description	Probability
1	equilateral	0.20
2	isosceles - obtuse	0.10
3	isosceles - right	0.20
4	scalene - right triangle	0.10
5	scalene - all acute	0.25
6	scalene - obtuse angle	0.15

Untuk menerapkan pengujian acak, penguji mungkin menghasilkan nomor untuk memilih kategori berdasarkan probabilitas dan kemudian nomor tambahan yang cukup untuk membuat kasus uji. Jika kategori yang dipilih adalah kasus sama sisi, penguji akan menggunakan nomor yang sama untuk ketiga masukan. Garis lurus sama kaki memerlukan bilangan acak untuk panjang kedua sisinya, dan kemudian penggunaan trigonometri untuk menghitung sisi lainnya.

Inferensi Statistik Dari Pengujian

Jika pengujian acak dilakukan dengan memilih kasus pengujian secara acak dari profil operasional, maka perilaku perangkat lunak selama pengujian harus sama dengan perilakunya di lingkungan operasional.

Contoh 10.13

Jika kami memilih 1000 kasus uji secara acak menggunakan profil operasional dan menemukan tiga kesalahan, kami dapat memperkirakan bahwa perangkat lunak ini akan memiliki tingkat kesalahan kurang dari tiga kegagalan per 1000 eksekusi di lingkungan operasional. Lihat Bagian 3.8 untuk informasi lebih lanjut tentang penggunaan tingkat kesalahan.

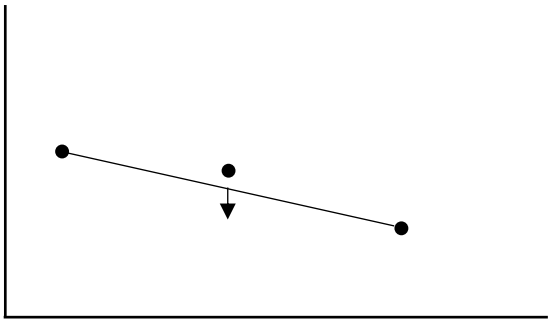
10.6 PENGUJIAN BATAS

Seringkali kesalahan terjadi pada batas antar domain. Dalam kode sumber, pernyataan keputusan menentukan batasan. Jika pernyataan keputusan ditulis sebagai $x < 1$ dan bukan

$x < 0$, batasnya telah bergeser. Jika keputusan ditulis $x =< 1$, maka batas $x = 1$ berada di subdomain sebenarnya. Dalam terminologi pengujian batas, kita mengatakan bahwa pengujian aktif berada dalam domain benar dan pengujian off adalah nilai x lebih besar dari 1 dan berada dalam domain salah.

Jika keputusan ditulis $x < 1$ dan bukannya $x =< 1$, maka batasnya, $x = 1$, kini berada di subdomain palsu dan bukan di subdomain sebenarnya. Pengujian batas bertujuan untuk memastikan bahwa batas sebenarnya antara dua subdomain sedekat mungkin dengan batas yang ditentukan. Dengan demikian, kasus uji dipilih pada batas dan di luar batas sedekat mungkin dengan batas tersebut. Uji batas standar adalah melakukan dua pengujian dengan jarak sejauh mungkin dan satu pengujian di dekat tengah batas.

Gambar 10-3 menunjukkan batas sederhana. Tanda panah menunjukkan bahwa pengujian batas berada pada subdomain di bawah batas. Kedua pengujian on berada di ujung batas dan pengujian off berada tepat di atas batas di tengah-tengah batas.



Gambar 10-3. Kondisi batas.

Contoh 10.14

Pada contoh segitiga, kondisi primitif, $a => b + c$ atau $b => a + c$ atau $c => a + b$, menentukan suatu batas. Karena ketiganya berada dalam tiga variabel, batasnya sebenarnya adalah bidang dalam ruang 3D. Pengujian on akan berupa dua (atau lebih) pengujian yang terpisah jauh dan memiliki persamaan—misalnya, (8,1,7) dan (8,7,1). Kedua hal ini benar. Tes off akan berada di domain lain (salah) dan berada di dekat tengah—misalnya, (7.9, 4,4).
Catatan: Untuk pembahasan pengujian berorientasi objek (OO), lihat Bab 13.

LATIHAN SOAL

1. Apa saja permasalahan mendasar dalam pengujian perangkat lunak?
2. Mengapa diperlukan spesifikasi untuk melakukan pengujian?
3. Mengapa pengujian jalur biasanya tidak praktis?
4. Apakah pengujian jalur mencakup cakupan pernyataan?
5. Penguji perangkat lunak terkadang mengatakan ''kesalahan terjadi di sudut-sudut''. Apa artinya ini?
6. Cakupan setiap pernyataan bukan merupakan kriteria pengujian subdomain. Apa pentingnya hal ini?

7. Apa perbedaan profil operasional terminal tempat penjualan di toko diskon dengan terminal tempat penjualan di toko mewah?
8. Pengembang perangkat lunak mungkin secara tidak sadar tidak menguji perangkat lunaknya secara menyeluruh. Mengapa kriteria cakupan pengujian dapat membantu?

Latihan Soal Tambahan (Diskusi)

1. Jika suatu program memiliki dua masukan bilangan bulat dan masing-masing dapat berupa bilangan bulat 32-bit, berapa banyak kemungkinan masukan yang dimiliki program ini?
2. Jika suatu program mempunyai 264 kemungkinan masukan dan satu pengujian dapat dijalankan setiap milidetik, berapa lama waktu yang diperlukan untuk mengeksekusi semua kemungkinan masukan tersebut?
3. Program penggajian akan menghitung gaji kotor mingguan dengan memasukkan jumlah jam kerja dan upah saat ini. Seorang pekerja tidak boleh bekerja lebih dari 80 jam per minggu, dan upah maksimum adalah Rp. 50.000.000 per jam. Membangun tes fungsional.
4. Sebuah program menghitung luas segitiga. Inputnya adalah tiga set koordinat (x, y). Membangun tes fungsional.
5. Sebuah program menerima dua waktu (dalam format 12 jam) dan mengeluarkan jumlah menit yang telah berlalu. Membangun tes fungsional.
6. Rutinitas pencarian biner mencari daftar nama dalam urutan abjad dan mengembalikan nilai true jika nama tersebut ada dalam daftar dan mengembalikan nilai false jika tidak. Membangun tes fungsional.
7. Untuk program penggajian pada Soal 10.3, kenali kondisinya dan buatlah matriks pengujian.
8. Untuk perhitungan luas segitiga pada Soal 10.4, kenali kondisinya dan buatlah matriks ujinya.
9. Untuk kalkulator waktu berlalu pada Soal 10.5, identifikasi kondisinya dan buatlah matriks pengujian.
10. Untuk pencarian biner rutin pada Soal 10.6, kenali kondisinya dan buatlah matriks uji.
11. Temukan himpunan kasus uji minimal yang dapat mencapai cakupan C0 dan C1 dari kodesemu masalah segitiga dari Contoh 10.3.
12. Pseudocode berikut mengimplementasikan soal waktu berlalu pada Soal 10.5 jika waktu berlalu kurang dari 24 jam. Pilih kasus uji hingga cakupan setiap pernyataan tercapai. Pilih kasus uji tambahan untuk mencapai cakupan setiap cabang.

```
read hr1 min1 AmOrPm1
read hr2 min2 AmOrPm2
if (hr1 == 12)
    hr1=0
if (hr2 == 12)
    hr2 = 0
if (AmOrPm1 == pm)
```

```
        hr1 = hr1 + 12
    if (AmOrPm2 == pm)
        hr2 = hr2 + 12
    if (min2 <min1)
        min2 = min2 + 1
        hr2=hr2-1
    if(hr2 <hr1)
        hr2 = hr2 + 24
    elapsed =min2-min1 + 60* (hr2-hr1)
    print elapsed
```

13. Untuk pseudocode Soal 10.12, temukan himpunan minimal kasus uji yang akan mencapai C0 dan himpunan minimal kasus uji yang akan mencapai C1.
14. Untuk kode berikut, identifikasi semua jalur yang layak, pengujian jalur, dan pengujian aliran data:

```
cin >> a >> b >> c; // node A
x=5; y=7;
if ( a > b && b > c) {
    a = a + 1; // node B
    x = x + 6;
    if (a = 10||b>20) {
        b = b+1; // node C
        x = y+4;
        }
    if (a<10 | | c = 20) { // node D
        b = b + 2; // node E
        y = 4
        }
    a = a + b + 1; // node F
    y = x + y;
    }
if (a>5 || c<10) { // node G
    b = c + 5; // node H
    x=x+1;
    }
cout >> x >> y; // node I
```

15. Dengan kode berikut, gambarkan CFG dan hasilkan satu set kasus uji minimal untuk setiap kriteria berikut: C0, C1, dpu, dan dcu.

```
cin>> a >> b // node A
if (b>a) {
    x= b; // node B
    if (b>20) {
        x= x + 9; // node C
```

```
    }
    else {
        x=x+1; // node D
    }
    x=x+1; // node E
}
else {
    x = a // node F
    if (a> 20_{
        x=x+15; // node G
    }
    x=x-5; // node H
}
if (b>a+20) // node I
{
    x=20; // node J
}
cout << x; // node K
```

BAB 11
PEMBANGUNAN BERORIENTASI OBJEK

11.1 PENDAHULUAN

Perangkat lunak berorientasi objek berbeda dari perangkat lunak konvensional. Ada potensi banyak manfaat untuk pengembangan berorientasi objek. Di antara manfaat ini adalah penyederhanaan persyaratan, desain, dan implementasi. Manfaat ini dicapai dengan memodelkan domain masalah dengan objek yang mewakili entitas penting, dengan merangkum fungsi-fungsi dengan data, dengan menggunakan kembali objek dalam suatu proyek dan antar proyek, dan dengan memiliki solusi yang lebih dekat secara intelektual1 dengan masalah tersebut. Unified Modeling Language (UML) adalah notasi standar untuk model berorientasi objek. Spesifikasi UML tersedia di Web.

Warisan

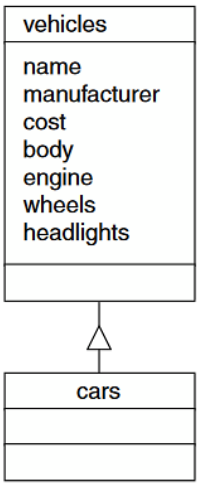
Salah satu ide inovatif dalam perangkat lunak berorientasi objek adalah pewarisan. Pewarisan berasal dari pengenalan hierarki gagasan dan konsep, dan bagaimana hierarki/klasifikasi ini melibatkan banyak penggunaan kembali gagasan yang melekat dan seterusnya dari konsep tingkat yang lebih tinggi ke spesialisasi tingkat yang lebih rendah dari konsep-konsep tersebut.

Ketika dua kelompok entitas dihubungkan dan salah satu entitas menjadi spesialisasi entitas lainnya, terdapat potensi hubungan pewarisan. Dalam hubungan pewarisan, kelas dasar (kelas yang lebih umum) akan berisi semua atribut yang umum. Kelas turunan (kelas yang lebih terspesialisasi) akan mewarisi semua atribut umum dari kelas dasar.

Misalnya, jika satu kelompok entitas terdiri dari kendaraan dan kelompok lainnya terdiri dari mobil, kita dapat menggunakan warisan. Mobil dapat mewarisi dari kendaraan. Mobil dapat berbagi banyak atribut dan pengoperasian kelas kendaraan. Semua atribut umum dapat ditempatkan di kelas dasar. Kelas turunan secara otomatis akan mewarisi atribut tersebut. Berbagi ini menghemat usaha.

Contoh 11.1

Gambarlah model objek yang mengidentifikasi semua kesamaan antara mobil dan kendaraan. Gambar 11-1 menunjukkan bahwa mobil dan kendaraan sama-sama memiliki bodi, mesin, roda (mungkin bukan empat), lampu depan, nama merek, pabrikan, dan harga. (Mungkin masih banyak lagi.)



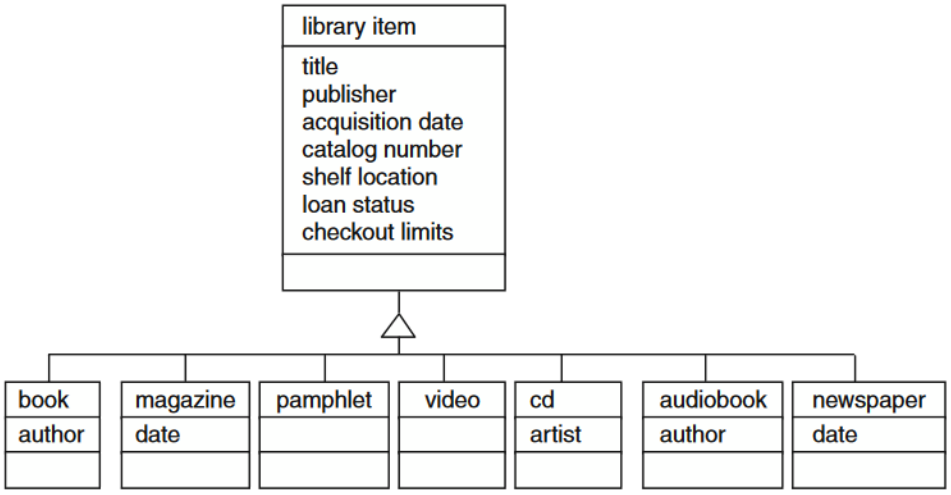
Gambar 11-1 Kueri Mobil

Contoh 11.2

Gambarkan model objek yang mengidentifikasi kesamaan dalam sistem perpustakaan yang menangani buku, majalah, pamflet, kaset video film, CD musik, kaset buku audio, dan surat kabar.

Gambar 11-2 menunjukkan bahwa semua item ini memiliki judul, penerbit, tanggal perolehan, nomor katalog, lokasi rak, status pinjaman, dan batas pembayaran.

Beberapa contoh atribut telah ditambahkan ke beberapa kelas turunan. Perhatikan bahwa buku audio dapat berasal dari buku, dan surat kabar serta majalah dapat berasal dari suatu objek yang disebut serial atau berkala.



Gambar 11-2 Atribut

Polimorfisme

Polimorfisme berarti ''mampu mengambil banyak bentuk.'' Ini mengacu pada fungsi yang dapat menangani versi atau bentuk berbeda dari objek atau daftar parameter. Dalam perangkat lunak berorientasi objek, sering kali berarti fungsi yang dapat menangani tipe dasar

atau tipe turunan. Dalam contoh mobil/kendaraan, kelas dasar akan memiliki fungsi polimorfik untuk tugas-tugas yang dilakukan semua kendaraan namun mungkin dilakukan secara berbeda, seperti berbelok di tikungan. Setiap kelas turunan akan menggunakan fungsi kelas dasar atau menyediakan versi yang sesuai untuk kelas turunan.

Contoh 11.3

Temukan dalam masalah perpustakaan (Contoh 11.2) fungsi-fungsi yang umum untuk semua item dan fungsi-fungsi yang harus dispesialisasikan untuk kelas turunan.

Umum—Fungsi check-out dan check-in (kecuali mungkin untuk batasan checkout).

Khusus—Fungsi katalogisasi akan berbeda.

11.2 MENGIDENTIFIKASI OBJEK

Salah satu pendekatan untuk mengidentifikasi persyaratan sistem yang diusulkan adalah memulai dengan mengidentifikasi objek dalam domain masalah. Objek-objek ini biasanya berupa kata benda dalam rumusan masalah.

Pendekatan Kata Benda Dalam Teks

Dalam pendekatan kata benda dalam teks, semua kata benda dalam teks diidentifikasi. Kata benda yang berbeda dapat digunakan untuk konsep yang sama. Kata benda yang setara dan kata benda yang terkait dengan setiap konsep harus diurutkan ke dalam kelompok. Beberapa kata benda akan berhubungan dengan lingkungan di luar sistem yang diusulkan dan mungkin dihilangkan. Di setiap kelompok, kata benda yang mewakili objek harus dipilih. Kata benda lain dalam grup mungkin menjadi atribut atau dibuang.

Contoh 11.4

Gunakan metode deskripsi kata benda dalam teks untuk mengidentifikasi objek dari masalah toko kelontong berikut:

Sebuah toko kelontong ingin mengotomatiskan inventarisnya. Ia memiliki terminal tempat penjualan yang dapat mencatat semua barang dan jumlah yang dibeli pelanggan. Ini memiliki terminal serupa untuk meja layanan pelanggan untuk menangani pengembalian. Ia memiliki terminal lain di dok pemuatan untuk menangani pengiriman yang datang dari pemasok. Departemen daging dan departemen produksi memiliki terminal untuk memasukkan kerugian/diskon akibat pembusukan.

Kata benda:

toko kelontong, inventaris, terminal tempat penjualan, item, jumlah, pelanggan, pembelian, meja layanan, pengembalian, dok pemuatan, pengiriman, pemasok, departemen daging, departemen produksi, kerugian, diskon.

Grup:

toko kelontong

inventaris, barang, jumlah, pengembalian, kerugian, diskon
pengiriman
pemasok
departemen daging, departemen produksi
pelanggan

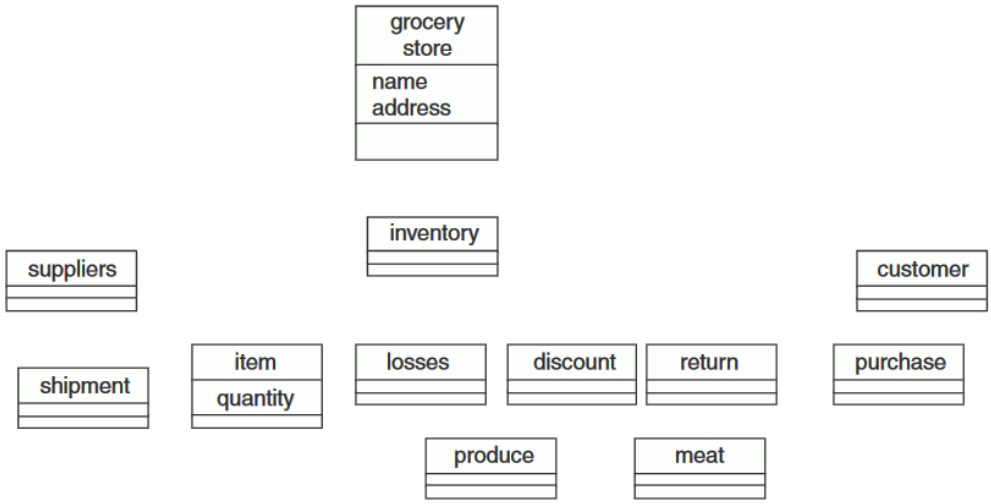
Entitas lingkungan yang berada di luar sistem:

terminal tempat penjualan, meja layanan, dermaga pemuatan, departemen daging, departemen produksi
Namun, item daging dan produk hasil bumi harus dimasukkan untuk mencerminkan proses pengolahan yang berbeda.
Terdapat pilihan apakah pelanggan berada di luar sistem atau sistem mengetahui dan melacak pelanggan.
Keputusan dibuat untuk melacak pelanggan.

Daftar akhir objek dan atribut:

`grocery store`
`inventory`
`items` with an attribute of `quantity`
`customer`
`purchases`
`returns`
`shipments`
`suppliers`
`losses`
`discounts`
`meat items`
`produce items`

Lihat Gambar dibawah ini



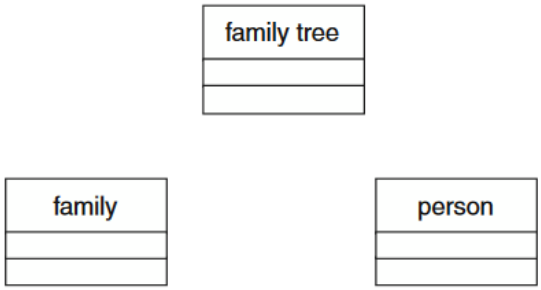
Contoh 11.5

Gunakan metode deskripsi kata benda dalam teks untuk mengidentifikasi objek dari masalah silsilah keluarga berikut:

Fred sedang mempelajari silsilah dan ingin membuat program untuk menyimpan informasi yang dia ketahui tentang keluarganya. Dia berasal dari keluarga besar dan memiliki banyak paman, bibi, dan sepupu.

Tidak mudah menerapkan metode kata benda dalam teks pada contoh ini. Kalimat pertama adalah motivasi dan hanya kata benda ''keluarga'' yang relevan. Kalimat kedua mengulangi kata benda ''keluarga'' dan kemudian mencantumkan kata benda yang merupakan hubungan antar manusia. Berbeda dengan contoh sebelumnya, relasi ini bukanlah kelas turunan. Seorang paman bukanlah suatu spesialisasi seseorang; itu adalah hubungan antar orang.

Keakraban dengan domain masalah akan diperlukan untuk mengidentifikasi objek. Kumpulan objek yang tepat untuk masalah ini adalah silsilah keluarga, orang, dan keluarga. Lihat Gambar dibawah ini



Mengidentifikasi Warisan

Warisan adalah hubungan ''semacamnya''. Kelas dasar adalah objek umum dan kelas turunan adalah contoh khusus dari objek umum. Pendekatan top-down adalah mengidentifikasi objek yang terkadang memerlukan pemrosesan khusus atau terkadang memiliki atribut khusus. Ini biasanya merupakan pendekatan yang efektif untuk menemukan warisan.

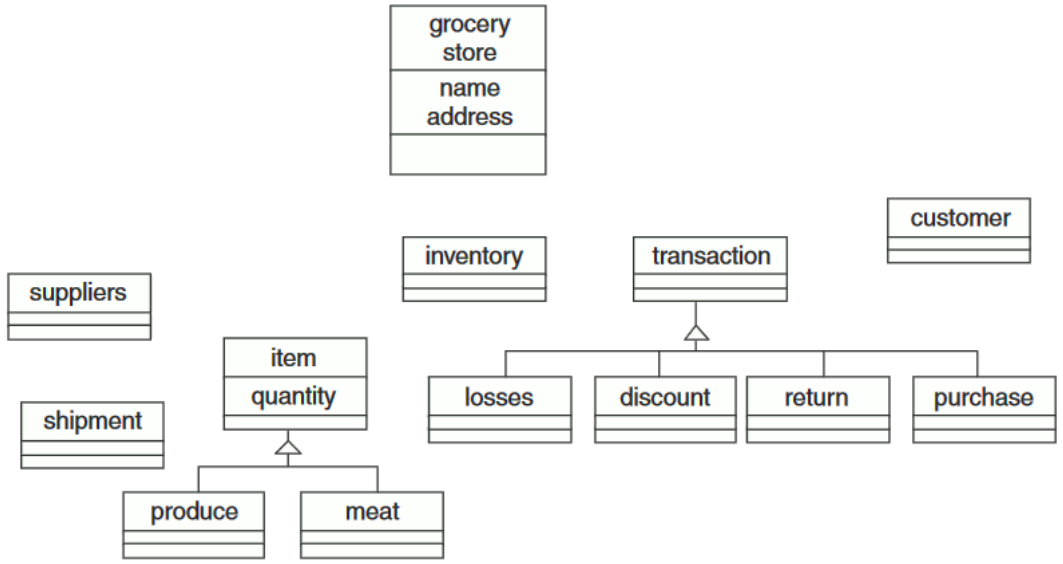
Pendekatan sebaliknya juga terkadang berguna. Ini adalah pendekatan bottom-up, yaitu mengelompokkan semua item yang serupa dan mencari kesamaannya. Perpotongan semua item serupa akan menjadi kelas dasar.

Contoh 11.6

Identifikasi kemungkinan warisan di toko kelontong (Contoh 11.4).

Pendekatan top-down akan membantu mewujudkan bahwa departemen daging dan departemen produksi memiliki pemrosesan khusus terhadap barang-barang tersebut. Hal ini akan menghasilkan kelas barang dasar dan kelas turunan daging dan hasil bumi. Seorang ahli dalam domain toko kelontong dapat membantu mengidentifikasi semua kelas turunan lain yang dapat terjadi di toko kelontong. Selain itu, pendekatan bottom-up akan menemukan kesamaan antara objek yang hilang, diskon, pengembalian, dan pembelian. Hal ini

menunjukkan bahwa objek-objek tersebut dapat diturunkan dari suatu objek transaksi. Lihat Gambar dibawah ini.



Mengidentifikasi Penggunaan Kembali

Penggunaan kembali adalah salah satu janji perangkat lunak berorientasi objek. Namun, penggunaan kembali jarang terjadi dengan sendirinya. Langkah pertama dalam mengidentifikasi penggunaan kembali adalah tugas yang disebut analisis domain. Analisis domain adalah proses meninjau domain masalah untuk menentukan objek dan fungsi apa yang umum untuk semua aktivitas. Tanpa pengetahuan domain yang baik, akan sulit untuk mengidentifikasi kesamaan apa yang ada di antara semua sistem serupa di domain tersebut. Agar penggunaan kembali menjadi efektif, bagian-bagian yang akan berguna dalam berbagai solusi dalam domain tersebut harus diidentifikasi. Ini berarti memahami potensi kesamaan.

Pendekatan penggunaan kembali dapat bersifat top-down atau bottom-up. Pendekatan bottom-up mencari kelas tingkat rendah atau menengah yang umum di sebagian besar solusi dalam domain tersebut. Pendekatan top-down melihat kesamaan dalam kerangka solusi dan perbedaan pada objek tingkat rendah.

Kecuali penggunaan kembali merupakan tujuan dan mengidentifikasi potensi penggunaan kembali serta merancang kelas agar dapat digunakan kembali adalah prioritas utama, penggunaan kembali akan sulit dilakukan.

Contoh 11.7

Identifikasi penggunaan kembali di domain toko kelontong (Contoh 11.4 dan 11.6).

Di ranah toko kelontong, nampaknya kesamaannya ada pada objek tingkat rendah. Sebagian besar toko kelontong menjual barang serupa. Beberapa aktivitas tingkat menengah akan memiliki banyak kesamaan, misalnya sistem inventaris, penyimpanan stok, dan pelacakan penjualan.

Penggunaan Pendekatan Kasus

Pendekatan lain untuk mengidentifikasi kebutuhan sistem yang diusulkan adalah memulai dengan mengidentifikasi skenario. Pendekatan ini memandang aktivitas sebagai cara

terbaik untuk menentukan fungsionalitas yang diperlukan dan objek yang diperlukan untuk mendukung fungsionalitas tersebut.

Contoh 11.8

Tuliskan skenario untuk masalah toko kelontong (Contoh 11.7). Kembangkan daftar objek dari skenario. Sebagian besar skenario akan didasarkan pada pengetahuan umum dan tidak dapat diturunkan hanya dari pernyataan masalah singkat.

Skenario 1: Persediaan hampir habis; pemasok dikirimi pesanan; pesanan tiba di dok pengiriman; barang dan jumlah dimasukkan ke dalam persediaan.

Skenario 2: Pelanggan membeli bahan makanan dan melakukan pembayaran; database pelanggan diperbarui (keputusan kembali dibuat agar sistem melacak pelanggan).

Skenario 3: Pelanggan baru memasuki toko dan diminta mengisi formulir informasi pelanggan baru dan menerima kartu keanggotaan.

Skenario 4: Petugas produksi memeriksa produk dan membuang selada tua; inventaris diperbarui.

Dari skenario ini, kita dapat dengan mudah mengidentifikasi objek berikut:

```
inventory, supplier, order, shipment, items, customer, membership
                        card, produce item
```

Hubungan pewarisan antara barang belanjaan (kelas dasar) dan barang produksi (kelas turunan) jelas.

11.3 MENGIDENTIFIKASI ASOSIASI

Setelah objek-objek dalam suatu domain teridentifikasi, langkah selanjutnya adalah mengidentifikasi hubungan antar objek. Asosiasi menunjukkan hubungan antara dua objek. Berbagai jenis asosiasi dijelaskan di Bagian 2.4. Adanya keterkaitan antar objek maksudnya dalam implementasi sistem akan ada keterkaitan antar objek. Jadi, pentingnya asosiasi adalah bahwa asosiasi menentukan akses apa yang dimiliki suatu objek terhadap objek lain. Akses ini penting untuk implementasi fungsionalitas yang efisien.

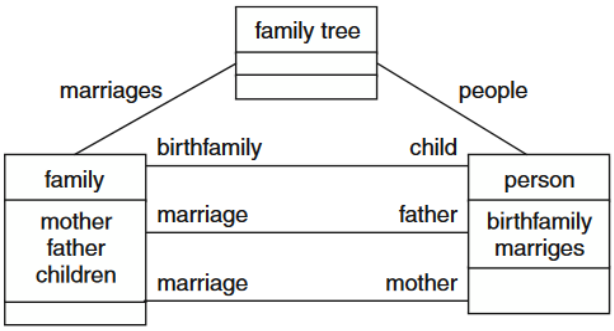
Ada pendekatan berbeda untuk menentukan hubungan antar objek. Salah satu pendekatannya adalah dengan mengidentifikasi asosiasi yang ada dalam domain masalah.

Contoh 11.9

Kembangkan asosiasi untuk masalah silsilah keluarga pada Contoh 11.5.

Rumusan masalah menyebutkan bibi, paman, dan sepupu. Ini semua adalah asosiasi (hubungan). Mereka bukanlah asosiasi primitif. Asosiasi dasar dalam silsilah adalah ibu, ayah, dan anak. Kebalikan dari masing-masing perkumpulan ini adalah perkawinan, perkawinan, dan keluarga kandung.

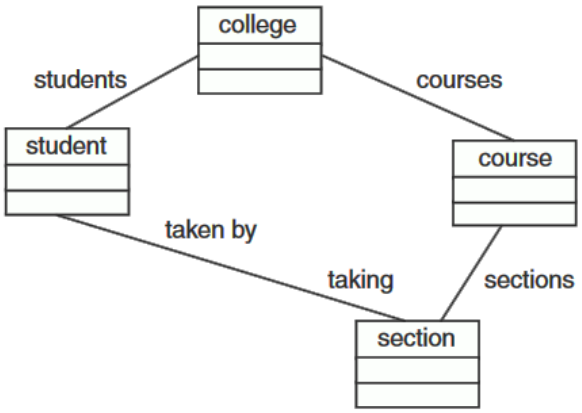
Selain itu, ada keterkaitan (agregasi) dari objek teratas, silsilah keluarga, hingga perkawinan dan ke orang. Ini masing-masing bisa disebut pernikahan dan manusia. Lihat dibawah ini.



Pendekatan lain untuk mengidentifikasi asosiasi adalah dengan memikirkan fungsionalitas yang diperlukan. Jika suatu objek diharuskan memiliki fungsionalitas yang memerlukan akses ke objek lain, maka asosiasi, atau rangkaian asosiasi, harus ada di antara objek tersebut.

Contoh 11.10

Sebuah perguruan tinggi menginginkan sistem yang menangani mata kuliah, bagian mata kuliah, dan mahasiswa. Gambarlah model objek dan identifikasi hubungan antar objek. Perguruan tinggi perlu mengakses siswa untuk mencetak informasi siswa. Untuk mencetak mata kuliah yang diambil mahasiswa, perlu adanya akses dari mahasiswa ke bagian. Untuk mencetak jadwal baris yang mencetak bagian-bagian yang tersedia, perlu ada akses ke kursus dan kemudian ke bagian untuk setiap kursus. Lihat Gambar disamping

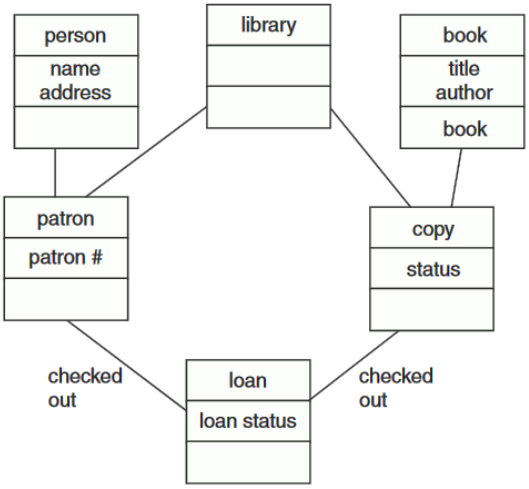


Ketergantungan Keberadaan

Pendekatan lain adalah dengan menggunakan hubungan ketergantungan keberadaan (Bagian 2.4) antar objek untuk menentukan asosiasi yang diperlukan. Dua objek mempunyai hubungan ketergantungan eksistensi jika keberadaan objek anak bergantung tepat pada satu instance objek induk. Artinya instance induk sudah ada sebelum instance anak dibuat dan instance anak dihapus sebelum instance induk dihapus.

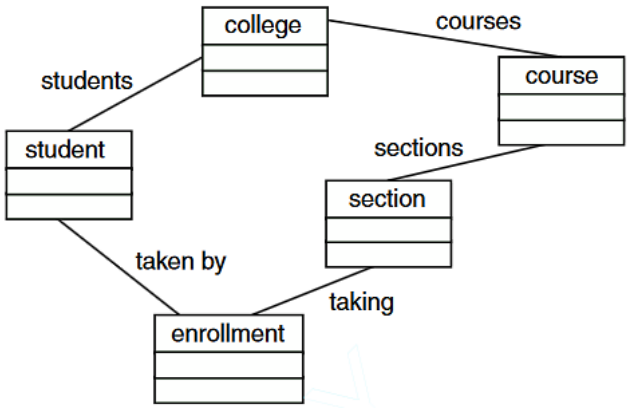
Contoh 11.11

Gunakan ketergantungan keberadaan untuk menyusun asosiasi dalam contoh perpustakaan, Contoh 2.6. Baik buku maupun manusia tidak bergantung pada perpustakaan. Namun, partisipasi mereka di perpustakaan dalam hal patron dan copy, masing-masing, memenuhi persyaratan ketergantungan keberadaan. Lihat Gambar disamping.



Contoh 11.12

Gunakan ketergantungan keberadaan untuk menentukan hubungan dalam soal bagian siswa pada Contoh 11.10. Model objek yang dikembangkan pada Contoh 11.10 tidak memenuhi aturan ketergantungan keberadaan, karena bagian tidak dapat bergantung pada keberadaan siswa atau sebaliknya. Jadi, objek tambahan yang disebut pendaftaran harus digunakan. Lihat Gambar dibawah ini



11.4 MENGIDENTIFIKASI MULTIPLISITAS

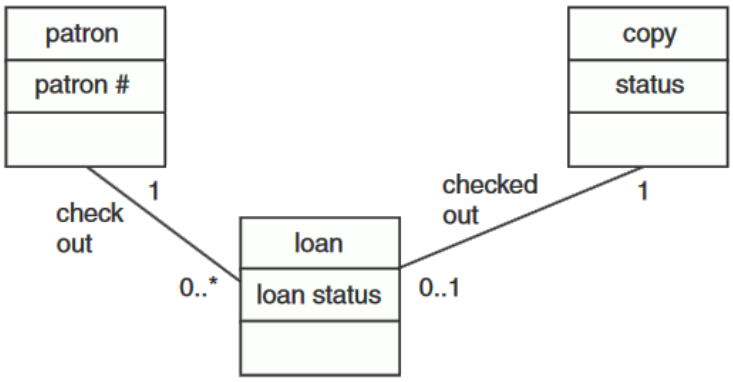
Multiplisitas adalah pembatasan pada asosiasi antar instance objek. Multiplisitas ditentukan oleh ekspresi di akhir asosiasi. Ekspresinya bisa berupa nilai tunggal, rentang nilai, atau daftar rentang atau nilai tunggal. Dalam rentang tersebut, kedua nilai dipisahkan oleh dua periode. Domain masalah sering kali memiliki batasan pada berapa banyak hubungan yang dapat dimiliki suatu instance suatu objek dengan instance objek lainnya.

Contoh 11.13

Gunakan multiplisitas untuk membatasi berapa kali salinan sebuah buku dapat dipinjam pada waktu tertentu.

Seperti yang ditunjukkan pada Gambar berikut ini, angka 0..1 di akhir pinjaman asosiasi membatasi salinan untuk berpartisipasi dalam paling banyak satu hubungan pinjaman pada satu waktu. Angka 1 di akhir salinan asosiasi mengharuskan pinjaman memiliki tepat satu asosiasi dengan salinannya. Artinya, tidak mungkin ada pinjaman tanpa tepat satu salinan yang terkait dengan pinjaman tersebut.

Asosiasi check-out membatasi contoh pinjaman untuk dikaitkan dengan satu pelindung saja. Patron dapat memiliki hubungan dengan nol atau lebih contoh pinjaman.

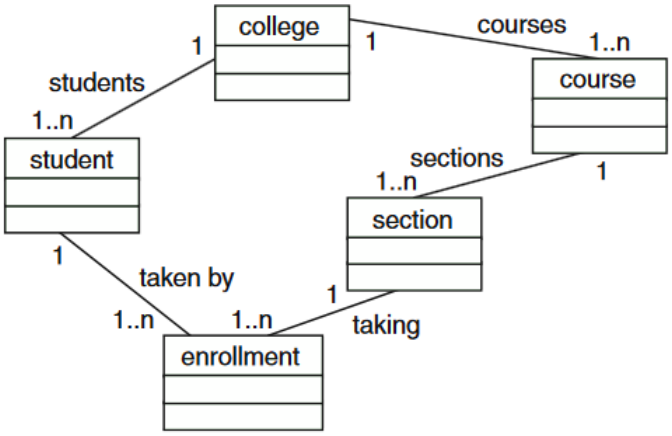


Contoh 11.14

Tentukan perkalian untuk soal bagian siswa dari Contoh 11.12.

Semua instance harus terkait dengan satu instance induk. Semua orang tua harus terkait dengan 1 hingga n instance anak. Misalnya, perguruan tinggi tanpa kursus tidak diperbolehkan (dengan model ini). Misalnya, 0..*, menetapkan bahwa tidak boleh ada nol atau lebih hubungan. Dan, setiap mata kuliah harus dikaitkan dengan satu perguruan tinggi.

Lihat Gambar dibawah ini.



LATIHAN SOAL

1. Mengapa mobil Ford merupakan spesialisasi dari mobil dan mesinnya bukan?
2. Apa perbedaan antara objek dan atribut?
3. Apa tujuan analisis domain?

Latihan soal tambahan (Diskusi)

1. Identifikasi objek dari pernyataan masalah B&B berikut:
   Tom dan Sue memulai penginapan dan sarapan di kota kecil di New England. Mereka akan memiliki tiga kamar tidur untuk para tamu. Mereka menginginkan sebuah sistem untuk mengelola pemesanan dan memantau pengeluaran dan keuntungan. Ketika calon pelanggan menelepon untuk melakukan reservasi, mereka akan memeriksa kalender, dan jika ada lowongan, mereka akan memasukkan nama pelanggan, alamat, nomor telepon, tanggal, harga yang disepakati, nomor kartu kredit, dan nomor kamar. Reservasi harus dijamin dengan pembayaran 1 hari.
   Reservasi akan dilakukan tanpa jaminan untuk waktu yang telah disepakati. Jika tidak ada jaminan pada tanggal tersebut, reservasi akan dibatalkan.
2. Identifikasi objek dari pernyataan masalah kantor gigi berikut:
   Tom memulai praktik dokter gigi di kota kecil. Dia akan memiliki asisten gigi, ahli kesehatan gigi, dan resepsionis. Dia menginginkan sistem untuk mengatur janji temu.
   Ketika seorang pasien meminta janji temu, resepsionis akan memeriksa kalender dan akan mencoba menjadwalkan pasien sedini mungkin untuk mengisi lowongan. Jika pasien puas dengan janji temu yang diusulkan, resepsionis akan memasukkan nama pasien dan tujuan janji temu. Sistem akan memverifikasi nama pasien dan memberikan rincian yang diperlukan dari catatan pasien termasuk nomor ID pasien. Setelah setiap pemeriksaan atau pembersihan, ahli kesehatan atau asisten akan menandai janji temu telah selesai, menambahkan komentar, dan kemudian menjadwalkan pasien untuk kunjungan berikutnya jika diperlukan.
   Sistem akan menjawab pertanyaan berdasarkan nama pasien dan tanggal. Rincian pendukung dari catatan pasien ditampilkan bersama dengan informasi janji temu. Resepsionis dapat membatalkan janji temu. Resepsionis dapat mencetak daftar notifikasi untuk melakukan panggilan pengingat 2 hari sebelum janji temu. Sistem

menyertakan nomor telepon pasien dari catatan pasien. Resepsionis juga dapat mencetak jadwal kerja harian dan mingguan dengan seluruh pasien.

3. Gambarkan model objek untuk permasalahan B&B (Soal 11.1).
4. Gambarlah model objek untuk permasalahan kantor gigi (Soal 11.2).

BAB 12
METRIK BERORIENTASI OBJEK

12.1 PENDAHULUAN

Pengukuran perangkat lunak berorientasi objek memiliki tujuan yang sama (lihat Bab 5) dengan pengukuran perangkat lunak konvensional: mencoba memahami karakteristik perangkat lunak. Berorientasi objek mengacu pada bahasa pemrograman dan gaya pemrograman di mana fungsi-fungsinya dienkapsulasi dengan data. Enkapsulasi dilakukan dengan membatasi aksesibilitas data pada fungsi-fungsi yang menjamin integritas data. Selain itu, perangkat lunak berorientasi objek melibatkan pewarisan dan pengikatan dinamis. Perangkat lunak berorientasi objek seharusnya memodelkan dunia nyata sehingga lebih mudah dipahami, lebih mudah dimodifikasi (dipelihara), dan lebih mudah digunakan kembali. Namun, sebagian besar kompleksitas perangkat lunak berorientasi objek tidak terlihat dalam struktur statis kode sumbernya. Area pengukuran perangkat lunak berorientasi objek adalah area penelitian.

Metrik yang disajikan di Bagian 12.2 dan 12.3 merupakan pandangan terkini mengenai hal-hal yang signifikan. Chidamber dan Kemerer mengusulkan metrik di Bagian 12.2. Bagian 12.3 menyajikan metrik MOOD. Pekerjaan lebih lanjut akan diperlukan sebelum ada konsensus mengenai metrik berorientasi objek mana yang berguna.

Pengukuran Tradisional

Pengukuran dari pengukuran perangkat lunak tradisional dapat diterapkan. Ini mungkin berguna dalam fungsi-fungsi besar. Namun, pengukuran perangkat lunak untuk perangkat lunak non-berorientasi objek menggunakan grafik aliran kontrol (dan variasi seperti grafik aliran data) sebagai abstraksi dasar perangkat lunak. Grafik aliran kontrol tampaknya tidak berguna sebagai abstraksi perangkat lunak berorientasi objek. Sedikit penelitian yang telah dipublikasikan untuk mengevaluasi penggunaan bilangan siklomatik McCabe atau ilmu perangkat lunak Halstead dalam pengembangan perangkat lunak berorientasi objek. Masalah intuitif dalam penerapan metrik perangkat lunak tradisional adalah bahwa kompleksitas perangkat lunak berorientasi objek tidak tampak pada struktur kontrol.

Abstraksi Berorientasi Objek

Pada sebagian besar perangkat lunak berorientasi objek, metodenya kecil dan jumlah keputusan dalam suatu metode seringkali sedikit. Sebagian besar kompleksitas tampaknya terletak pada pola pemanggilan di antara metode-metode tersebut. Hanya ada sedikit penelitian di bidang ini, dan hanya ada sedikit kesepakatan mengenai abstraksi mana yang penting. Abstraksi yang lebih umum dalam pengembangan perangkat lunak berorientasi objek adalah diagram yang digunakan dalam Unified Modeling Language (UML).

12.2 RANGKAIAN METRIK UNTUK DESAIN BERORIENTASI OBJEK

Metric Suite untuk Desain Berorientasi Objek dimaksudkan sebagai pendekatan komprehensif untuk mengevaluasi kelas dalam suatu sistem. Sebagian besar dihitung

berdasarkan per kelas. Artinya, tidak ada satupun yang mengevaluasi sistem secara keseluruhan. Tidak jelas bagaimana memperluas metrik ini ke seluruh sistem. Rata-ratanya pada kelas-kelas dalam suatu sistem biasanya tidak tepat.

Metrik 1: Metode Tertimbang Per Kelas (WMC)

Metrik metode berbobot per kelas didasarkan pada intuisi bahwa jumlah metode per kelas merupakan indikasi signifikan kompleksitas perangkat lunak. Pertimbangan lebih lanjut mungkin diperlukan untuk menghindari memberikan bobot penuh pada metode sepele, misalnya metode dapatkan dan tetapkan. WMC mencakup ketentuan untuk memberi bobot pada metode. Misalkan C adalah himpunan kelas yang masing-masing memiliki jumlah metode $M_1; \ldots, M_n$. Misalkan $c_1; \ldots, c_n$ adalah kompleksitas (bobot) kelas (nilai yang akan digunakan untuk c_1 tidak didefinisikan dalam buku ini).

$$\text{WMC} = \frac{1}{2} \times \sum_{i=0}^{n} c_i \times M_i$$

Ini adalah satu-satunya metrik dalam rangkaian yang dirata-ratakan pada kelas-kelas dalam suatu sistem.

Dalam contoh ini, kita asumsikan c_1 sama dengan 1.

Metrik 2: Kedalaman Pohon Warisan (DIT)

Metrik kedalaman pohon warisan hanyalah panjang maksimum dari setiap node ke akar pohon warisan untuk kelas tersebut. Warisan dapat menambah kompleksitas perangkat lunak. Metrik ini dihitung untuk setiap kelas.

Metrik 3: Jumlah Anak (NOC)

Tidak hanya kedalaman pohon warisan yang penting, namun juga lebar pohon warisan. Metrik jumlah anak adalah jumlah subkelas terdekat yang berada di bawah suatu kelas dalam hierarki pewarisan. Metrik ini dihitung untuk setiap kelas.

Metrik 4: Koupling Antar Kelas Objek (CBO)

Kopling antar modul selalu menjadi perhatian (lihat Bagian 9.5). Dalam perangkat lunak berorientasi objek, kita dapat mendefinisikan kopling sebagai penggunaan metode atau atribut di kelas lain. Dua kelas akan dianggap digabungkan ketika metode yang dideklarasikan dalam satu kelas menggunakan metode atau variabel instan yang ditentukan oleh kelas lainnya. Koplingnya simetris. Jika kelas A digabungkan ke kelas B, maka B digabungkan ke A. Metrik kopling antar kelas objek (CBO) akan menjadi hitungan jumlah kelas lain yang digabungkan. Metrik ini dihitung untuk setiap kelas.

Metrik 5: Respon Untuk Kelas (RFC)

Kumpulan respons suatu kelas, {RS}, adalah kumpulan metode yang berpotensi dieksekusi sebagai respons terhadap pesan yang diterima oleh objek kelas tersebut. Ini adalah gabungan semua metode di kelas dan semua metode yang dipanggil oleh metode di kelas. Itu hanya dihitung pada satu tingkat panggilan.

$$\text{RFC} = |\text{RS}|$$

Metrik ini dihitung untuk setiap kelas.

Metrik 6: Kurangnya Kohesi Dalam Metode (LCOM)

Modul (atau kelas) bersifat kohesif jika semuanya berkaitan erat. Kurangnya kohesi dalam metrik metode mencoba mengukur kurangnya kohesivitas.

Misalkan I_i adalah himpunan variabel instan yang digunakan oleh metode i.

Misalkan P merupakan himpunan perpotongan nol berpasangan dari I_i.

Misalkan Q merupakan himpunan perpotongan bukan nol berpasangan.

Metrik LCOM dapat divisualisasikan dengan mempertimbangkan grafik bipartit. Satu set node terdiri dari atribut, dan set node lainnya terdiri dari fungsi. Atribut ditautkan ke suatu fungsi jika fungsi tersebut mengakses atau menyetel atribut tersebut. Himpunan busur adalah himpunan Q. Jika terdapat n atribut dan m fungsi, maka kemungkinan terdapat $n \times m$ busur. Jadi ukuran P adalah $n \times m$ dikurangi ukuran Q.

$$\text{LCOM} = \max(|P| - |Q|, 0)$$

Metrik ini dihitung berdasarkan kelas.

Contoh 12.1

Hitung rangkaian metrik Chidamber pada contoh program C++ berikut yang menyediakan daftar tautan persegi panjang:

```
class point {
  float x;
  float y;
public:
  point (float newx, float newy) {x=newx; y=newy; }
  getx () {return x; }
  gety () {return y; }
};
class rectangle {
  point pt1, pt2, pt3, pt4;
public:
  rectangle(float pt1x, ptly, pt2x, pt2y, pt3x, pt3y, pt4x, pt4y)
    { pt1=new point(pt1x, pt1y); pt2=new point(pt2x, pt2y) ;
    pt3=new point(pt3x, pt3y); pt4 =new point (pt4x, pt4y) ; }
  float length (point r, point s) {return sqrt((r.getx()-
  s.getx())^2+
    (r.gety()-s.gety())^2); }
  float area(){return length(pt1,pt2) *length(pt1,pt3); }
}
class linklistnode {
  rectangle* node;
  linklistnode* next;
public:
  linklistnode (rectangle* newRectangle) {node=newRectangle;
  next=0;}
  linklistnode* getNext () {return next; }
  rectangle* getRectangle() {return node; }
```

```
  void setnext (linklistnode* newnext) {next=newnext; }
};
class rectanglelist {
  linklistnode* top;
public:
  rectanglelist () {top =0;}
  void addRectangle(float x1, y1, x2, y2, x3, y3, x4, y4) {
    linklistnode* tempLinkListNode; rectangle* tempRectangle;
    tempRectangle = new rectangle(x1,y1,x2,y2,x3,y3,x4,y4);
    tempLinkListNode = new linkListNode (tempRectangle) ;
    tempLinkListNode->setnext (top) ;
    top=tempLinkListNode; }
  float totalArea () {float sum; sum=0; linklistnode* temp;
  temp=top;
    while (temp != 0) { sum=sum + temp->getRectangle()->area();
      temp=temp->getNext (); }
    return sum; }
};
```

Metrik 1: Metode Tertimbang per Kelas

Class	#Method
point	3
Rectangle	3
Linklistnode	4
rectanglelist	3

WMC = 13/4 = 3.25 metode=kelas

Metrik 2: Kedalaman Pohon Warisan (DIT)

Tidak ada warisan dalam contoh ini.

Metrik 3: Jumlah Anak (NOC)

Tidak ada warisan dalam contoh ini.

Metrik 4: Kopling antar Kelas Objek (CBO)

Lihat gambar dibawah ini.

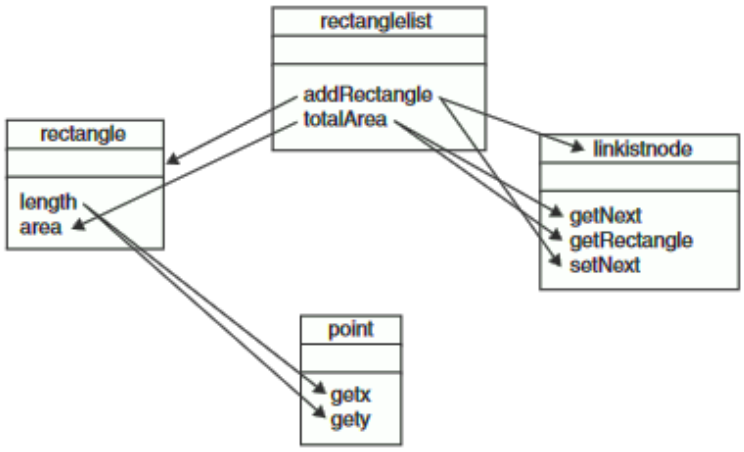


Diagram kelas dianotasi dengan panah untuk menunjukkan fungsi (atau konstruktor) mana yang dipanggil oleh setiap fungsi (hanya panggilan di kelas lain yang ditampilkan).

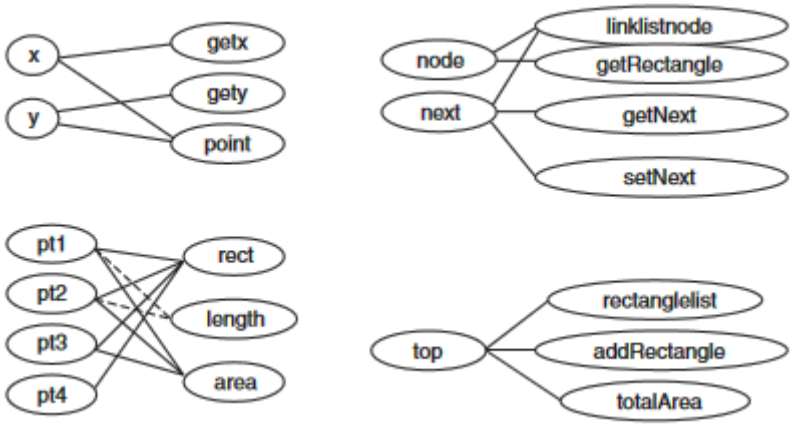
Class	Coupled Classes	CBO
point	rectangle	1
rectangle	point, rectanglelist	2
linklistnode	rectanglelist	1
rectanglelist	rectangle, linklistnode	2

Metrik 5: Respons untuk Kelas (RFC)

Class	Response Set	RFC
point	point, getx, gety	3
rectangle	rectangle, point, length, getx, gety, area	6
linklistnode	linkListNode, getNext, getRectangle, setNext	4
rectanglelist	rectangleList, addRectangle, rectangle, setNext, totalArea, gelRectangle, area, getNext	8

Metrik 6: Kurangnya Kohesi dalam Metode (LCOM)

Lihat Gambar 12-2.



Garis antara panjang dan titik diberi garis putus-putus karena bergantung pada parameter mana yang sebenarnya diakses pada panggilan tertentu.

Class	LCOM
point	max(0,(6-4)-4) = 0
rectangle	max(0,(12-9)-9)=0
linklistnode	max(0,(8-5)-5) =0
rectanglelist	max(0,(3-3)-3) = 0

12.3 METRIK MOOD

Rangkaian metrik MOOD (lihat halaman 183) juga dimaksudkan sebagai satu set lengkap yang mengukur atribut enkapsulasi, pewarisan, penggandengan, dan polimorfisme suatu sistem.

Misalkan TC adalah jumlah total kelas dalam sistem.

Misal $M_d(C_i)$ adalah banyaknya metode yang dideklarasikan dalam suatu kelas.

Misalkan predikat Is_visible($M_{m,i}, C_j$) dimana $M_{m,i}$ adalah metode m pada kelas i dan C_j adalah kelas j. Predikat ini bernilai 1 jika saya $i! = j$ dan C_j boleh menelepon $M_{m,i}$. Jika tidak, predikatnya adalah 0. Misalnya, metode publik di C++ dapat dilihat oleh semua kelas lainnya. Metode privat di C++ tidak terlihat oleh kelas lain.

Visibilitas, $V(M_{m,i})$, suatu metode, $M_{m,i}$ didefinisikan sebagai berikut:

$$V(M_{m,i}) = \frac{\sum_{j=1}^{TC} \text{Is\_visible}(M_{m,i}, C_j)}{TC - 1}$$

ENKAPSULASI

Faktor penyembunyian metode (MHF) dan faktor penyembunyian atribut (AHF) berupaya mengukur enkapsulasi.

$$\text{MHF} = \frac{\sum_{i=1}^{TC} \sum_{m=1}^{M_d(C_i)} (1 - V(M_{m,i})}{\sum_{i=1}^{TC} M_d(C_i)}$$

$$\text{AHF} = \frac{\sum_{i=1}^{TC} \sum_{m=1}^{A_d(C_i)} (1 - V(A_{m,i})}{\sum_{i=1}^{TC} A_d(C_i)}$$

CONTOH 12.2

Hitung MHF dan AHF untuk kode C++ berikut:

```
Class A{
  int a;
public:
  void x();
  void y();
};
Class B {
  int b;
  int bb;
  void w();
public:
  void z () :
};
Class C {
  int c;
  void v();
};
TC = 3
```

method	is_vis(A)	is_vis(B)	is_vis(C)	$V(M_{m,i})$
A::x()	0	1	1	1
A::y()	0	1	1	1
B::w()	0	0	0	0
B::z()	1	0	1	1
C::v()	0	0	0	0

MHF = 2/5 = 0.4

attribute	is_vis(A)	is_vis(B)	is_vis(C)	$A(M_{m,i})$
A::a()	0	0	0	0
B::b()	0	0	0	0
B::bb()	0	0	0	0
C::c()	0	0	0	0

AHF = 4/4 = 1.0

FAKTOR WARISAN

Ada dua ukuran pewarisan, yaitu faktor pewarisan metode (MIF) dan faktor pewarisan atribut (AIF).

$M_d(C_i)$ = Jumlah metode yang dideklarasikan di kelas i

$M_i(C_i)$ = Jumlah metode yang diwarisi (dan tidak ditimpa) di kelas i

$M_a(C_i) = M_d(C_i) + M_i(C_i)$ = Jumlah metode yang dapat dipanggil dalam kaitannya dengan kelas i

$$\text{MIF} = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

$A_d(C_i)$ = Jumlah atribut yang dideklarasikan pada kelas i

$A_i(C_i)$ = Jumlah atribut dari kelas dasar yang dapat diakses di kelas i

$A_a(C_i) = A_d(C_i) + A_i(C_i) =$ Banyaknya atribut yang dapat diakses terkait dengan kelas i

$$\text{AIF} = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

CONTOH 12.3

Hitung MIF dan AIF dari kode C++ berikut:

```
Class A{
protected:
  int a;
public:
  void x();
  virtual void y ();
};
Class B public A {
  int b;
protected:
  int bb;
public:
  void z():
  void y();
  void w();
};
```

```
Class C public B {
  int c;
  void v();
};
```

class	Md	Mi	Ad	Ai
A	x(),y()	None	a	none
B	w(),z(),y()	A::x()	b,bb	A::a
C	v()	B::w(),z(),y()	c	B::bb
		A::x()		

MIF = 5/11 AIF = 2/6

Faktor Koupling

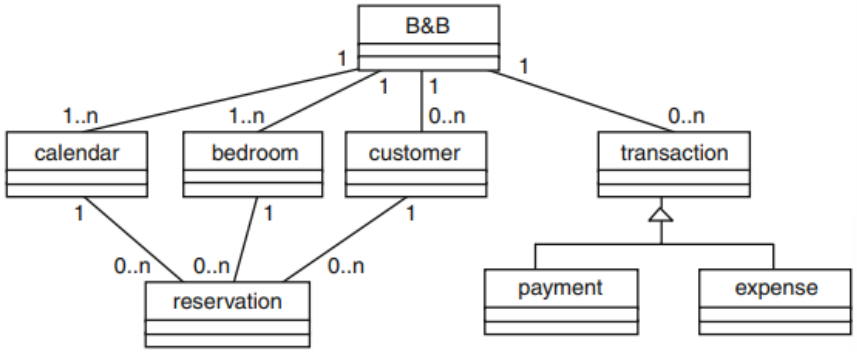
Faktor penggandengan (CF) mengukur penggandengan antar kelas tidak termasuk penggandengan karena pewarisan.

Misalkan is_client(c_i, c_j) = 1 jika kelas i mempunyai relasi dengan kelas j; jika tidak, nilainya nol. Relasinya mungkin kelas i memanggil metode di kelas j atau memiliki referensi ke kelas j atau atribut di kelas j. Hubungan ini tidak dapat diwariskan.

$$\text{CF} = \frac{\sum_{i=1}^{TC}\sum_{j=1}^{TC} is\_client(c_{i,}, c_i)}{TC^2 - TC}$$

CONTOH 12.4

Hitung faktor penghubung pada model objek yang ditunjukkan pada Gambar dibawah ini untuk masalah tempat tidur dan sarapan (Soal 11.4). Asumsikan suatu hubungan hanya jika diperlukan oleh asosiasi yang ditunjukkan pada diagram.



TC = 7

Class	is_client classes
B&B	calendar, bedroom, customer, transaction
calendar	reservation
bedroom	reservation
customer	reservation
transaction	none
payment	none

expense	none

CF = 7/42

Faktor Polimorfisme

Faktor polimorfisme (PF) adalah ukuran potensi polimorfisme.

Misalkan $M_o(C_i)$ adalah banyaknya metode utama pada kelas i.

Misalkan $M_n(C_i)$ adalah banyaknya metode baru pada kelas i.

Misalkan $DC(C_i)$ adalah banyaknya turunan dari kelas i.

$$\text{PF} = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC}[M_n(C_i) \times DC(C_i)]}$$

CONTOH 12.5

Hitung faktor polimorfisme pada kode C++ dari Contoh 12.3.

Class	Mn	Mo	DC
A	x(),y()	none	2
B	w(),z()	y()	1
C	v()	none	0

PF = 1/(2 x 2 + 2 x 1 + 1 x 0) = 1/6

LATIHAN SOAL

1. Mengapa bilangan siklomatik McCabe dan ilmu perangkat lunak Halstead tidak mudah diterapkan pada perangkat lunak berorientasi objek?
2. Abstraksi apa yang tersedia dalam desain berorientasi objek untuk digunakan sebagai dasar metrik berorientasi objek?
3. Kapan metrik untuk keseluruhan sistem harus berbeda dengan jumlah atau rata-rata metrik yang dihitung untuk setiap kelas?
4. Apakah LCOM yang tinggi baik atau buruk?
5. Ada yang berpendapat bahwa dalam LCOM hanya menggunakan selisih antara ukuran P dan Q saja. Artinya, penggunaan maksimum nol dan selisih ini tidak efektif. Apa dampak dari perubahan ini?

Latihan Soal Tambahan

1. Hitung metrik Chidamber untuk kode berikut yang mengelola array orang/siswa:

```
class person{
  char* name;
  char* ssn;
public:
  person () {name = new char [NAMELENGTH] ; ssn = new char
  [SSNLENGTH] ; }
  ~person () {delete name; delete ssn;}
  void addName (char* newname) {strcpy (name, newname)
  void addSsn (char* newssn) {strcpy(ssn, newssn) ;}
  char* getName () {return name; }
  void virtual display () {cout << "the person's name i" << name; }
};
```

```
class student public person {
  float gpa;
public:
  void addGpa (float newgpa) {gpa = newgpa; }
  void display () {cout << "the student's name is"
  << getName () << " and gpa is " << gpa; }

};
class personlist {
  person* list [MAX] ;
  int listIndex;
public:
  personlist () {listIndex=0;}
  void addPerson (char* newname, char*
  newssn) {list [listIndex] =new person;
    list [listIndex]->addName (newname); list[listIndex]
    ->addSsn (newssn) ;
    listIndex++; }
  void addStudent (char* newname, char* newssn, float gpa)
    {student* temp = new student;
    temp->addName (newname); temp->addSsn (newssn) ;
    temp->addGpa (newgpa) ; list[listIndex++]=temp; }
  void display () {int j; for(j=0; j<listIndex; j++) list[j]
  ->display();}
};
```

BAB 13
PENGUJIAN BERORIENTASI OBJEK

13.1 PENDAHULUAN

Pengujian perangkat lunak berorientasi objek menghadirkan beberapa tantangan baru. Banyak teknik konvensional yang masih sesuai. Misalnya, pengujian fungsional perangkat lunak berorientasi objek tidak akan berbeda dengan pengujian fungsional perangkat lunak konvensional. Kasus uji akan dikembangkan berdasarkan fungsionalitas yang diperlukan seperti yang dijelaskan dalam dokumentasi persyaratan. Namun, pengujian struktural perangkat lunak berorientasi objek akan sangat berbeda. Dua pendekatan pengujian struktural akan dibahas: pengujian MM dan pengujian pasangan fungsi

Perangkat Lunak Konvensional

Pengujian perangkat lunak konvensional sering kali didasarkan pada kriteria cakupan yang ditentukan pada struktur perangkat lunak. Pendekatan standar (lihat Bab 10) mencakup cakupan pernyataan, cakupan cabang, dan cakupan aliran data. Kriteria cakupan ini didasarkan pada diagram alir kendali atau diagram alir kendali yang dimodifikasi.

Perangkat Lunak Berorientasi Objek

Perangkat lunak berorientasi objek menambah kompleksitas baru pada pengujian perangkat lunak. Diagram alur kendali tidak lagi merupakan representasi yang baik dari struktur perangkat lunak. Akan lebih tepat untuk mendasarkan pengujian struktural pada model objek. Namun, tidak ditemukan ukuran cakupan model objek yang efektif.

Metode-metode di kelas hendaknya diuji dengan teknik-teknik yang telah disajikan. Kriteria cakupan yang sama dapat diterapkan pada perangkat lunak berorientasi objek. Namun secara intuitif, pernyataan dan kriteria cakupan cabang tampaknya tidak sesuai untuk menguji secara menyeluruh kompleksitas perangkat lunak berorientasi objek. Interaksi antar metode perlu diuji.

Salah satu pendekatan pengujian berorientasi objek adalah mencakup semua panggilan ke metode. Ini kadang-kadang disebut pengujian MM.

13.2 PENGUJIAN MM

Cakupan pengujian MM (pesan-metode) mengharuskan setiap pemanggilan metode diuji. Jadi, dalam setiap metode, setiap panggilan ke metode lain harus diuji setidaknya satu kali. Jika suatu metode memanggil metode lain beberapa kali, setiap panggilan hanya perlu diuji satu kali. Tampaknya ini adalah kriteria cakupan yang paling mendasar. Pengujian MM tidak mencakup cakupan setiap pernyataan (lihat Bagian 10.3).

Contoh 13.1

Identifikasi cakupan pengujian MM untuk masalah daftar persegi panjang yang ditautkan.

```
class point {
  float x;
```

```
  float y;
public:
  point (float newx, float newy) {x=newx; y=newy; }
  getx() {return x; }
  gety () {return y; }
};
class rectangle {
  point pt1, pt2, pt3, pt4;
public:
  rectangle(float pt1x, ptly, pt2x, pt2y, pt3x, pt3y, pt4x, pt4y)
    {pt1= new point(pt1x, pt1y); pt2=new point(pt2x, pt2y);
    pt3=new point(pt3x, pt3y); pt4=new point(pt4x, pt4y) ; }
  float length (point r, point s) {return sqrt ((r.getx()-
    s.getx())^2+(r.gety()-s.gety())^2); }
  float area(){return length(pt1,pt2) *length(pt1,pt3); }
};
class linklistnode {
  rectangle*node;
  linklistnode*next;
public:
  linklistnode(rectangle* newRectangle) {node=newRectangle; next=0;}
  linklistnode* getNext () {return next; }
  rectangle* getRectangle () {return node; }
  void setnext (linklistnode* newnext) {next=newnext; }
};
class rectanglelist {
  linklistnode* top;
public:
  rectanglelist () {top =0;}
    void addRectangle(float x1, y1, x2, y2, x3, y3, x4, y4) {
    linklistnode* tempLinkListNode; rectangle* tempRectangle;
    tempRectangle= new rectangle(x1,y1,x2,y2,x3,y3,x4,y4);
    tempLinkListNode = new linkListNode (tempRectangle) ;
    tempLinkListNode->setnext (top) ;
    top=tempLinkListNode; }
  float totalArea() {float sum; sum=0; linklistnode* temp; temp=top;
    while (temp != 0) {sum=sum + temp->getRectangle()->area();
      temp=temp->getNext (); }
    return sum; }
};
```

Struktur pemanggilan ditunjukkan sebagai berikut. Untuk setiap kelas, fungsi-fungsi dari kelas tersebut dicantumkan dan kemudian untuk setiap fungsi yang memanggil fungsi lain, fungsi-fungsi yang disebut tersebut didaftar. Untuk pengujian MM, setiap panggilan tersebut harus dijalankan. Misalnya, empat panggilan ke titik akan dilakukan. Tidak ada keputusan yang ditampilkan; namun, dalam program ini, tidak ada keputusan yang mempengaruhi urutan pemanggilan.

```
    class point
      point()
```

```
    getx()
    gety()

class rectangle
  rectangle()
    point::point()
    point::point()
    point::point()
    point::point()
  length()
    point::getx()
    point::getx()
    point::gety()
    point::gety()
  area()
    length()
    length()

class linklistnode
  linklistnode()
  getNext()
  getRectangle()
  setnext()
class rectanglelist
  rectanglelist()
  addRectangle()
    rectangle::rectangle()
    linklistnode::linklistnode()
    linklistnode::setnext()
  totalArea()
    linklistnode::getRectangle()
    rectangle::area()
    linklistnode::getNext()
```

Pengujian MM: Setiap kasus uji yang membangun setidaknya satu persegi panjang dan kemudian mendapatkan luas total akan menjalankan semua panggilan ini.

13.3 CAKUPAN PASANGAN FUNGSI

Cakupan pasangan fungsi mensyaratkan bahwa untuk semua kemungkinan rangkaian eksekusi metode, rangkaian yang panjangnya dua harus diuji. Hal ini biasanya dilakukan berdasarkan diagram mesin status atau ekspresi reguler yang menunjukkan kemungkinan eksekusi metode.

Karena ekspresi reguler dapat dipetakan ke mesin keadaan terbatas, kedua pendekatan ini setara. Meskipun mesin keadaan terbatas yang digunakan untuk menggambarkan perilaku sistem perangkat lunak mungkin tidak minimal, memiliki keadaan tambahan akan meningkatkan efektivitas set pengujian.

Contoh 13.2

Identifikasi cakupan pengujian pasangan fungsi untuk program daftar persegi panjang tertaut pada Contoh 13.1. Pertimbangkan struktur pemanggilan fungsi.

```
class point
  point()
  getx()
  gety()

class rectangle
  rectangle()
    point()point()point()point ()
  length()
    getx()getx()gety()gety()
  area()
    1ength()length()

class linklistnode
  linklistnode()
  getNext()
  getRectangle()
  setnext()

class rectanglelist
  rectanglelist()
  addRectangle()
    rectangle()linklistnode()setnext()
  totalArea()
    (getRectangle()area()getNext())*
```

Sebagian besar ekspresi reguler untuk masing-masing fungsi memiliki daftar pemanggilan metode yang tetap. Hanya fungsi `totalArea` yang merupakan pengulangan nol atau lebih dari perulangan `while`.

Menggabungkan semua ini menjadi satu ekspresi reguler dan mempertimbangkan bahwa daftar persegi panjang harus dibuat terlebih dahulu dan kemudian `addRectangle` atau `totalArea` dapat dilakukan memberikan ekspresi reguler berikut:

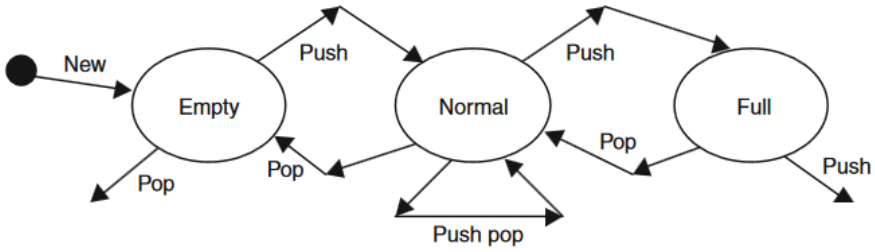
```
rectanglelist ( (addRectangle rectangle point point point point
linklistnode setnext) | (totalArea (getRectangle area length getx getx
gety gety length getx getx gety gety getNext) *)
```

Pengujian pasangan fungsi dapat dicapai dengan set pengujian berikut:

1. Membuat satu persegi panjang, lalu menghitung luasnya
2. Membuat dua persegi panjang atau lebih, lalu menghitung luasnya
3. Tidak membuat persegi panjang apa pun, lalu menghitung luasnya
4. Membuat persegi panjang setelah menghitung luasnya

Contoh 13.3

Identifikasi cakupan pengujian pasangan fungsi untuk contoh tumpukan terbatas yang ditunjukkan pada mesin status pada Gambar dibawah iniTransisi kesalahan ditampilkan sebagai busur tanpa status tujuan.



Contoh ini memerlukan pasangan berikut dalam pengujian:

1. `new pop`(on empty - error)
2. `new push`
3. `push` (from empty) `push`
4. `push` (from empty) `pop`
5. `push` (from normal to normal) `push` (still in normal)
6. `push` (from normal to normal) `push` (into full)
7. `push` (from normal to normal) `pop`
8. `push` (from normal to full) `push` (error)
9. `push` (from normal to full) `pop`
10. `pop` (from normal to normal) `push` (still in normal)
11. `pop` (from normal to normal) `pop` (still in normal)
12. `pop` (from normal to normal) `pop` (into empty)
13. `pop` (into empty) `push`
14. `pop` (into empty) `pop` (error)

Contoh 13.4

Identifikasi pasangan fungsi yang perlu dicakup untuk cakupan pengujian pasangan fungsi untuk contoh kode berikut. Buat ekspresi reguler untuk setiap fungsi yang memiliki pemanggilan metode di dalam tubuhnya dan untuk keseluruhan program.

```
  class D {
    int x;
    int Db(int s) {return 2*s; }
    int Dc (int s) {return s; }
  public:
    D(){x=0;}
    int Da(ints){if (x=1) {x=0; returnDb(s);} else {x=1; return
Dc(s) ; } }
  };
  class A {
    int x;
    int y;
    D* m;
```

```
public:
  A(){m=new D; }
  virtual int Aa(int s) {cout << x; return m->Da (y) ; }
    void add(int s, int u) {x=s;y=u; }
  };
class B {
  A* w[MAX];
  int z;
  int q;
public:
  B() {z=0; q =1; }
  int Bread(){cin>>s>>u; if (z=MAX) return 0;
    if(s<q) {w[z]=new A; w[z]->add(s,u);}
    else {w[z]=new C; w[z]->add(u,s) ;w[z]->Atadd(s);}
    z++; return z; }
  int Ba(int s) {q=w[s]->Aa(q); return q; }
};
class C public A{
  int t;
public:
  int Aa(int r) {cout << t; return m->Da (t) ; }
  void Atadd (int x) {t = x; }
};
```

Ekspresi reguler setiap fungsi dengan pemanggilan fungsi di dalamnya:

```
class D
  Da : (Db | Dc)
class A
  A: D
  Aa: Da
class B
  Bread: e | (A add | C add Atadd)
  Ba : Aa
class C
  C: A
  Aa: Da
```

Ekspresi reguler untuk kemungkinan panggilan:

```
B (Bread (e|A D add| C A D add Atadd))|Ba Aa Da (Db|Dc))*
```

Cakupan pasangan fungsi harus mencakup pasangan berikut:

```
B Bread
Bread A
Bread C
```

```
Bread Bread
B Ba
Da Db
add Bread
add Ba
Da Dc
Atadd Bread
Atadd Ba
Db Ba
Db Bread
Dc Bread
Dc Ba
```

LATIHAN SOAL

1. Bagaimana pengujian fungsional perangkat lunak berorientasi objek dilakukan?
2. Apakah cakupan pernyataan perangkat lunak berorientasi objek bermanfaat?
3. Apakah pengujian MM mencakup cakupan pernyataan? (Lihat Bagian 10.3)
4. Apa keuntungan dari cakupan pasangan fungsi?

Latihan Soal Tambahan

1. Dengan kode berikut, buat kasus uji yang memenuhi kriteria pengujian MM dan setiap pengujian pasangan fungsi:

```
class Threes {
  char cout;
public:
  Threes () { count = 'a' ; }
  void PlusOne () { if (count == 'a') count = 'b'; if (count ==
    'b')
  count='c';
    if (count == 'c') count = 'a'; }
  char* IsDiv(){if (count=='a'){return 'yes'; }else{return
  'no' ; } }
}
class Num {
  Threes* SumMod3; int last; int number;
  void Digit (int newnum) {int j; for (j=1; j <= newnum; j++)
  SumMod3->PlusOne ();}
public:
  Num () { SumMod3 = new Threes; }
  void Reduce() {while (number > 0) {last = number - (number/
    10)*10;
    Digit (last); number = number/10; }
  char* IsDivisibleBy3 (int newnum) {number = newnum; Reduce;
    return SumMod3->IsDiv (); }
}
Main () {
  Num* Test = new Num;
  int value;
```

```
  char* answer;
  cin >> value;
  answer = test->IsDivisibleBy3 (value);
  cout << answer;
};
```

BAB 14
NOTASI FORMAL

14.1 PENDAHULUAN

Notasi formal adalah notasi yang didasarkan pada matematika. Sintaks dan/atau semantik notasi mempunyai dasar matematis. Notasi formal mempunyai potensi luar biasa untuk mengurangi kesalahan selama pengembangan perangkat lunak. Manfaatnya belum terealisasi terutama karena kesulitan membangun dan menggunakan spesifikasi formal.

Masalah dengan bahasa alami adalah bahasa tersebut bersifat ambigu. Seringkali, spesifikasi bergantung pada semantik/makna kata untuk menyampaikan pemahaman yang diperlukan. Spesifikasi seharusnya menjawab pertanyaan. Spesifikasi apa pun, formal atau tidak, dapat dievaluasi seberapa baik spesifikasi tersebut dapat menjawab pertanyaan pengembang tentang perilaku yang ditentukan. Spesifikasi formal mampu menjawab pertanyaan lebih banyak dengan tepat.

Ada tiga tingkat formalisme:

- ***Informal—***Teknik yang memiliki aturan fleksibel dan tidak membatasi model yang dapat dibuat
- ***Semiformal—***Teknik yang memiliki sintaksis yang terdefinisi dengan baik
- ***Formal—***Teknik yang mendefinisikan sintaksis dan semantik secara ketat

14.2 SPESIFIKASI FORMAL

Spesifikasi formal menggunakan model yang didefinisikan secara formal untuk membuat pernyataan tentang perilaku perangkat lunak. Misalnya, spesifikasi formal mungkin menggunakan notasi himpunan sebagai modelnya. Harus ada cara untuk memetakan dari perangkat lunak ke model formal, untuk menghubungkan proses dalam perangkat lunak dengan proses dalam model formal, dan untuk memetakan pernyataan dalam model formal kembali ke pernyataan dalam perangkat lunak.

Misalnya, kita dapat menentukan perilaku tumpukan menggunakan notasi matematika suatu barisan. Kita dapat menentukan persamaan matematis untuk setiap operasi tumpukan. Kemudian, dengan adanya serangkaian operasi pada tumpukan, kita dapat memetakan operasi tersebut ke dalam operasi pada barisan matematika. Setelah menyelesaikan operasi pada urutan tersebut, kita dapat memetakan hasilnya kembali ke tumpukan. Dengan demikian, kita dapat menggunakan urutan matematis untuk secara tepat menentukan perilaku tumpukan.

Pernyataan yang biasanya dibuat dalam spesifikasi formal terbagi dalam tiga kategori: prakondisi, pascakondisi, dan invarian.

Prasyarat/Prakondisi

Prekondisi adalah pernyataan yang terkait dengan suatu fungsi yang harus benar sebelum fungsi tersebut dapat dijalankan. Ada dua gaya menafsirkan prasyarat:

- *Jangan tentukan penanganan kesalahan*—Jika prasyarat tidak terpenuhi, beberapa penanganan kesalahan akan dilakukan. Gaya ini mengasumsikan bahwa implementasi akan diperluas untuk menangani kondisi kesalahan tersebut.
- *Tentukan semua penanganan kesalahan*—Diasumsikan bahwa fungsi tidak akan dipanggil jika prasyarat tidak terpenuhi. Dengan demikian, spesifikasi diperluas untuk menentukan semua kondisi kesalahan yang diharapkan dapat ditangani oleh fungsi yang diimplementasikan.

Kondisi Pasca

Kondisi apost juga dikaitkan dengan suatu fungsi. Kondisi pasca menentukan perubahan yang terjadi pada penyelesaian fungsi. Biasanya notasi formal mempunyai notasi untuk menunjukkan situasi sebelum eksekusi dan situasi setelah eksekusi selesai. Misalnya, beberapa notasi menggunakan tanda kutip untuk menandai variabel guna mewakili nilai variabel setelah selesai, dan variabel tanpa tanda kutip mewakili nilai sebelum eksekusi fungsi dimulai.

INVARIAN

Invarian adalah pernyataan yang selalu benar. Sebenarnya, hal ini mungkin tidak benar selama eksekusi suatu fungsi atau pernyataan. Namun, hal ini akan berlaku sebelum dan sesudah selesainya setiap fungsi.

Contoh invarian untuk tumpukan mungkin adalah tumpukan tersebut berisi kurang dari atau sama dengan jumlah maksimum item yang diperbolehkan. Invarian lainnya mungkin adalah bahwa beberapa bidang tidak nol.

14.3 BAHASA KENDALA OBJEK (OCL)

Object Constraint Language (OCL) adalah bagian dari spesifikasi UML.2 Awalnya digunakan untuk menentukan bagian dari UML. Saat ini, tidak ada alat yang mendukung analisis pernyataan OCL dalam spesifikasi UML.

OCL menggunakan model objek untuk menyediakan konteks spesifikasi. Sebagian besar pernyataan OCL mengevaluasi kumpulan entitas. OCL mencakup operasi dan perbandingan yang dapat diterapkan pada koleksi yang dihasilkan.

Pernyataan OCL selalu ditulis dengan konteks. Konteksnya biasanya berupa kelas dalam model objek. Konteksnya diwakili oleh nama kelas yang digarisbawahi.

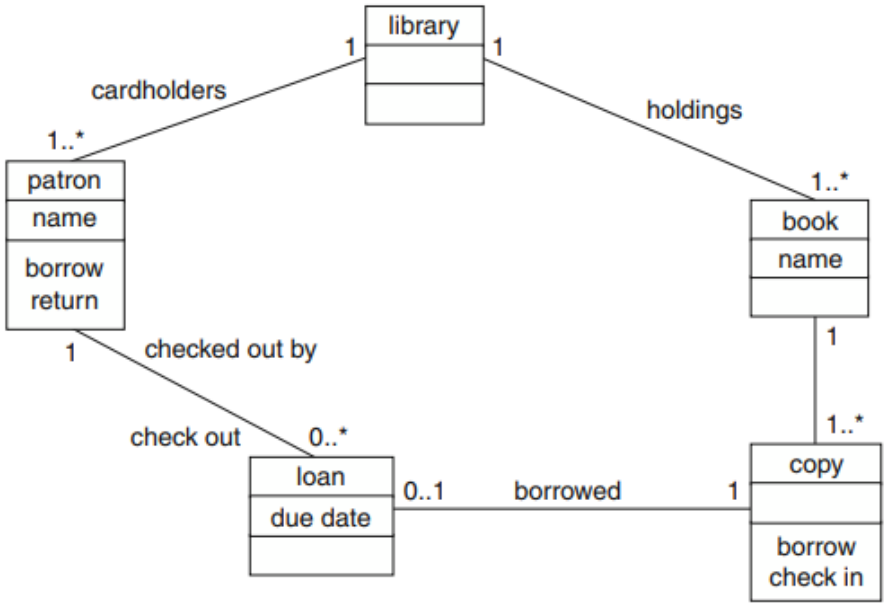
Ekspresi OCL sendiri memulai navigasi. Ini mengacu pada sebuah instance dari kelas.

Navigasi

Ekspresi OCL dapat menggunakan nama peran dari sisi berlawanan dari suatu asosiasi, nama asosiasi, atau nama kelas. Hasilnya bisa berupa koleksi atau elemen. Jika multiplisitasnya 0 atau 1, maka itu akan menjadi sebuah objek tunggal. Kalau tidak, itu akan menjadi koleksi.

Contoh 14.1

Pernyataan OCL berikut semuanya mengevaluasi himpunan seluruh pinjaman buku yang saat ini dipinjam dari perpustakaan (lihat objek yang ditunjukkan pada Gambar dibawah ini).




```
library
self.holdings.copy.borrowed
self.cardholders.checkedout
```

`library` yang digarisbawahi menyatakan konteksnya. `Self` menunjukkan sebuah instance dari perpustakaan kelas. Ekspresi `self.holdings` mengevaluasi himpunan semua instance `book` kelas. Ekspresi `self.holdings.copy` mengevaluasi himpunan semua `copy` semua `book`. Ekspresi `self.holdings.copy.borrowed` mengevaluasi himpunan semua contoh `loan book` dari `library`.

Ekspresi `self.cardholders` mengevaluasi himpunan semua contoh `patron`. Ekspresi `self.cardholders.checkedout` mengevaluasi himpunan semua contoh pinjaman.

INVARIAN

Invarian pada OCL selalu ditulis dengan konteks yang ditampilkan sebagai nama objek yang digarisbawahi. Biasanya navigasi digunakan untuk mengidentifikasi suatu koleksi atau elemen yang dibandingkan dengan koleksi atau elemen lainnya. Fungsi dapat diterapkan pada hasil navigasi.

CONTOH 14.2

Tulislah sebuah invarian untuk masalah perpustakaan menggunakan ekspresi dari Contoh 14.1.

```
library
self.holdings.copy.borrowed = self.cardholders.checkedout
```

Perpustakaan yang digarisbawahi menyatakan konteks invarian ini. Invarian menyatakan bahwa himpunan peminjaman buku yang dipinjam oleh pemegang kartu sama dengan himpunan salinan buku yang diperiksa.

```
library
self.holdings.copy = self.cardholders.checkedout.borrowed
```

Invarian sebelumnya adalah tipe yang benar, tetapi tidak benar. Ekspresi `self.holdings.copy` mengevaluasi himpunan semua salinan buku di perpustakaan. Ekspresi lainnya mengevaluasi himpunan semua contoh salinan buku yang sedang diperiksa. Ini hanya akan terjadi jika semua buku di perpustakaan sudah diperiksa.

Atribut

Ekspresi juga dapat merujuk pada nilai suatu atribut. Notasi titik yang sama digunakan.

Contoh 14.3

Tulis sebuah invarian yang mengatakan bahwa ''Grapes of Wrath'' tidak ada di perpustakaan.

```
book
self.name<>''Grapes of Wrath''
```

Konteksnya adalah buku kelas. Ekspresi `self.name` mengevaluasi nilai dari nama buku.

Operasi Yang Telah Ditentukan Sebelumnya

OCL memiliki banyak operasi pada koleksi: ukuran, jumlah (objek), termasuk (objek), jumlah, dan termasuk semua (koleksi).

Contoh 14.4

Tulis invarian yang mengatakan bahwa tidak ada patron yang dapat memeriksa lebih dari 10 buku sekaligus.

```
patron
self.checkedout->size < 10
```

Ekspresi `self.checkout` mengevaluasi kumpulan pinjaman yang terkait dengan patron. Ukuran operasi mengembalikan angka tersebut, dan invarian mengharuskan angka tersebut kurang dari 10.

Kondisi Pre- Dan Post-

Di OCL, konteks kondisi sebelum dan sesudah harus ditampilkan sebagai fungsi yang digarisbawahi. Sintaks `pre:` dan `post` membedakan kondisi sebelum dan sesudah. Hasil kata kunci dapat digunakan untuk menunjukkan hasil operasi. Sintaks `@pre` digunakan di OCL untuk menentukan nilai sebelum operasi.

Contoh14.5

Tulis ketentuan sebelum dan sesudah untuk memastikan bahwa pelindung tidak dapat membaca lebih dari 9 buku.

```
patron::borrow
pre.self.checkedout->size < 9
post.self.checkedout->size < 10
```

Atau

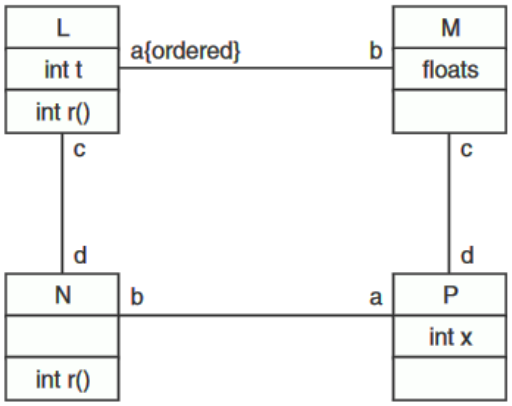
```
pre.self.checkedout-@pre>size+1=self.checkedout->size
```

LATIHAN SOAL

1. Pertanyaan seperti apa yang seharusnya dapat dijawab oleh spesifikasi?
2. Mengapa ambiguitas bisa menjadi masalah?
3. Mengapa gagasan matematika, seperti himpunan, merupakan dasar yang baik untuk spesifikasi?
4. Apa perbedaan antara prakondisi, pascakondisi, dan invarian?

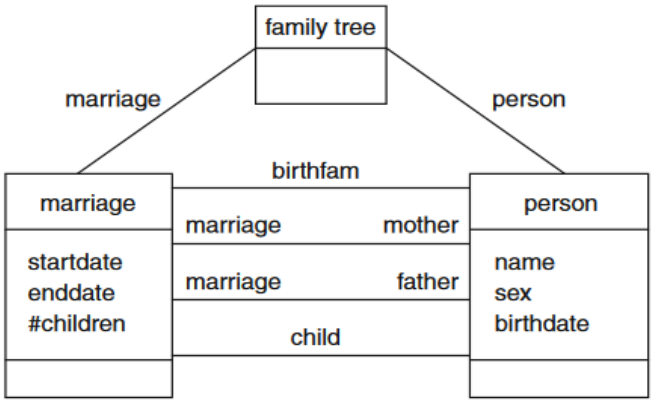
Latihan Soal Tambahan

1. Mengingat model objek yang ditunjukkan pada Gambar dibawah ini, evaluasi setiap pernyataan OCL yang diberikan. Jika pernyataan tersebut salah, jelaskan apa yang salah dan tentukan koreksi yang paling sederhana.




```
L
self.c->size= 10
self.a=self.c.b.d

L::r():int
pre: self.a.b=self.c.a
post: t = t@pre + 1
post: result = self.a->first.s

P
self.a.d->size > max
self.a.b=self.d.c

N::q():int
pre: self.b->isEmpty
pre: self.d->forall(1 |1.t <10)
post: result = self.d->size
```

2. Berdasarkan model objek yang ditunjukkan pada Gambar dibawah ini, jelaskan setiap pernyataan OCL. Apa yang ditentukannya? Apakah invarian OCL masuk akal? Apakah itu selalu benar?




```
familytree
  a) self.person = self.marriage.child
marriage
  b) self.child.birthfam =self
  c) self.husband.birthdate < self.wife.birthdate
person
  d) self.birthfam.child_include(self)
  e) self.marriage->size = 1
  f) self.marriage.wife.birthdate < self.birthdate
```

3. Tulis batasan OCL untuk restoran tanpa bagian merokok yang menempatkan pelanggan berdasarkan urutan kedatangan.

```
Class group
  Char* name
  Int number
  Int arrivalorder
Class waitlist
  Group* list [MAX]
  Int listptr
  Void addtolist (group* newgroup)
  Group* seatnext ()
Class restaurant
  Waitlist* waiting
  Void arrive(group* newgroup)
  Group* seat ()
```

DAFTAR PUSTAKA

Adams, Robert. 2021. "Software Quality Assurance: Principles and Practices." Hoboken: Wiley.

Baker, Richard. 2020. "Software Requirements Specification: A Unified Framework." New York: Wiley.

Brown, Susan. 2021. "Introduction to Software Engineering." Boston: Pearson Education.

Chen, Cheng. 2021. "Software Configuration Management Best Practices." San Francisco: Morgan Kaufmann.

Chen, Jing. 2020. "User Interface Design for Software Engineers." Amsterdam: Elsevier.

Chen, Wei. 2020. "Agile Software Development: Principles, Patterns, and Practices." San Francisco: Morgan Kaufmann.

Chen, Wei. 2023. "Software Design Patterns: Principles and Examples." San Francisco: Morgan Kaufmann.

Clark, Emily. 2021. "Secure Software Development: A Practical Guide." Boston: Addison-Wesley.

Garcia, Antonio. 2023. "Model-Driven Software Engineering in Practice." Boca Raton: CRC Press.

Garcia, Juan. 2020. "Requirements Engineering Processes and Techniques." Boston: Pearson Education.

Garcia, Luis. 2021. "Domain-Driven Design: Tackling Complexity in the Heart of Software." Boston: Addison-Wesley.

Garcia, Maria. 2022. "Software Testing Techniques and Strategies." Boca Raton: CRC Press.

Garcia, Maria. 2023. "DevOps Handbook: Continuous Deployment and Automation." Seattle: O'Reilly Media.

Hernandez, Carlos. 2022. "Software Engineering for Embedded Systems." New York: Wiley.

Kim, Hyun. 2021. "Software Metrics: A Comprehensive Guide." New York: Wiley.

Kim, Jung. 2023. "Software Performance Testing: Methods and Tools." New York: Addison-Wesley.

Kim, Minji. 2021. "Software Quality Assurance: Principles and Practices." Berlin: Springer.

Kim, Yuna. 2022. "Agile Software Development with Scrum." Amsterdam: Elsevier.

Kumar, Ajay. 2021. "Software Testing: Concepts and Techniques." Amsterdam: CRC Press.

Lee, David. 2022. "Software Architecture: Foundations, Theory, and Practice." San Francisco: Morgan Kaufmann.

Lee, Jiho. 2021. "Software Project Estimation Techniques." London: Springer.

Lee, Minwoo. 2021. "Software Product Management: Essentials." Boca Raton: CRC Press.

Li, Tao. 2022. "Software Prototyping: Principles and Techniques." London: Springer.

Lopez, Carlos. 2022. "Lean Software Development: Agile Practices for Agile Teams." New York: Addison-Wesley.

Lopez, Carlos. 2023. "Model-Driven Software Engineering." Amsterdam: Elsevier.

Martinez, Javier. 2022. "Software Maintenance and Evolution: A Roadmap." Amsterdam: Elsevier.

Martinez, Sofia. 2023. "Software Evolution and Maintenance." Berlin: Springer.

Nguyen, Anh. 2022. "Software Architecture: Foundations, Theory, and Practice." Berlin: Springer.

Nguyen, Minh. 2023. "Artificial Intelligence in Software Engineering." Cambridge: MIT Press.

Nguyen, Thi. 2020. "Requirements Engineering: A Unified Framework." Berlin: Springer.

Park, Eunice. 2020. "Software Metrics: A Practical Guide." Boca Raton: CRC Press.

Park, Minho. 2022. "Software Architecture Documentation in Practice." Berlin: Springer.

Park, Seung. 2023. "Lean Software Development Principles and Practices." Amsterdam: Elsevier.

Patel, Anil. 2021. "Software Configuration Management: Principles and Practice." Amsterdam: Elsevier.

Patel, Deepak. 2020. "Software Reliability Engineering: Principles and Practices." New York: Wiley.

Patel, Ramesh. 2020. "Requirements Engineering: A Good Practice Guide." London: Springer.

Patel, Ravi. 2020. "Software Design Patterns: Elements of Reusable Object-Oriented Software." New York: Addison-Wesley.

Scott, David. 2020. "Software Engineering Ethics: Principles and Guidelines." Hoboken: Wiley.

Smith, John. 2023. "Agile Software Development: Principles and Practices." New York: Wiley.

Smith, Peter. 2023. "Software Verification and Validation: Concepts and Techniques." London: Springer.

Taylor, Sarah. 2021. "Empirical Software Engineering: Concepts and Practices." Boston: Pearson Education.

Thompson, Laura. 2021. "Software Process Improvement: Best Practices." San Francisco: Morgan Kaufmann.

Turner, Michael. 2020. "Human-Centered Software Engineering." Boca Raton: CRC Press.

Wang, Jing. 2022. "Software Testing Automation: Tools and Techniques." London: Springer.

Wang, Li. 2022. "Software Project Management: Modern Approaches." Berlin: Springer.

Wang, Tao. 2023. "Empirical Studies in Software Engineering." Boston: Pearson Education.

Wang, Xiaoyan. 2023. "Software Architecture Patterns and Anti-Patterns." Boston: Addison-Wesley.

Wilson, Andrew. 2020. "Software Architecture Patterns." New York: Wiley.

Young, Megan. 2023. "Agile Software Testing: Strategies and Techniques." San Francisco: Morgan Kaufmann.

Zhang, Wei. 2022. "Continuous Integration and Continuous Deployment: Tools and Practices." Boston: Pearson Education.

LAMPIRAN
KUNCI JAWABAN

BAB 1

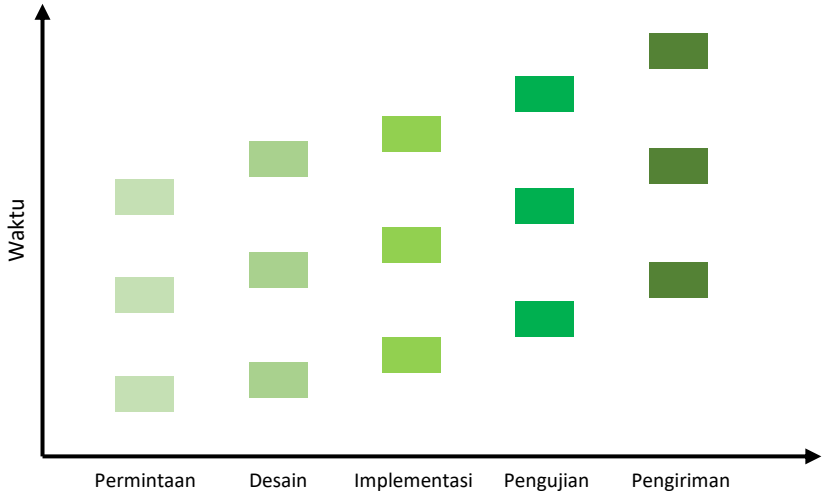
LATIHAN SOAL

1. Bagaimana model siklus hidup bertahap membantu manajemen perangkat lunak?
   Siklus hidup bertahap meningkatkan visibilitas proyek. Proyek ini dapat dikelola dengan menggunakan tahapan sebagai tonggak sejarah. Fase yang lebih rinci akan memungkinkan pemantauan kemajuan yang lebih dekat.
2. Apa dua karakteristik yang diperlukan dari suatu pencapaian?
   Milestone (1) harus berkaitan dengan kemajuan dalam pengembangan perangkat lunak dan (2) harus jelas ketika hal tersebut telah dicapai.
3. Dokumen dalam siklus hidup perangkat lunak:

Panduan pengguna akhir	Fase implementasi
Desain arsitektur	Fase desain
rencana SQA	Tahap perencanaan proyek
Spesifikasi modul	Fase desain
Kode sumber	Fase implementasi
Laporan kerja	Fase kelayakan
Rencana pengujian	Fase persyaratan
Panduan pengguna awal	Fase persyaratan
Desain yang rinci	Fase desain
Estimasi biaya	Tahap perencanaan proyek
Rencana proyek	Tahap perencanaan proyek
Laporan pengujian	Fase pengujian
Dokumentasi	Fase implementasi

4. Urutan tugas:
   Analisis Pasar
   Perencanaan proyek, perkiraan biaya, spesifikasi kebutuhan (dapat dilakukan secara bersamaan)
   Tinjauan persyaratan
   Desain tingkat tinggi
   Desain tingkat rendah
   Tinjauan desain
   Penerapan
   Pengujian satuan
   Pengujian sistem
   Ujian penerimaan

5. Gambarlah diagram yang mewakili model siklus hidup berulang. Lihat Gambar.



BAB 2

LATIHAN SOAL

No	Soal	Jawaban
1	Apa perbedaan antara model siklus hidup perangkat lunak dan model proses?	Model siklus hidup perangkat lunak (SLC) menunjukkan fase-fase utama dan hasil utama, sedangkan model proses menggambarkan tugas-tugas tingkat rendah, artefak yang dibutuhkan dan diproduksi, dan aktor yang bertanggung jawab untuk setiap tugas tingkat rendah.
2	Apa perbedaan antara model proses deskriptif dan model proses preskriptif?	Model proses deskriptif menggambarkan apa yang terjadi dalam pengembangan perangkat lunak. Hal ini sering kali berkembang sebagai hasil analisis postmortem. Model preskriptif menggambarkan apa yang harus dilakukan selama pengembangan perangkat lunak, termasuk tanggapan terhadap situasi kesalahan.
3	Mengapa pengambilan keputusan lebih umum dilakukan pada model proses preskriptif dibandingkan model proses deskriptif?	Keputusan lebih umum terjadi pada model proses preskriptif karena model proses preskriptif mencoba mencakup apa yang dilakukan dalam situasi alternatif. Oleh karena itu, mungkin perlu untuk menentukan keputusan yang digunakan untuk memutuskan apa yang harus dilakukan selanjutnya. Model proses deskriptif menggambarkan apa yang terjadi, dan tindakan alternatif biasanya tidak disertakan.
4	Mengapa tugas-tugas dalam model proses harus dipisahkan berdasarkan artefak?	Jika satu proses mengikuti proses lainnya, beberapa informasi atau dokumen dari proses pertama diperlukan oleh proses kedua. Jika hal ini tidak benar, maka kedua proses tersebut akan independen satu sama lain dan proses yang satu tidak akan mengikuti proses yang lain. Informasi atau dokumen ini harus diidentifikasi dan didokumentasikan.

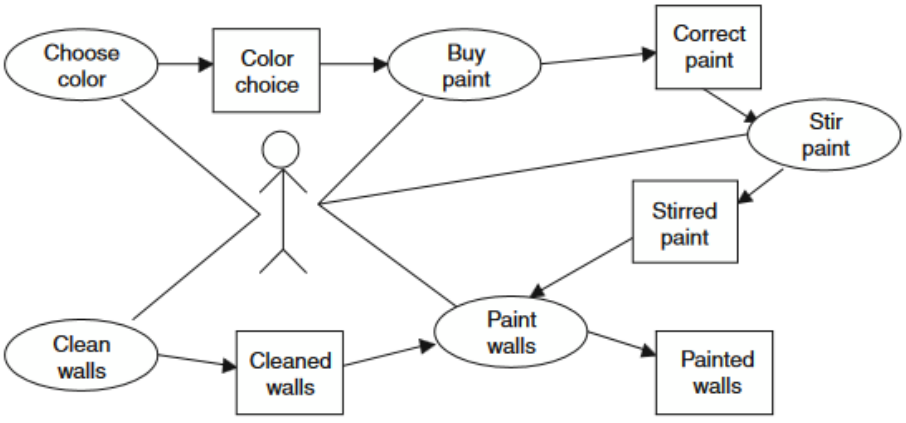
5	Mengapa tugas dalam model proses tidak dapat dimulai sampai artefak masukannya ada?	Jika suatu proses bergantung pada informasi dari proses lainnya, proses kedua ini tidak dapat dimulai sampai proses pertama selesai. Beberapa notasi proses lainnya memungkinkan konkurensi antara dua tugas, dimana proses pertama harus dimulai sebelum proses kedua dimulai dan proses kedua harus selesai setelah proses pertama selesai.
6	Mengapa setiap node dalam model proses harus mempunyai jalur dari node awal ke node itu sendiri dan jalur dari node itu sendiri ke node terminal?	Jika tidak ada jalur dari node awal ke setiap node, maka beberapa node dalam model proses tidak akan pernah dapat dijangkau dan dapat dihilangkan. Jika tidak ada jalur dari node saat ini ke node terminal, maka terdapat loop tak terhingga. Tak satu pun dari situasi ini yang diinginkan dan perlu diselidiki.
7	Dalam Contoh 2.3, bagaimana seseorang dapat membedakan antara tinjauan pengujian yang hanya terjadi setelah semua pengujian unit selesai dan tinjauan pengujian yang terjadi setelah setiap modul diuji unit?	Label pada tanda panah hasil pengujian mengandung arti bahwa keluaran dari modul pengujian unit mempunyai hasil eksekusi lebih dari satu kali. Oleh karena itu, ini menyiratkan bahwa peninjauan pengujian hanya terjadi setelah beberapa unit diuji.
8	Apa yang ditentukan oleh diagram aliran data tentang aliran kontrol?	Diagram aliran data tidak menentukan aliran kontrol. Beberapa informasi urutan mungkin dapat disimpulkan tetapi tidak ada yang lain.
9	Kapan posisi penembakan jaring petri menyala?	Posisi penembakan jaring petri diaktifkan ketika terdapat token pada setiap node masukan dari posisi penembakan.
10	Apa yang terjadi bila jaring petri menyala?	Sebuah token ditempatkan pada setiap node keluaran dari posisi penembakan.
11	Apa perbedaan antara domain masalah dan ruang solusi?	Ranah permasalahan merupakan bagian dari dunia nyata dan terdiri dari entitas-entitas yang ada di dunia nyata. Ruang solusi terdiri dari entitas perangkat lunak dalam implementasi solusi.
12	Perubahan apa saja yang terjadi pada model objek mulai dari persyaratan hingga desain?	Awalnya, objek adalah entitas dalam domain masalah. Saat pengembangan memasuki tahap desain, objek-objek tersebut menjadi entitas dalam ruang solusi.
13	Klasifikasikan masing-masing hubungan berikut sebagai hubungan warisan, hubungan agregasi, atau asosiasi umum:	Mobil—Mobil kota Lincoln Warisan Orang—Siswa Warisan Perpustakaan—Pelindung perpustakaan Pengumpulan Buku—Salinan Asosiasi umum Mobil—Supir Asosiasi umum Pelindung—Pinjaman buku Asosiasi umum Kelas—Siswa Pengumpulan
14	Klasifikasikan masing-masing berikut ini sebagai suatu kelas atau turunan dari suatu kelas:	Mobil saya Contoh Orang Kelas Fred Contoh Kendaraan Kelas Profesor Kelas

		departemen CIS Instansi
15	Apa hubungan antara skenario dan diagram keadaan yang menunjukkan semua kemungkinan rangkaian tindakan?	Skenarionya hanya berupa satu jalur yang melalui sebagian atau seluruh diagram keadaan.
16	Dalam diagram interaksi, apakah kelas pemanggil atau kelas yang dipanggil berada di ujung panah?	Kelas yang dipanggil berada di ujung panah. Fungsi pada panah harus merupakan fungsi dari kelas di ujung panah.
17	Jelaskan mengapa relasi agregasi merupakan relasi pada domain masalah dan bukan pada domain implementasi?	Tidak terdapat perbedaan penerapan relasi agregasi dengan relasi asosiasi lainnya. Faktanya, sulit untuk memutuskan apakah suatu relasi benar-benar merupakan agregasi atau bukan. Misalnya, jelas bahwa mobil merupakan kumpulan bagian-bagian mobil. Namun, tidak jelas apakah sebuah toko harus dianggap sebagai kumpulan pelanggan.
18	Mengapa diagram aliran data tidak memiliki aturan tentang jangkauan antar node?	Diagram aliran data tidak menunjukkan kontrol. Dengan demikian, dua proses mungkin tidak dihubungkan dalam diagram aliran data. Jika setiap proses menggunakan data masukan yang berbeda dan menghasilkan data keluaran yang berbeda, dan tidak ada keluaran dari satu proses yang digunakan sebagai masukan untuk proses lainnya, maka tidak akan ada busur di antara proses-proses tersebut

Latihan Tambahan (Diskusi)

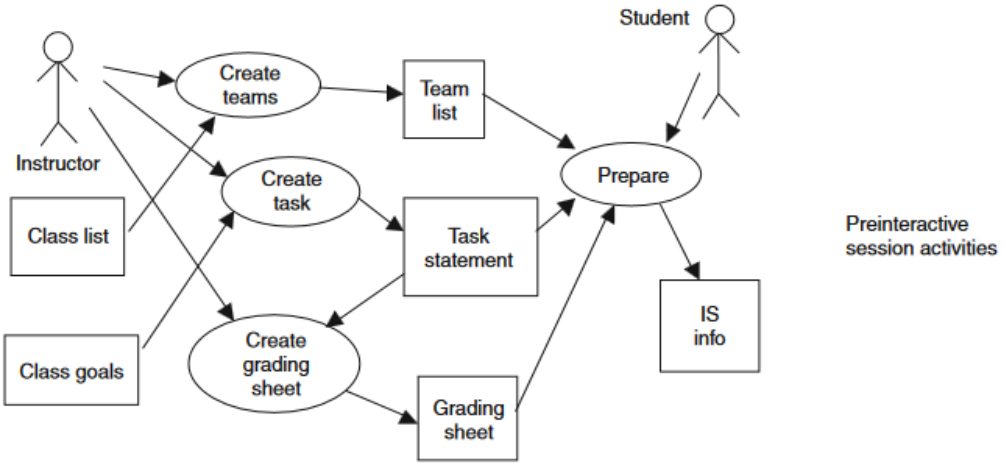
1. Menggambar model proses tugas pengecatan dinding suatu ruangan. Termasuk tugas-tugas berikut: memilih warna, membeli cat, membersihkan dinding, mengaduk cat, dan mengecat dinding.

   Jawab:

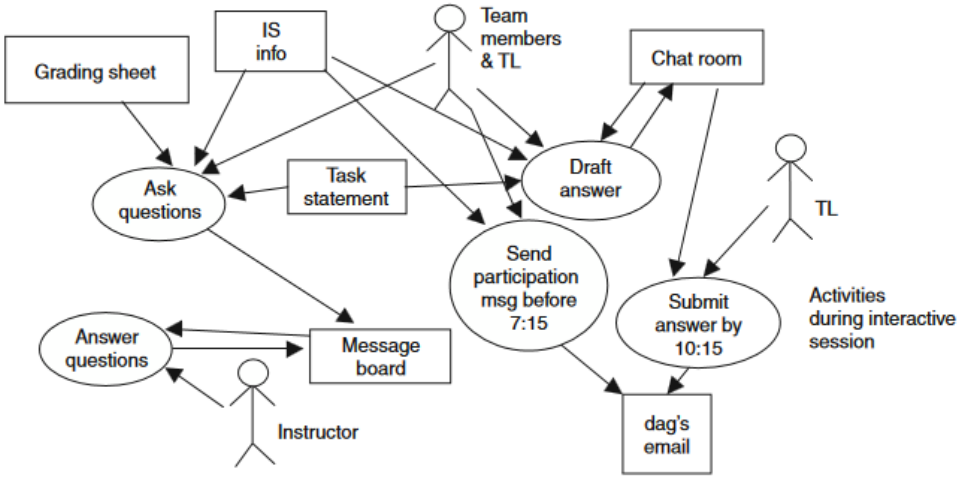


2. Penulis menggunakan sesi interaktif ketika dia mengajar mata kuliah yang mencakup siswa pembelajaran jarak jauh. Penulis membagi siswa menjadi beberapa tim dan memposting masalahnya di halaman Web. Tim mengerjakan masalah menggunakan ruang obrolan, mengajukan pertanyaan kepada instruktur menggunakan papan pesan,

dan menyampaikan solusi melalui email. Instruktur kemudian menilai solusi menggunakan lembar penilaian. Gambarkan model proses untuk sesi interaktif.
Jawab: Lihat Gambar dibawah ini

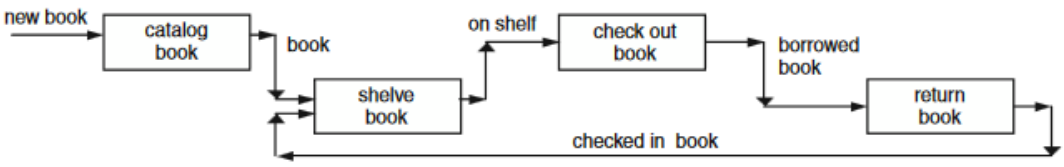


Model proses untuk mempersiapkan sesi interaktif.

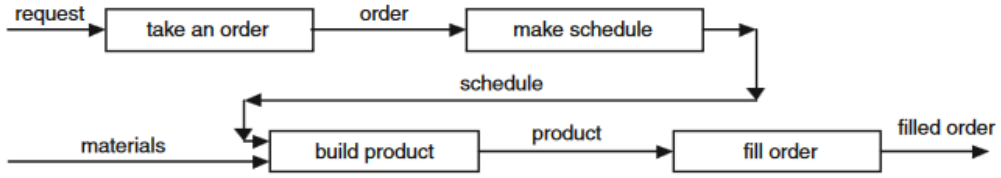


Model proses untuk tugas selama sesi interaktif.

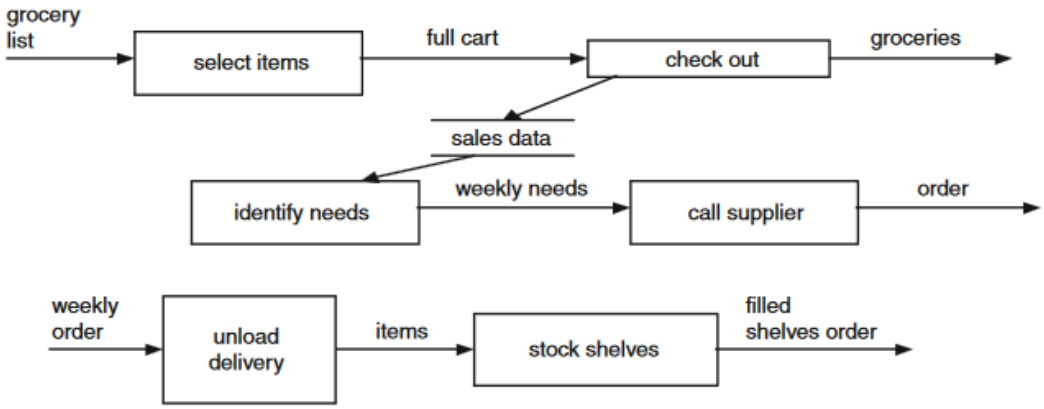
3. Gambarlah diagram aliran data untuk masalah perpustakaan sederhana.
Jawab: Lihat Gambar dibawah ini



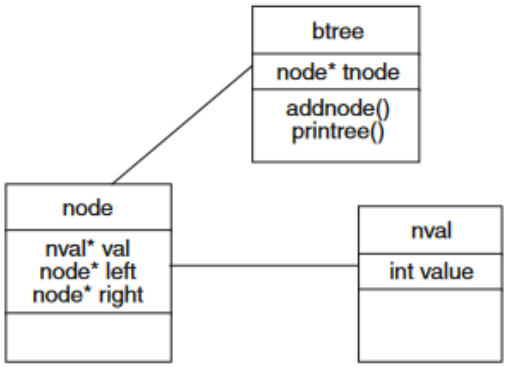
4. Gambarlah diagram aliran data untuk masalah pabrik
Jawab: Lihat gambar dibawah ini



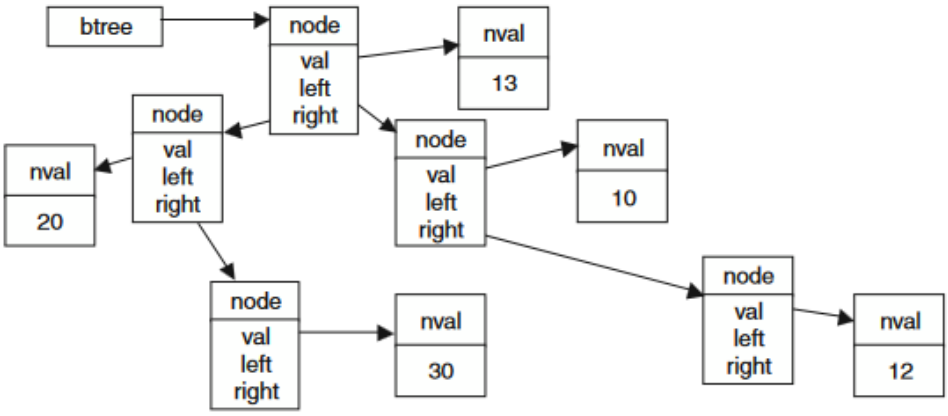
5. Gambarlah diagram aliran data untuk toko kelontong. Lihat Gambar 2-22.
   Jawab: Liat gambar berikut ini



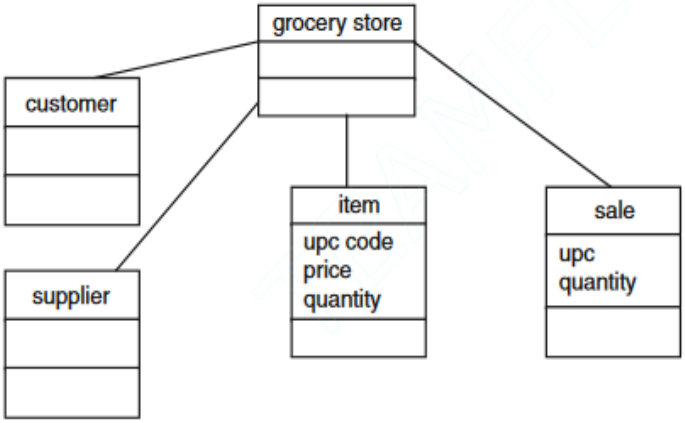
6. Gambarlah model objek untuk pohon biner.
   Jawab: Lihat gambar berikut ini



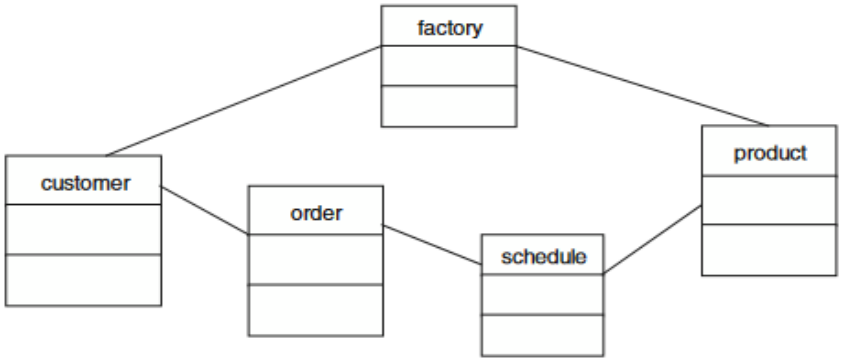
7. Gambarlah diagram contoh model objek pohon biner.
   Jawab: Lihat gambar berikut ini



8. Gambarlah model objek untuk soal toko kelontong.
   Jawab: Lihat gambar berikut ini

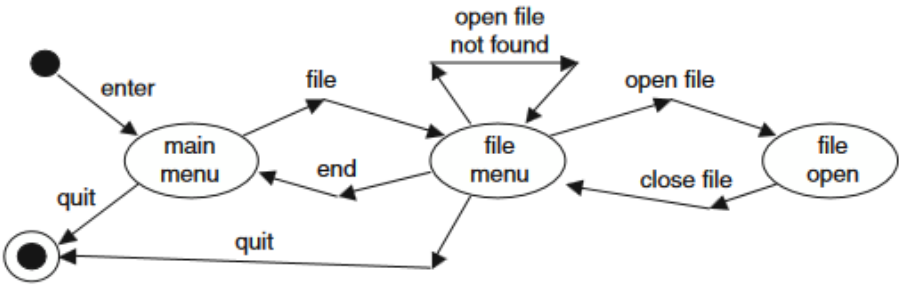


9. Gambarlah model objek untuk soal pabrik
   Jawab: Lihat gambar berikut ini



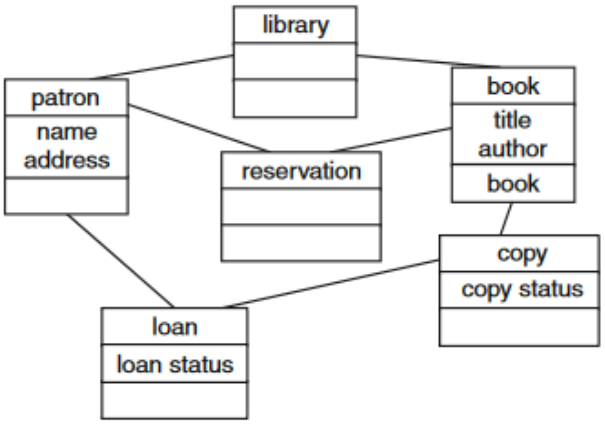
10. Tuliskan skenario tambahan untuk patron yang memeriksa buku dari Contoh 2.11. Fred pergi ke perpustakaan dan tidak dapat menemukan buku untuk dibaca?
    Jawab: Fred pergi ke perpustakaan dan memeriksa dua buku. Kemudian dia kembali ke perpustakaan dan memeriksa tiga buku lagi. Fred mengembalikan tiga buku kedua tepat waktu. Fred terlambat mengembalikan dua buku pertama.

11. Gambarlah diagram keadaan untuk antarmuka pengguna grafis yang memiliki menu utama, menu file dengan perintah buka file, dan perintah keluar pada setiap menu. Asumsikan hanya satu file yang dapat dibuka dalam satu waktu.
    Jawab:

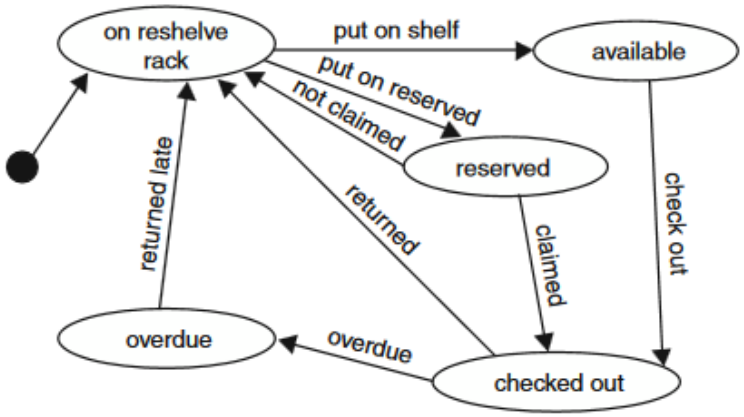


Perhatikan bahwa ''tutup file'' tidak disebutkan dalam spesifikasi masalah tetapi transisi keluar dari status ''buka file'' diperlukan.

12. Perluas model objek berikut untuk masalah perpustakaan dengan menyertakan objek reservasi sehingga pengunjung dapat memesan buku yang semua salinannya telah diperiksa.
    Jawab: perhatikan gambar dibawah ini



13. Membangun mesin negara untuk masalah perpustakaan dengan kemampuan memesan buku.
    Jawab: Perhatikan gambar dibawah ini



BAB 3
LATIHAN SOAL

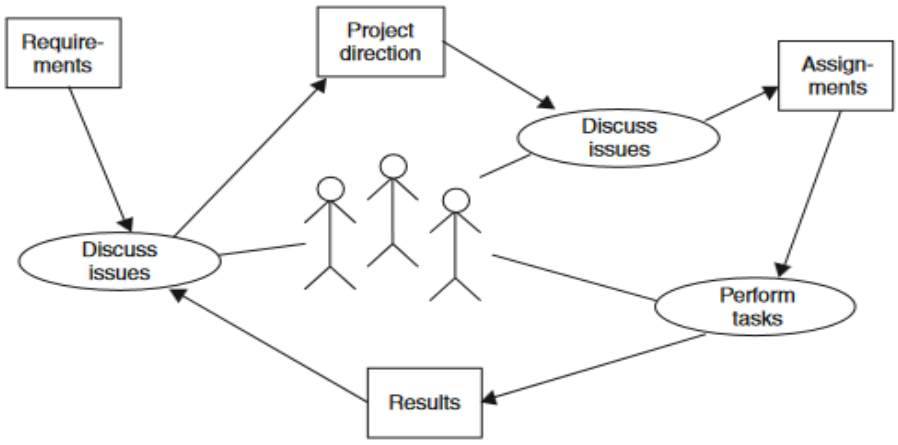
No	Soal	Jawaban
1	Apa yang dimaksud dengan visibilitas?	Visibilitas adalah atribut untuk dapat melihat kemajuan atau kekurangan kemajuan dalam suatu proyek.
2	Apa perbedaan antara pendekatan proses dan pendekatan proyek?	Pendekatan proses mirip dengan jalur perakitan, dimana setiap orang mempunyai tugas yang harus diselesaikan. Pengembang dapat melakukan tugas yang sama pada beberapa proyek—misalnya, tim pengujian atau tim desain. Penekanan proyek akan memberikan tim tanggung jawab atas seluruh upaya dalam mengembangkan proyek.

3	Untuk proyek baru yang sangat berbeda dari proyek sebelumnya, apakah pendekatan proses atau manajemen proyek akan lebih baik?	Manajemen proses bekerja dengan baik dengan proyek yang dipahami dengan baik. Sebuah proyek baru yang sangat berbeda mungkin akan lebih baik dikelola dengan pendekatan proyek yang menekankan keberhasilan dalam proyek tersebut.
4	Apa keuntungan membuat banyak tugas kecil yang hanya berukuran ''inci-kerikil''?	Jika tenggat waktu atau tugas terlewat, proyek akan tertunda. Semakin kecil waktu antar tenggat waktu, semakin cepat diketahui jika suatu proyek terlambat. Dikatakan bahwa sebuah proyek hanya dapat melewatkan jangka waktu penugasannya sebelum manajer dapat melihat penundaannya.
5	Ukuran kemajuan nilai perolehan apa yang dapat dikurangi selama suatu proyek?	Semua kecuali nilai yang diperoleh, yang harus meningkat.
6	Pengukuran kemajuan nilai yang diperoleh manakah yang nilainya lebih besar dari 1 barang?	Untuk SPI dan SV, nilai yang lebih besar dari 1 menunjukkan bahwa pencapaian lebih banyak dari yang dijadwalkan. Untuk CPI dan CV, nilai yang lebih besar dari 1 menunjukkan bahwa upaya tersebut kurang dari perkiraan. Jadi keempatnya bagus jika nilainya lebih besar dari 1.
7	Apa keuntungan menggunakan invers SPI dan CPI?	Kebalikannya masing-masing dapat digunakan sebagai alat proyeksi. Jika kebalikan dari SPI adalah 2, berarti proyek akan memakan waktu dua kali lebih lama dari perkiraan. Jika kebalikan dari CPI adalah 2, maka proyek tersebut akan memerlukan upaya dua kali lipat dari perkiraan.

Latihan Soal Tambahan (Diskusi)

1. Gambarkan model proses untuk tim yang memiliki struktur lemah dan bergantung pada diskusi tim untuk menetapkan arah dan menyelesaikan masalah.

   Jawab:



2. Dengan menggunakan log waktu berikut, hitung produktivitas pemrogram dalam LOC/hari. Asumsikan proyek 1 adalah 120 LOC dan proyek 2 adalah 80 LOC.

   Jawab:

Tanggal	Mulai	Selesai	Interupsi	Delta	Tugas
2 Januari	08:30	16:30	60 menit makan siang		Proyek 1 Coding
2 Februari	09:00	17:00	30 menit makan siang		Proyek 1 Coding

2 Mei	09:00	17:30	30 menit makan siang, 60 menit meeting		Proyek 2 Coding
2 Juni	07:30	12:00			Proyek 2 Coding

Waktu delta hari 1 adalah 8 jam 1 jam makan siang = 420 menit; hari ke 2 adalah 8 jam 30 menit = 450 menit. Jadi produktivitas proyek 1 adalah 120 LOC/ 870 menit = 120 LOC/ 2.175 hari = 55 LOC/hari programmer (asumsikan 400 menit per hari programmer). Waktu delta untuk hari ke 3 dan 4 adalah 7 jam dan 4,5 jam = 690 menit. Produktivitasnya adalah 80 LOC/ 1,725 hari = 46,4 LOC/programmer-hari. Secara keseluruhan, programmer rata-rata mendapatkan 200 LOC/ 3,9 hari = 51,3 LOC/programmer-hari.

3. Dengan menggunakan log pekerjaan berikut, hitung semua ukuran dasar dan indikator kemajuan. Apakah proyeknya sesuai jadwal?
   Asumsikan saat ini tanggal 5 Januari.

Tugas	Perkiraan Pengerjaan (hari)	Upaya Aktual Sampai saat ini (Hari)	Perkiraan Tanggal Penyelesaian	Tanggal Sebenarnya selesai
1	50	70	15 Januari	1 Februari
2	35	20	15 Februari	15 Februari
3	20	40	25 Februari	1 Maret
4	40	40	15 April	1 April
5	60	10	1 Juni	
6	80	20	1 Juli	

BCWS adalah 50+35+20+40=145 hari programmer. BAC adalah 50+35+20+40+60+80=285 hari programmer. Nilai rencana (PV) tugas pekerjaan adalah 17,5 persen, 12,3 persen, 7,0 persen, 14,0 persen, 21,1 persen, 28,1 persen. Nilai yang diperoleh adalah 17,5 persen + 12,3 persen + 7 persen + 14 persen=50,7 persen. BCWP untuk 01/05/01 sama dengan BCWS dalam contoh ini karena pekerjaan yang dijadwalkan telah selesai. Jadi, SPI=145/145=1.

Varians jadwalnya adalah 145 – 145=0. Indeks kinerja biaya= 145/170 = 85,3 persen. Hal ini menunjukkan bahwa upaya aktual lebih besar dibandingkan upaya yang diperkirakan. Varians biayanya adalah 145 – 170= – 25. Hal ini juga menunjukkan bahwa upaya yang diperlukan lebih besar dari perkiraan.

Proyek ini tampaknya sesuai jadwal tetapi memakan biaya lebih dari yang direncanakan

4. Gunakan spreadsheet untuk menghitung PV dan indikator kemajuan untuk proyek berikut dengan interval setengah bulan dari 1 Januari hingga 1 September

Jawab:

Tugas	Perkiraan Pengerjaan (hari)	Upaya Aktual Sampai saat ini (Hari)	Perkiraan Tanggal Penyelesaian	Tanggal Sebenarnya selesai
1	30	37	1 Januari	1 Februari
2	25	24	15 Februari	15 Februari
3	30	41	1 Maret	15 Maret
4	50	47	15 April	1 April

5	60	63	1 Mei	15 April
6	35	31	15 Mei	1 Juni
7	55	58	1 Juni	1 Juni
8	30	28	15 Juni	15 Juni
9	45	43	1 Juli	15 Juli
10	25	29	1 Agustus	15 Agustus
11	45	49	15 Agustus	1 September

	bcw	pv	acw	jadwal	aktual
1	30	0.070	37	1 Januari	1 Febuari
2	25	0.058	24	15 Februari	15 Februari
3	30	0.070	41	1 Maret	15 Maret
4	50	0.116	47	15 April	1 April
5	60	0.140	63	1 Mei	15 April
6	35	0.081	31	15 Mei	1 Juni
7	55	0.128	58	1 Juni	1 Juni
8	30	0.070	28	15 Juni	15 Juni
9	45	0.105	43	1 Juli	15 Juni
10	25	0.058	29	1 Agustus	15 Agustus
11	45	0.105	49	15 Agustus	1 September

BAC 430

	bcw	bcw	acwp	ev	spi	sv	cpi	cv
1 Jan	30	0	0	0.00	0	-30	0	0
15 Feb	30	0	0	0.00	0	-30	0	0
1 Feb	30	30	37	0.07	1.00	0	0.81	-7
15 Feb	55	55	61	0.13	1.00	0	0.90	-6
1 Mar	85	55	61	0.13	0.65	-30	0.90	-6
15 Mar	85	85	102	0.20	1.00	0	0.83	-17
1 Apr	85	135	149	0.31	1.59	50	0.91	-14
15 Apr	135	195	212	0.45	1.44	60	0.92	-17
1 Mei	195	195	212	0.45	1.00	0	0.92	-17
15 Mei	230	195	212	0.45	0.85	-35	0.92	-17
1 Jun	285	285	301	0.66	1.00	0	0.95	-16
15 Jun	315	315	329	0.73	1.00	0	0.96	-14
1 Jul	360	315	329	0.73	0.88	-45	0.96	-14
15 Jul	360	360	372	0.84	1.00	0	0.97	-12
1 Ags	385	360	372	0.84	0.94	-25	0.97	-12
15 Ags	430	385	401	0.90	0.90	-45	0.96	-16
1 Sep	430	430	450	1.00	1.00	0	0.96	-20

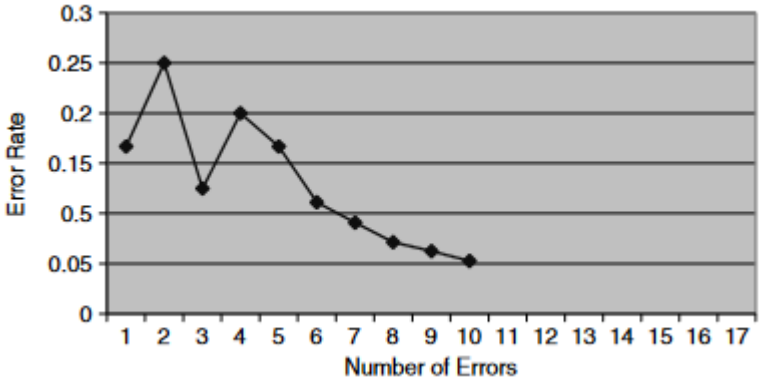
5. Seorang profesor mempunyai 40 pekerjaan rumah dan 40 ujian yang harus dinilai. Ujian biasanya memakan waktu 3 kali lebih lama untuk menilai tugas pekerjaan rumah. Hitung PV untuk setiap pekerjaan rumah dan untuk setiap ujian. Setelah 5 jam, jika profesor telah menyelesaikan setengah dari ujiannya, berapa lama waktu yang dibutuhkan untuk menyelesaikan penilaiannya?

Jawab: Asumsikan satuan penilaian sama dengan 1 tugas pekerjaan rumah. Maka tugas ini mempunyai total 40*1+40*3=160 unit penilaian. Setiap pekerjaan rumah mempunyai nilai rencana sebesar 1/160 = 0,625 persen, dan setiap ujian mempunyai nilai rencana sebesar 1,875 persen. Setelah 5 jam, 20 ujian selesai atau 37,5 persen. Jadi, 5/0,375 = 13,33 jam sebagai perkiraan waktu total, atau tersisa 8,33 jam.

6. Mengingat waktu antar-kesalahan berikut (yaitu, waktu antara terjadinya kesalahan), gunakan plot untuk memperkirakan jumlah total kesalahan awal dan waktu untuk menghilangkan seluruh kesalahan.
   Jawab:

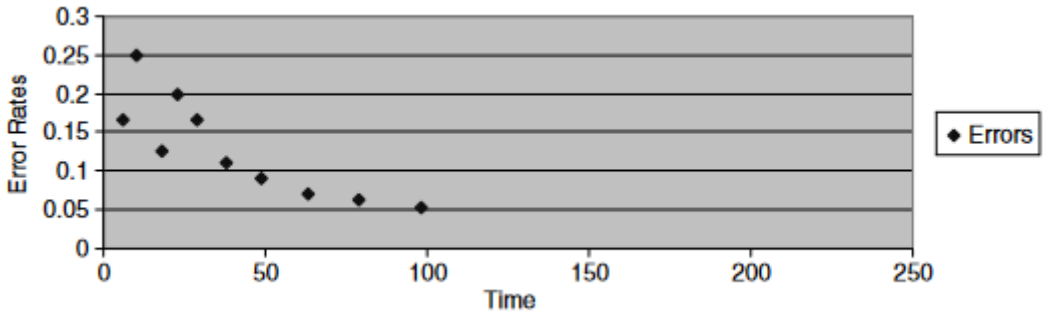
   `6, 4, 8, 5, 6, 9, 11, 14, 16, 19`

   Kebalikan dari waktu antar-kesalahan adalah tingkat kesalahan sesaat. Merencanakan angka-angka ini terhadap angka kesalahan akan menghasilkan plot yang menunjukkan tren penurunan angka kesalahan, seperti yang ditunjukkan pada gambar dibawah ini



   Memasang garis lurus akan menunjukkan perpotongan x sekitar 15. Menggunakan ini sebagai perkiraan jumlah total kesalahan awal, kami memperkirakan bahwa masih ada lima kesalahan dalam perangkat lunak.

   Tingkat kesalahan juga dapat diplot terhadap waktu yang berlalu (jumlah waktu antar kesalahan sebelumnya), seperti yang ditunjukkan pada gambar dibwah ini.



   Memasangkan garis lurus ke titik-titik ini akan menghasilkan perpotongan x mendekati 160. Hal ini akan memberikan waktu pengujian tambahan sebesar 62 unit untuk menghilangkan semua kesalahan. Perpotongan y akan menjadi sekitar 0,25. Area di bawah garis ini adalah 0:5 160 0:25, atau 20 kesalahan. Ini berarti tersisa sekitar 10

kesalahan. Perbedaan antara kedua perkiraan ini menunjukkan betapa kasarnya pendekatan ini.

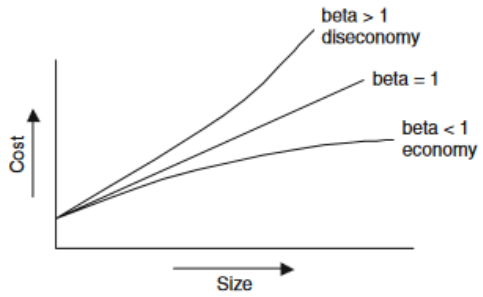
7. Proyek dimulai pada tanggal 1 Januari dan harus selesai pada tanggal 1 Juni. Sekarang tanggal 1 Maret. Lengkapi tabel berikut. Hitung EV, SPI, SV, dan CV. Tentukan apakah proyek tersebut tepat waktu. Benarkan jawaban Anda. Tunjukkan pekerjaan Anda.
   Jawab:

Tugas #	Estimasi Waktu	Waktu Aktual	PV	Due Date	Selesai
1	30 Hari	10 Hari	.15	1 Feb	
2	20 Hari	30 Hari	.10	1 Maret	YA
3	50 Hari	30 Hari	.25	1 Mei	YA
4	100 Hari	5 Hari	.50	1 Juni	

   BAC = 200 BCWS = 50 BCWP = 70 ACWP = 60
   EV = 70/200 = 0.35 SV = 70 – 50 = 1.4 CV = 70 – 60 = 10

BAB 4
LATIHAN SOAL

No.	Soal	Jawaban
1	Apa perbedaan antara WBS dan model proses?	Model proses menggambarkan aktivitas perangkat lunak dalam pengertian umum, yaitu untuk banyak proyek berbeda. Ini menggambarkan proses, artefak yang diproduksi dan digunakan, dan aktor yang bertanggung jawab atas kegiatan tersebut. Model proses adalah grafik; itu bisa memiliki siklus dan seterusnya. Struktur rincian kerja adalah pohon yang memperluas aktivitas model proses dengan rincian dan subtugas penting yang spesifik untuk proyek tertentu. Tugas WBS yang muncul di setiap proyek juga harus ada dalam model proses.
2	Mengapa WBS harus berupa pohon?	Jika ada loop di WBS, itu berarti beberapa tugas bergantung pada dirinya sendiri secara rekursif, dan ini tidak mungkin. Perulangan biasanya akan terjadi ketika beberapa tugas tidak didefinisikan dengan benar. Jika ada dua jalur menuju suatu tugas, biasanya ini berarti bahwa dua tugas berbeda dengan tingkat lebih tinggi bergantung pada subtugas umum tersebut. Itu hanya perlu ditampilkan (dan dilakukan) sekali.
3	Apa jadinya jika tidak ada kriteria penyelesaian suatu tugas di WBS?	Artinya, tidak jelas apakah kemajuan telah dicapai atau kapan hal tersebut telah tercapai. Hal ini juga dapat berarti bahwa tidak jelas apa yang sebenarnya perlu dilakukan. Misalnya, subtugas seperti ''penelitian XXX'' tidak ditentukan dengan baik. Harus ada tujuan penelitian dan harus dinyatakan dengan jelas dalam uraian tugas.
4	Apa keuntungan menggunakan diagram PERT?	Meskipun WBS akan mengembangkan daftar tugas, mungkin sulit untuk melihat tugas mana yang harus

		diselesaikan terlebih dahulu dan tugas mana yang akan menentukan waktu penyelesaian akhir. Tugas-tugas ini berada pada jalur kritis.
5	Mengapa menunda tugas di jalur kritis menunda keseluruhan proyek?	Jalur kritis didefinisikan sebagai serangkaian tugas yang menentukan waktu minimum untuk menyelesaikan proyek. Praktisnya, jika suatu tugas berada pada jalur kritis dan tertunda, berarti waktu mulai dan waktu berakhir tugas berikutnya pada jalur kritis akan tertunda. Hal ini akan berdampak pada tugas terakhir di jalur kritis, dan proyek akan tertunda.
6	Apakah jalur kritis penting jika hanya satu orang yang mengerjakan suatu proyek?	Hal ini penting hanya jika semua tugas berada pada jalur kritis. Jika ada tugas yang tidak berada pada jalur kritis dan tugas tersebut tidak dapat diselesaikan secara paralel karena hanya ada satu orang, maka waktu yang diperlukan untuk tugas tersebut akan ditambahkan ke waktu jalur kritis untuk menentukan waktu penyelesaian.
7	Apa pentingnya waktu senggang?	Meskipun tugas yang tidak berada pada jalur kritis tidak menentukan waktu penyelesaian minimal, namun jika tugas tersebut tidak diselesaikan tepat waktu maka waktu penyelesaian akan tertunda. Waktu kelonggaran menunjukkan rentang waktu di mana tugas tersebut harus diselesaikan.
8	Mengapa slack time didasarkan pada waktu mulai paling awal dari tugas-tugas berikutnya?	Waktu kendur didasarkan pada waktu mulai paling awal karena jika tugas tersebut ditunda melewati waktu mulai tersebut, tugas tersebut tidak akan selesai tepat waktu dan efek riaknya akan menunda keseluruhan proyek.
9	Gambarlah diagram yang menunjukkan skala ekonomi dan skala disekonomi. Beri label pada diagram dan jelaskan yang mana.	Garis yang melengkung ke atas menunjukkan skala disekonomis. Artinya, dengan proyek yang lebih besar, biaya per unit ukurannya meningkat. Garis yang melengkung ke bawah menunjukkan skala ekonomi. Artinya, semakin besar proyek maka semakin murah biaya per unit ukurannya. 
10	Sangat umum menggunakan versi estimasi default. Pertimbangkan kapan terakhir kali seseorang meminta Anda memberikan perkiraan tentang sesuatu. Apakah estimasi yang Anda berikan merupakan definisi default	Ketika mahasiswa saya bertanya kapan suatu tugas akan dinilai, saya terlalu sering memberikan waktu yang diperlukan jika saya tidak mempunyai tugas lain yang harus dikerjakan dan saya tidak disela. Jawaban tersebut tidak realistis karena selalu ada tugas dan interupsi yang lebih mendesak.

	estimasi atau definisi estimasi yang diusulkan DeMarco?	
11	Mengapa parameter estimasi biaya harus ditentukan dari data perusahaan?	Setiap perusahaan memiliki praktik, standar, kebijakan, dan jenis perangkat lunak berbeda yang dikembangkannya. Adalah tidak realistis untuk mengharapkan bahwa parameter-parameter yang dianggap sebagai alat prediksi yang baik oleh kontraktor-kontraktor pertahanan besar harus sama dengan parameter-parameter yang digunakan untuk proyek-proyek pembangunan berskala kecil yang dikelola sendiri.

Latihan Soal Tambahan (Diskusi)

1. Membuat WBS untuk mengecat ruangan. (Aktivitas model proses tidak ditampilkan.)
   Jawab:

A. Pilih warna untuk ruangan	Warna memutuskan
B. Beli cat	Kaleng cat
C. Beli kuas	kuas
D. Siapkan dinding	Dinding bersih
E. Buka kaleng cat	Kaleng terbuka
F. Aduk cat	Cat diaduk
G. Cat dinding	Dinding dicat
H. Membersihkan	Dinding dibersihkan dan dicat

2. Membuat WBS untuk masalah kantor gigi.
   (Fase siklus hidup dan aktivitas PM telah ditinggalkan.)
   Jawab:

Mengembangkan penilaian risiko	Spesifikasi penilaian
Perkirakan upaya	Estimasi biaya
Jadwal rencana	Jadwal
Tinjau risiko, estimasi, dan jadwal dengan dokter gigi	persetujuan dokter gigi
Model objek desain	Model objek
Tinjau model objek dengan dokter gigi	persetujuan dokter gigi
Dapatkan spesifikasi sistem catatan pasien	Spesifikasi
Mengembangkan antarmuka prototipe dengan catatan pasien	Prototipe yang berfungsi
Review form daftar pengingat dengan resepsionis dan dokter Gigi	persetujuan dokter gigi
Review form jadwal harian dan mingguan dengan staff gigi	persetujuan dokter gigi
Tinjau desain model kelas dengan tim	Tinjau dokumen
Melaksanakan	Kode yang dikompilasi
Uji unit berhasil dengan cakupan C0	Laporan pengujian
Integrasikan sistem	Kode yang dikompilasi
Tes sistem	Laporan pengujian
Pelatihan dan pengujian alfa di kantor gigi	Laporan pengujian
Tinjau sistem dengan staf gigi	persetujuan dokter gigi
Tes penerimaan oleh staf	persetujuan dokter gigi
Memberikan panduan pengguna dan dokumentasi	Manual dikirimkan

3. Buat WBS untuk masalah B&B.
   (Aktivitas PM dan hasil yang dicapai telah ditinggalkan.)
   Jawab:

Analisis Kelayakan
- Diskusikan opsi berbasis web dengan Tom dan Sue.
- Tentukan Merancang prototipe sistem reservasi.
- Bagian biaya dan keuntungan desain.
- apakah sistem berbasis web atau berdiri sendiri.

Persyaratan
- Dapatkan persyaratan dari Tom dan Sue.
- Dokumen persyaratan pembuatan.
- Tinjau persyaratan dengan Tom dan Sue.

Desain
- Merancang prototipe sistem reservasi.
- Desain bagian biaya dan keuntungan.

Penerapan
- Membangun prototipe sistem reservasi.
- Menerapkan bagian biaya dan keuntungan.

Pengujian
- Prototipe uji.
- Uji bagian pengeluaran dan keuntungan.

Pengiriman
- Tinjau prototipe dengan Tom dan Sue.
- Tinjau keseluruhan sistem dengan Tom dan Sue.

4. Membuat WBS untuk masalah dealer mobil.
(Fase siklus hidup dan aktivitas PM telah ditinggalkan.)
Jawab:

Persyaratan
- Dapatkan persyaratan dari dealer.
- Dokumen persyaratan pembuatan.
- Tinjau persyaratan dengan dealer.
- Buat panduan pengguna awal.
- Tinjau panduan pengguna awal.
- Membangun kasus uji.

Desain
- Desain prototipe sistem mobil.
- Format desain untuk laporan.

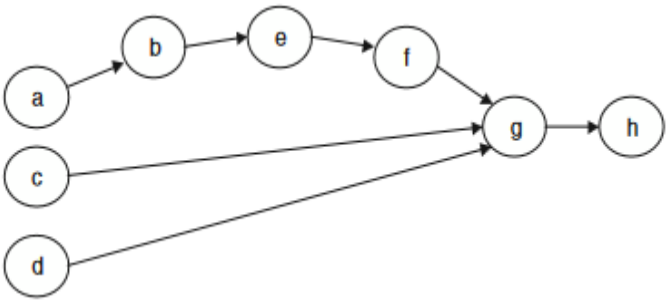
Penerapan
- Membangun prototipe sistem mobil.
- Menerapkan bagian kueri dan laporan.
- Tinjau prototipe dengan dealer.
- Menerapkan versi final.

Pengujian
- Uji versi final di dealer.

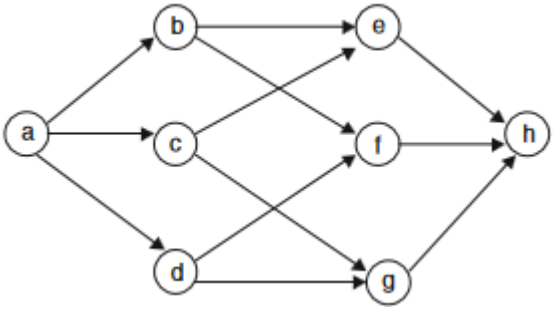
Pengiriman
- Staf kereta

Kirimkan dokumentasi.

5. Gambarlah diagram PERT untuk mengecat ruangan: Diagram PERT ditunjukkan pada Gambar dibawah ini:
   Jawab:



6. Gambarlah diagram PERT dan lengkapi tabelnya.
   (Jalur kritis ditunjukkan dengan tanda bintang pada tabel.)
   Jawab:



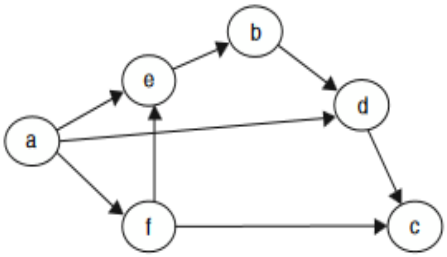
Simpul	Bagian	Waktu	Mulai	Selesai
a		10	0	10*
b	a	5	10	15*
c	a	2	10,24	12,26
d	a	3	10,21	13,24
e	b,c	7	12,26	19,33
f	b,d	9	24	33*
g	c,d	5	13,28	18,33
h	e,f,g	6	33	39*

7. Gambarlah diagram PERT dan lengkapi tabel untuk kumpulan tugas dan ketergantungan yang diberikan.
   Jawab:

Simpul	Dependensi	Waktu	Mulai	Selesai
a		10	0	10
b	e	10	23	33
c	d,f	10	53	63
d	a,f,b	20	33	53

e	a,f	8	15	23
f	a	5	10	15

Semuanya berada pada jalur kritis; tidak ada waktu senggang, Lihat gambar berikut ini:



8. Perkirakan parameter biaya dari kumpulan data yang diberikan?
   Jawab:
   Biaya = 3.8 x Ukuran (KLOC)

9. Perkirakan parameter biaya dari kumpulan data yang diberikan:
   Jawab:
   Cost = 4.0 x Size (KLOC) +5.0

10. Hitung upaya COCOMO, TDEV, staf rata-rata, dan produktivitas untuk proyek organik yang diperkirakan berjumlah 39.800 baris kode.
    Jawab:
    Sebuah proyek organik menggunakan formula aplikasi. Biaya = 2.4 x $(KDSI)^{1.05}$
    Biaya = 2.4 x $39.8^{1.05}$ = 2.4 x 47.85 = 114.8 programmer-bulan
    TDEV = 2.5 x $(PM)^{0.38}$ = 2.5 x 6.06 = 15.15 bulan
    Biaya = staf rata-rata/TDEV = 114.8/15.15 = 7.6 programmer
    Produktivitas = 39.800 LOC/(114,8 PM x 20 hari/bulan) = 17.3 LOC/programmer-hari

11. Hitung titik-titik fungsi yang belum disesuaikan untuk uraian soal pada Soal 2.
    Jawab:

Jenis	**Contoh**	**Rata-rata**	**Kompleks**	**Total**
Input	Nama Pasein Apoteker Selesai Tujuan Apoteker	Batalkan Aplikasi		13
Output	Komentar	Kalender Detail pendukung Informasi aplikasi Daftar Pemberitahuan	Jadwal harian Jadwal Mingguan	38
Kebutuhan	Kueri Berdasarkan nama Kueri berdasarkan Tanggal	Verifikasi Pasien Periksa Kalender Aplikasi yang tersedia		18
Files		Data Pasien		10
Interface				
Total				79

BAB 5
LATIHAN SOAL

1. Jelaskan mengapa contoh tinggi badan memenuhi kriteria metrik yang valid.
   Jawab:
   Kriteria 1: Orang yang berbeda dapat memiliki tinggi badan yang berbeda.
   Kriteria2: Dua orang mana pun yang kami rasa mempunyai tinggi badan yang sama, maka tinggi badan mereka akan sama.
   Kriteria 3: Setiap tambahan tinggi badan seseorang sesuai dengan peningkatan numerik tinggi badan.
   Kriteria 4: Orang yang berbeda bisa saja mempunyai tinggi badan yang sama.
2. Sebuah penelitian terhadap anak-anak sekolah dasar menemukan korelasi yang tinggi antara ukuran sepatu dan kemampuan membaca. Apakah ini berarti ukuran sepatu merupakan ukuran kecerdasan yang baik?
   Jawab:
   Tidak, ukuran sepatu berkorelasi baik dengan usia. Usia berkorelasi baik dengan kemampuan membaca. Keduanya bukanlah ukuran yang baik, karena keduanya tidak memenuhi syarat representasi.
3. Jelaskan mengapa uang merupakan ukuran skala rasio dan bukan sekedar skala interval.
   Jawab:
   Uang memiliki gagasan nol yang dipahami dengan baik. Di semua sistem moneter, ada angka nol dan angka ini setara dengan nol di semua sistem lainnya. Intervalnya sudah ditentukan. Jadi, jika Anda memiliki dolar AS dua kali lebih banyak daripada saya, jumlahnya akan tetap dua kali lipat jika kita mengonversikan kedua uang tersebut ke dalam pound Inggris (dengan asumsi tidak ada penalti konversi dan menggunakan nilai tukar yang sama).
4. Jelaskan mengapa IPK tidak sesuai dengan teori pengukuran.
   Jawab:
   Agar nilai dapat dirata-ratakan, maka nilai harus berupa skala interval. Ini menyiratkan bahwa perbedaan antara nilai-nilai dapat dibandingkan. Artinya, selisih antara A dan B sama dengan selisih antara D dan F. Biasanya, hal ini bahkan tidak dipertimbangkan saat mengalokasikan nilai.
5. Mengapa kompleksitas tidak mudah diukur?
   Jawab:
   Kompleksitas tidak didefinisikan dengan baik. Ada banyak aspek kompleksitas, dan setiap orang mungkin memiliki penafsiran yang sedikit berbeda mengenai apa yang kompleks. Faktanya, kompleksitas dapat dianggap sebagai interaksi antara seseorang dan kode.
6. Apa keuntungan memiliki tatanan parsial pada sistem hubungan empiris?
   Jawab:

Sistem hubungan empiris harus mempunyai hubungan yang diterima. Seringkali lebih mudah untuk menemukan sebagian perintah yang diterima dengan baik, sedangkan menemukan kesepakatan dalam semua kasus mungkin sulit.

7. Mengapa jumlah keputusan ditambah 1 merupakan metode penting untuk menghitung bilangan siklomatik McCabe?

   Jawab:

   Akan sangat memakan waktu jika harus membuat grafik aliran kendali untuk program besar.

8. Mengapa monotonisitas merupakan karakteristik penting dari metrik ukuran atau upaya seperti metrik upaya Halstead?

   Jawab:

   Jika menambahkan lebih banyak kode dapat menyebabkan nilai metrik upaya menurun, maka perilaku metrik tersebut tidak dapat dimengerti. Ini mungkin juga berarti bahwa metrik dapat dimanipulasi.

Latihan Soal Tambahan (Diskusi)

1. Identifikasi skala yang tepat untuk setiap ukuran berikut:

   LOC
   Bilangan siklomatik McCabe
   Rata-rata kedalaman sarang
   Kedalaman sarang maksimum

   Jawab:

LOK	Mutlak
bilangan siklomatik McCabe	Selang
Kedalaman sarang rata-rata	Urut
Kedalaman bersarang maksimum	Urut

2. Tunjukkan bahwa skala suhu Celcius dan Fahrenheit merupakan skala interval yang menggunakan suhu Celcius 20, 30, dan 40 derajat.

   Jawab:

   Setara Fahrenheit adalah 68, 86, dan 104 derajat. Perbedaan antara suhu terendah dan menengah adalah 10 Celsius dan 18 Fahrenheit. Perbedaan antara bagian bawah dan atas adalah dua kali lipatnya, yaitu 20 Celcius dan 36 Fahrenheit.

3. Tunjukkan bahwa bilangan siklomatik McCabe memenuhi teori pengukuran representasional.

   Jawab:

   Untuk sistem empiris, pertimbangkan himpunan semua grafik aliran kendali. Hubungannya adalah bahwa satu CFG lebih kecil atau sama dengan CFG kedua jika CFG kedua dapat dibuat dari CFG pertama dengan menambahkan node dan busur.

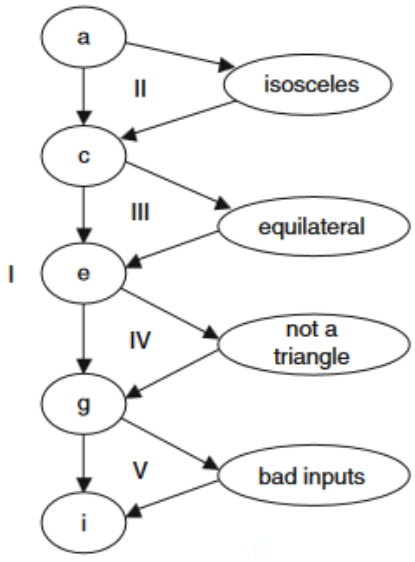
   Sistem numerik (Answer Set) dapat berupa bilangan bulat. Relasi pada bilangan bulat merupakan standar yang kurang dari atau sama dengan.

Pemetaannya rumusnya $e - n + 2$. Operasinya hanya ada dua, yaitu penjumlahan node dan penjumlahan busur. Menambahkan busur berarti meningkatkan nilai e. Menambahkan node berarti menambahkan node pada busur. Artinya e dan n bertambah 1, sehingga nilainya tetap sama. Jadi, untuk dua CFG x dan y, jika x lebih kecil dari y, maka y dapat dibuat dari x dengan menambahkan busur dan titik. Oleh karena itu, nilai pemetaan harus meningkat atau tetap sama. Oleh karena itu, kondisi representasi yang tidak terlalu ketat terpenuhi. (Kondisi representasi yang lebih ketat tidak dapat dipenuhi, karena urutan pada CFG adalah urutan parsial).

4. Tunjukkan bahwa bilangan siklomatik McCabe merupakan suatu ukuran skala interval.
   Jawab:
   Karena bilangan siklomatik McCabe lebih banyak satu daripada banyaknya keputusan, maka setiap interval dalam bilangan siklomatik disebabkan oleh banyaknya keputusan tambahan tersebut. Jadi selisih 2 dan 3 sama dengan selisih 10 dan 11. Ini bukan ukuran skala rasio karena tidak ada angka nol yang jelas. Faktanya, bilangan siklomatik tidak boleh nol.
5. Hitung bilangan siklomatik McCabe pada kode sumber berikut. Gambarlah grafik aliran kendali. Labeli daerah tersebut dengan angka Romawi.

```
read x,y,z;
type=`` scalene'';
if (x == y or x == z or y == z) type="isosceles'';
if (x == y and x == z) type ="'equilateral'';
if (x >= y+z or y >= x+z or z >= x+y) type ="'not a triangle''
if (x <= 0 or y <= 0 or| z <= 0) type ="'bad inputs'';
print type;
```

Jawab:



Banyaknya daerahnya 5, jadi bilangan siklomatisnya 5. Bisa juga dihitung dengan keputusan. Ada keputusan tentang cara keluar dari node a, c, e, dan g. Jadi, ada 4 keputusan. Bilangan

siklomatis adalah banyaknya keputusan ditambah 1 sehingga menjadi 5. Dapat juga dihitung dengan rumus $e - n + 2$. Disini $e = 12$ dan $n = 9$, jadi $e - n + 2 = 5$.

BAB 6

LATIHAN SOAL

1. Mengapa manajemen risiko itu penting?

   Jawab: Risiko dapat dikelola dan dampak risiko dapat diminimalkan. Namun, meminimalkan risiko mengharuskan Anda mengidentifikasi dan mengelola risiko.

2. Pertimbangkan berkendara ke bandara untuk naik pesawat dari maskapai penerbangan yang belum pernah Anda gunakan sebelumnya. Resiko apa yang mungkin unik dalam perjalanan ke bandara ini, dan risiko apa saja yang dapat dikelola sebagai bagian dari perjalanan normal ke bandara?

   Jawab:

   Risiko normal—Kehabisan bahan bakar, ban kempes, keterlambatan cuaca, kecelakaan lalu lintas, lupa koper

   Risiko unik—Pembangunan jalan raya menuju bandara, kemungkinan terminal berbeda, penundaan check-in khusus untuk maskapai penerbangan ini

Latihan Tambahan (Diskusi)

1. Analisis potensi risiko masalah klinik gigi di Bab 4, Masalah 2. Klasifikasikan risiko sebagai normal atau unik untuk proyek ini.

   Jawab:

   Risiko normal—Kesalahpahaman terhadap keinginan pengguna, miskomunikasi dengan pengguna, masalah perangkat keras pengguna, pembengkakan biaya, penundaan proyek, dan sebagainya

   Risiko unik—Berinteraksi dengan sistem rekam medis pasien

2. Pertimbangkan sebuah proyek yang mempunyai kemungkinan 0,5 persen untuk kesalahan yang tidak terdeteksi sehingga perusahaan harus membayar denda sebesar Rp. 1.000.000.000. Hitung eksposur risiko.

   Jawab:

   Eksposur risiko merupakan penjumlahan dari eksposur risiko untuk setiap kemungkinan.

   0.005 x 1.000.000.000 + 0.995 x 0 = Rp 5.000.000

3. Pertimbangkan penggunaan tinjauan tambahan untuk Soal 2 yang memerlukan biaya Rp. 1.000.000 namun menghilangkan kesalahan tersebut sebanyak 50 persen. Hitung eksposur risiko baru ini dengan menggunakan tinjauan tambahan. Apakah pendekatan tinjauan tambahan lebih baik?

   Jawab:

   0.0025 x 1.000.000 + 0.9975 x 1.000.000 = Rp. 3.500.000

4. Apa yang akan berubah pada Soal 3 jika tinjauan tambahan hanya efektif 10 persen saja?

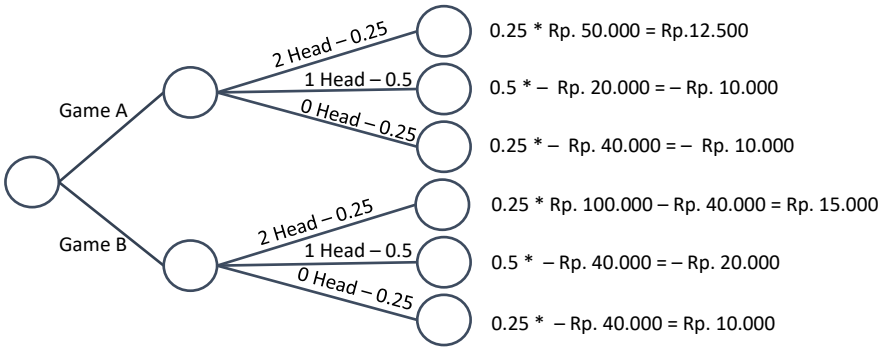
   Jawab:

   0.0045 x 100,100 + 0.9955 x 1.000.000 = Rp. 5.500.000

5. Buatlah pohon keputusan untuk permasalahan pada Contoh 6.3 jika pada Game A, bayarannya adalah Rp. 50.000 dan jika biaya bermain di Game B adalah Rp. 40.000. Haruskah Anda memainkan salah satu game tersebut?

   Jawab:

   Pohon keputusan risiko ditunjukkan pada Gambar dibawah ini Di kedua permainan Anda pasti akan kehilangan uang. Di game A Anda akan kehilangan rata-rata Rp. 7.500, dan di Game B Anda akan kehilangan rata-rata Rp. 15.000.



6. Perusahaan X mempunyai data historis yang menunjukkan tingkat kesalahan normal sebesar 0,0036 kesalahan per KLOC. Sebuah studi tentang teknik tinjauan baru menunjukkan bahwa biayanya Rp. 10.000.000 per 100 KLOC dan mengurangi jumlah kesalahan sebesar 50 persen. Asumsikan bahwa setiap kesalahan merugikan perusahaan rata-rata sebesar Rp. 10.000.000. Proyek saat ini diperkirakan berukuran 50 KLOC. Hitung eksposur risiko untuk setiap pendekatan. Apakah teknik review baru ini layak dilakukan?

   Jawab:

   Kasus 1—Tidak ada tinjauan

   0.0036 x 50 KLOC x Rp. 100.000.000 = Rp. 18.000.000

   Kasus 2—Dengan tinjauan

   0.0018 x 50 KLOC x Rp. 100.000.000 = Rp. 14.000.000

   Ya, lebih baik melakukan review.

BAB 7

LATIHAN SOAL

No	Soal	Jawaban
1	Apa perbedaan antara tinjauan dan tinjauan teknis formal?	Inspeksi atau tinjauan teknis formal (FTR) memerlukan produk yang jelas dan lengkap. Produser harus menjadi peserta aktif dalam

		peninjauan/pemeriksaan. Inspeksi/FTR harus menjadi bagian dari proses perangkat lunak yang ditentukan. Tujuan utamanya adalah untuk menemukan cacat. Inspeksi harus mengikuti proses yang ditentukan dengan peran dan langkah yang spesifik.
2	Bagaimana cara mengevaluasi daftar periksa?	Gunakan dalam meninjau kode/dokumen perangkat lunak. Catat permasalahan yang teridentifikasi pada setiap item daftar periksa. Catat kesalahan yang ditemukan setelah item berhasil ditinjau.
3	Revisi checklist apa saja yang harus dilakukan?	Item yang tidak terkait dengan deteksi kesalahan harus dihapus. Kesalahan yang terdeteksi pada fase selanjutnya dari siklus hidup harus digunakan untuk menghasilkan item daftar periksa baru.
4	Faktor-faktor apa yang mempengaruhi efektivitas tinjauan teknis formal?	Waktu persiapan dan waktu peninjauan.
5	Bagaimana efektivitas tinjauan teknis formal diukur?	Biasanya dalam tingkat penemuan kesalahan. Hal ini dapat diukur dalam jam penemuan kesalahan/KLOC dan jam penemuan kesalahan/peninjau.
6	Apa yang terjadi jika produsen tidak dapat menyelesaikan suatu permasalahan?	Jika manajer tidak dapat menyelesaikan masalah, mungkin akan dilakukan pemeriksaan ulang melalui semua tahapan.
7	Apa yang terjadi jika satu atau lebih anggota tim inspeksi tidak setuju dengan mayoritas?	Kelompok minoritas akan membuat laporan minoritas yang memaparkan pandangan mereka.
8	Jika sebuah dadu jujur dilempar sebanyak 5 kali, berapa peluang munculnya angka 6 pada salah satu pelemparan?	$(5/6)^5 = 0.4018$
9	Jika sebuah dadu jujur dilempar 5 kali, berapakah peluang munculnya angka 6 pada paling sedikit salah satu pelemparan?	$1 - (5/6)^5 = 1 - 0.4018 = 0.5982$

Latihan Tambahan (Diskusi)

1. Buatlah daftar periksa untuk meninjau kode C++.

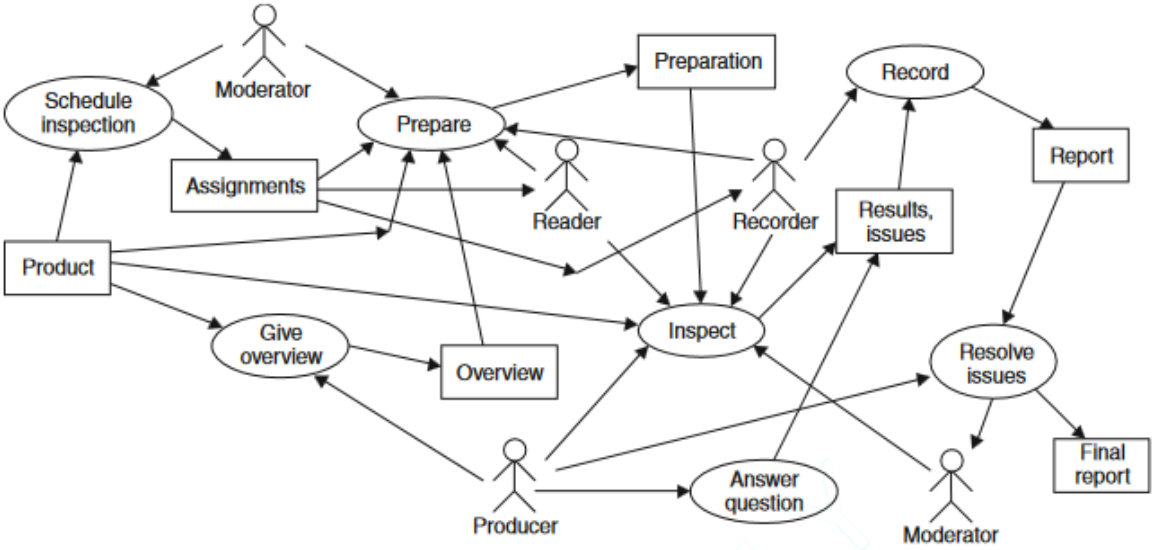
   Jawab:

   1. Apakah semua pointer diinisialisasi di konstruktor?
   2. Apakah semua variabel dideklarasikan?
   3. Apakah setiap ‘‘{’’ memiliki ‘‘}’’ yang cocok?
   4. Apakah setiap perbandingan kesetaraan mempunyai tanda ganda ‘‘=’’?
   5. Apakah ada kondisi while atau if yang ditutup dengan tanda '';''?
   6. Apakah setiap deklarasi kelas diakhiri dengan ‘‘;’’?

Sebelumnya adalah contoh item daftar periksa yang saya kembangkan berdasarkan pengalaman pribadi membuat kesalahan C++. Secara khusus, saya telah menghabiskan waktu lama untuk menemukan kesalahan yang disebabkan oleh item 5.

2. Buatlah daftar periksa untuk meninjau desain perangkat lunak.
   Jawab:
   1. Apakah semua fungsi penting ditampilkan dalam desain?
   2. Apakah semua atribut penting dispesifikasikan dalam desain?
   3. Apakah semua nama berkaitan dengan tujuan dan jenisnya serta tidak ambigu?
   4. Apakah semua hubungan antar kelas dispesifikasikan?
   5. Apakah semua fungsi memiliki data yang diperlukan agar fungsi tersebut dapat dijalankan?

3. Gambarkan model proses untuk inspeksi formal.
   Jawab:



4. Dengan asumsi bahwa pengujian tersebut mewakili situasi operasional, hitung keandalan sistem perangkat lunak yang memiliki 10 kesalahan dalam 200 kasus pengujian.
   Jawab:

$$\boldsymbol{F(1) = 10/200 = 0.05}$$
$$\boldsymbol{R(1) = 0.95}$$

5. Asumsikan teknik FTR A membutuhkan 2 jam/persiapan KLOC dan memberikan waktu 1 jam/waktu review KLOC dan teknik FTR B memerlukan 1 jam/waktu persiapan KLOC dan 4 jam/waktu review KLOC. Asumsikan juga bahwa dalam eksperimen terkontrol dengan kode sumber yang sama, teknik A menemukan 12 kesalahan/KLOC dan B menemukan 14 kesalahan/KLOC. Bandingkan kedua teknik untuk efektivitasnya.

Jawab:

Teknik A memakan 80 persen waktu teknik B dan menemukan 85 persen kesalahan yang ditemukan B. Jadi A sedikit lebih efisien daripada B. Dari sudut pandang perbaikan marjinal, diperlukan waktu 2 jam/KLOC upaya lebih banyak untuk menemukan 2 kesalahan terakhir/KLOC.

6. Jika perangkat lunak mengalami 5 kegagalan dalam 100 pengujian selama 10 hari pengujian, berapa perkiraan yang baik mengenai keandalan perangkat lunak pada hari berikutnya? sepekan?

   Jawab:

   $$\mathrm{F}(1) = 0.05;\ \mathrm{R}(1) = 0.95$$

   Hari berikutnya?

   Asumsikan 10 tes per hari (rata-rata dari 10 hari terakhir).

   $$\mathrm{R}(10) = \mathrm{R}(1)^{10} = 0.95^{10} = 0.598$$

   Minggu depan? Asumsikan 70 Tes

   $$\mathrm{R}(10) = \mathrm{R}(1)^{70} = 0.95^{70} = 0.027$$

7. Mengembangkan rencana SQA untuk masalah B&B (Bab 4, Masalah 3).

   Bagian 1. Tujuan

   XYZ Corporation sedang mengembangkan perangkat lunak untuk Tom and Sue's Bed and Breakfast Inn. Perangkat lunak ini dimaksudkan untuk menyimpan informasi reservasi dan membantu dalam memantau pengeluaran dan keuntungan. Paket SQA ini adalah versi 1.0 yang akan dikirimkan ke Tom dan Sue paling lambat tanggal 1 November.

   Rencana SQA ini mencakup siklus hidup pengembangan sistem yang lengkap mulai dari spesifikasi kebutuhan hingga pengujian perangkat lunak.

   Bagian 2. Dokumen Referensi

   Pernyataan Kerja, versi B&RSOW1.0 tanggal 1/1

   Standar Pengkodean Perusahaan XYZ, versi 4.6, tertanggal 7/8/95

   Bagian 3. Manajemen

   3.1 Organisasi

   Pimpinan proyek (Tom)

   Kelompok pengembangan (Bill, Jane, Fred)

   Grup SQA (John)

   3.2 Tugas

   Analisis dan spesifikasi persyaratan

   Perencanaan proyek

   Perkiraan biaya

Desain arsitektur
Desain tingkat rendah
Penerapan
Pengujian
Pemantauan proyek
Inspeksi
Ulasan
Dokumentasi

3.3 Tanggung jawab

Pimpinan Proyek—Tom

Tanggung jawab: Perencanaan proyek, estimasi biaya, pemantauan proyek, persetujuan semua ringkasan laporan dan rencana

Persyaratan Pimpinan—Bill

Tanggung jawab: Analisis dan spesifikasi persyaratan
Anggota tim: Jane, Fred

Desain—Jane

Tanggung jawab: Implementasi desain arsitektur

Pemimpin—Jane

Tanggung jawab: Desain dan implementasi tingkat rendah
Anggota tim: Fred, Bill

Pimpinan Tes—Bill

Tanggung jawab: Semua laporan pengujian dan pengujian
Anggota tim: Fred

Dokumentasi—Jane

Tanggung Jawab: Panduan pengguna

SQA—John

Tanggung jawab: Melakukan semua peninjauan, penelusuran, dan inspeksi, peninjauan semua dokumen dan laporan, pelacakan semua laporan masalah, dan penyerahan laporan masalah mingguan

Bagian 4. Dokumentasi

Spesifikasi Kebutuhan Perangkat Lunak
Perancangan Perangkat Lunak Menggunakan UML dan OCL
Rencana Uji Perangkat Lunak
Panduan Pengguna Laporan
Pengujian Perangkat Lunak

Setiap dokumen akan direview pada versi draft dan diperiksa pada versi final. Pimpinan SQA (John) akan bertanggung jawab untuk melakukan semua tinjauan dan inspeksi.

Bagian 5. Standar, Praktik, Konvensi, dan Metrik

Standar Pengkodean Perusahaan XYZ akan digunakan. Tim SQA akan melakukan inspeksi terhadap seluruh kode untuk memastikan kepatuhan. Laporan kepatuhan akan diserahkan kepada pimpinan proyek.

Titik fungsi yang tidak disesuaikan akan dihitung pada semua komponen. LOC akan dihitung pada semua kelas.

Bagian 6. Peninjauan dan Audit

Ulasan berikut akan dilakukan. Pimpinan SQA akan melakukan setiap tinjauan dan menyerahkan laporan kepada pimpinan proyek untuk disetujui.

Ulasan:

Tinjauan Desain Awal Persyaratan Perangkat Lunak

Panduan Inspeksi Kode Desain Setiap Kelas

Bagian 7. Tes

Pengujian akan diselesaikan sesuai dengan rencana pengujian yang akan dikembangkan oleh tim SQA dan disetujui oleh pimpinan proyek.

Bagian 8. Pelaporan Masalah

Semua masalah akan dilaporkan kepada pimpinan yang sesuai. Jika sesuai, pimpinan akan menyerahkan laporan masalah kepada pimpinan SQA, yang akan memasukkan laporan masalah tersebut ke dalam sistem pelacakan masalah. Penyelesaian masalah akan dilaporkan kepada pimpinan SQA. Laporan pelacakan masalah mingguan akan diserahkan kepada pimpinan proyek.

Bagian 9. Alat, Teknik dan Metodologi

Tidak ada.

Bagian 10. Pengendalian Kode

Sistem manajemen konfigurasi XYZ Corporation akan digunakan.

Bagian 11. Pengendalian Media

T/A

Bagian 12. Pengendalian Pemasok (untuk outsourcing)

T/A

Bagian 13. Catatan—Pengumpulan, Pemeliharaan, dan Penyimpanan

T/A

Bagian 14. Pelatihan

T/A

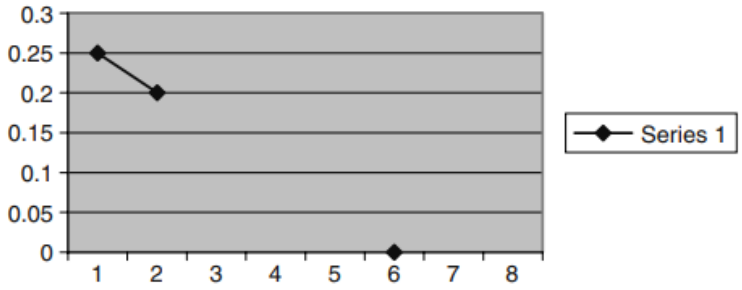
Bagian 15. Manajemen Risiko

T/A

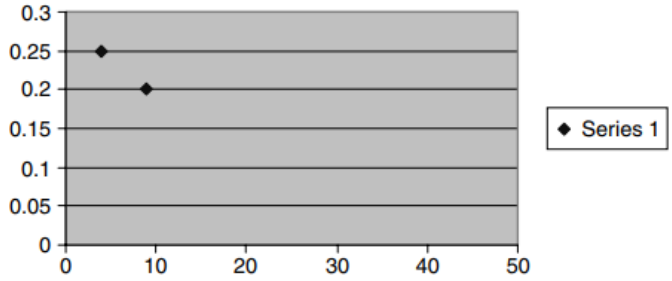
8. Kesalahan 1 terjadi setelah 4 hari, dan kesalahan 2 terjadi 5 hari kemudian. Plot grafik tingkat kesalahan versus angka kesalahan dan grafik tingkat kesalahan versus waktu, dan perkirakan jumlah kesalahan dalam sistem dan waktu untuk menghilangkan semua kesalahan.

   Jawab:

   Seperti yang ditunjukkan pada gambar dibawah ini, memplot tingkat kesalahan versus jumlah kesalahan menunjukkan intersep sekitar 6. Jadi, ada 4 kesalahan yang tersisa dalam sistem.



   Seperti yang ditunjukkan pada gambar dibawah ini, memplot tingkat kesalahan versus waktu yang berlalu menunjukkan intersep sekitar 29 hingga 30. Dengan demikian, diperlukan sekitar 20 unit lagi untuk menghilangkan kesalahan sepenuhnya.



BAB 8

LATIHAN SOAL

1. Apa kelebihan DFD dibandingkan diagram lainnya?

   Jawab: Aliran data melalui sistem dan data spesifik yang tersedia untuk setiap proses ditampilkan dengan jelas.

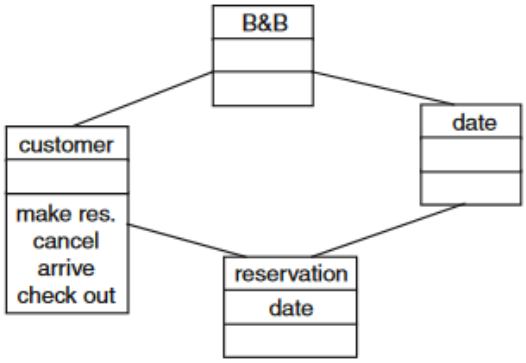
2. Apa tujuan spesifikasi perilaku dan diagram dalam spesifikasi kebutuhan?

   Jawab: Tujuannya adalah untuk mengkomunikasikan gambaran sistem yang diusulkan kepada pengguna untuk memastikan pemahaman bersama tentang keseluruhan perilaku sistem yang diusulkan.

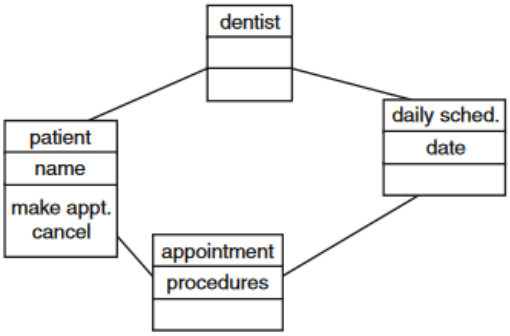
3. Fungsi apa yang penting untuk disertakan dalam diagram use case?
   Jawab: Fungsi-fungsi penting adalah fungsi-fungsi penting yang menyampaikan fungsionalitas yang diperlukan.
4. Kriteria apa yang harus digunakan untuk mengevaluasi skenario?
   Jawab: Setiap rangkaian fungsi yang signifikan perlu ditampilkan dari sudut pandang pengguna.
5. Kriteria apa yang harus digunakan untuk mengevaluasi diagram keadaan?
   Jawab: Diagram keadaan perlu menunjukkan semua kemungkinan transisi. Setiap busur dalam diagram harus memiliki peristiwa yang menyebabkan transisi. Juga harus ada jalur dari node awal ke setiap node dan dari setiap node ke node terminal.
6. Apa keuntungan dari diagram sistem?
   Jawab: Karena kurang formal, maka bisa lebih fleksibel dalam mengekspresikan ide-ide tentang sistem secara keseluruhan.
7. Apa masalah utama pada diagram sistem?
   Jawab: Kurangnya formalitas dapat menyebabkan diagram ambigu dimana satu simbol digunakan untuk ide yang berbeda.

Latihan Tambahan (Diskusi)

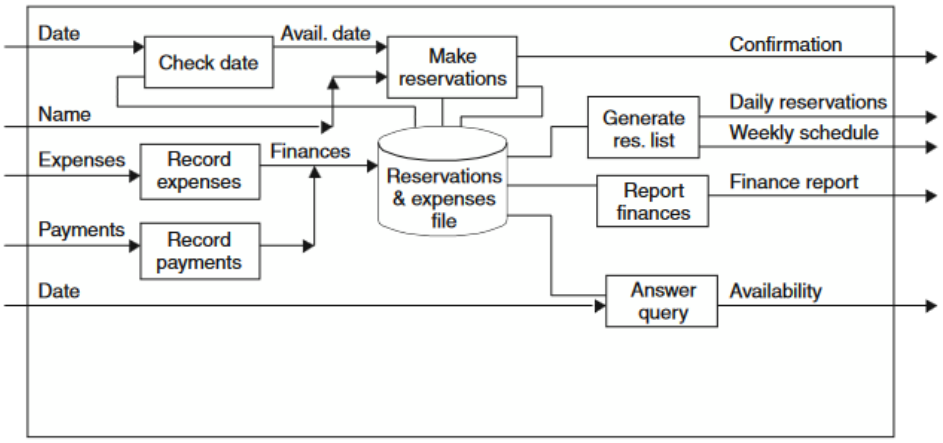
1. Gambarkan model objek untuk permasalahan B&B (lihat Soal 4.3).
   Jawab:



2. Gambarkan model objek untuk permasalahan kantor gigi (lihat Soal 4.2).
   Jawab:

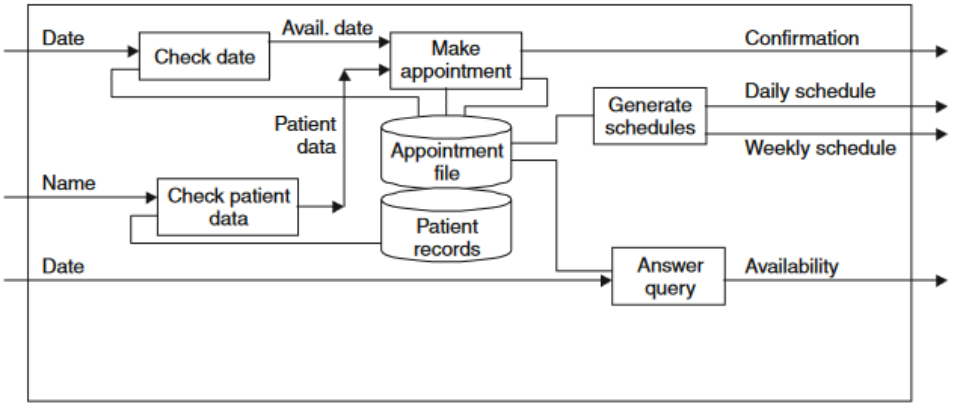


3. Gambarkan DFD untuk sistem B&B (lihat Soal 4.3).
   Jawab:



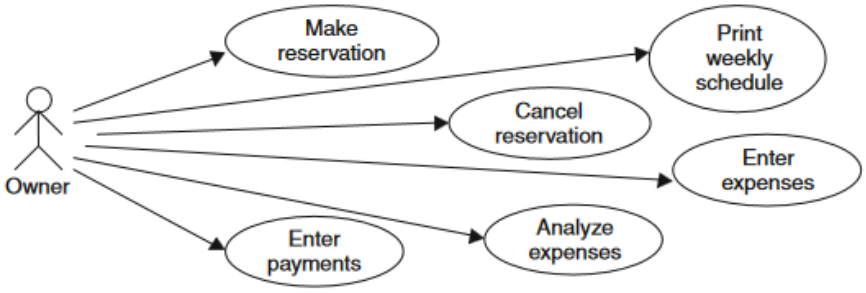
4. Gambarlah DFD untuk sistem kantor gigi (lihat Soal 4.2)

   Jawab:



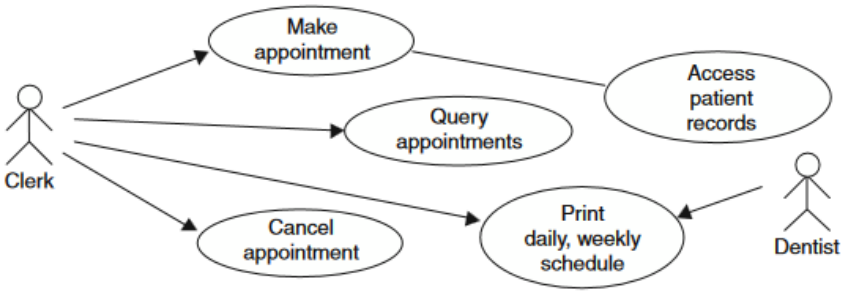
5. Gambarkan diagram use case untuk masalah B&B (lihat Soal 4.3).

   Jawab:



6. Gambarkan diagram use case untuk permasalahan klinik gigi (lihat Soal 4.2).

   Jawab:



7. Tuliskan skenario untuk masalah B&B (lihat Soal 4.3).

Jawab:

Panggilan Pelanggan 1:

Pelanggan menelepon tentang ketersediaan pada tanggal yang ditentukan.
Sue menampilkan kalender untuk minggu itu.
Ada lowongan.
Sue mengutip harga.
Sue mendapatkan nama, alamat, nomor telepon, dan nomor kartu kredit.
Sue memasukkan informasi.
Pelanggan memberikan kartu kredit untuk menjamin reservasi.

Panggilan Pelanggan 2:

Pelanggan menelepon tentang ketersediaan pada tanggal yang ditentukan.
Sue menampilkan kalender untuk minggu itu.
Tidak ada lowongan.

Panggilan Pelanggan 3:

Pelanggan menelepon tentang ketersediaan pada tanggal yang ditentukan.
Sue menampilkan kalender untuk minggu itu.
Ada lowongan.
Sue mengutip harga.
Sue mendapatkan nama, alamat, nomor telepon, dan nomor kartu kredit.
Sue memasukkan informasi.
Pelanggan menjamin reservasi. Pesan berdasarkan tanggal berlalu.
Pelanggan lain meminta tanggal tersebut.
Reservasi tanpa jaminan dihapus.

8. Tuliskan skenario untuk permasalahan klinik gigi (lihat Soal 4.2).

Jawab:

1—Biasa

Seorang pasien meminta janji. Nama pasien dikenali oleh sistem. Sistem menyarankan waktunya. Pasien menerima waktu itu, dan resepsionis memasuki janji temu. Dua hari sebelum janji temu, resepsionis mendapat daftar pengingat berisi

nama dan nomor telepon pasien. Resepsionis menelepon untuk mengingatkan pasien. Pasien datang untuk membuat janji. Setelah janji temu, asisten gigi menjadwalkan janji temu berikutnya dengan pasien.

2—Pasien Baru

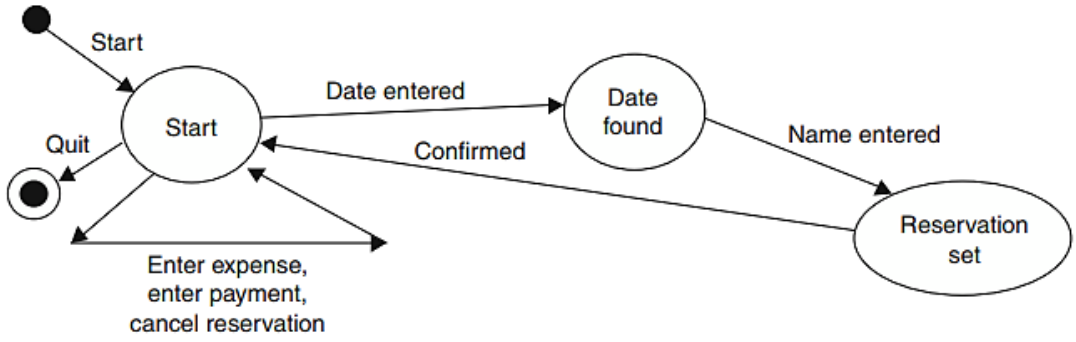
Seorang pasien meminta janji. Nama pasien tidak dikenali oleh sistem. Pasien harus dimasukkan ke dalam sistem pencatatan pasien.

3—Beberapa Janji

Seorang pasien menelepon dan ingin membuat janji temu 6 bulan untuk 2 tahun ke depan. Resepsionis memasukkan namanya ke dalam sistem dan, ketika diterima, memasukkan janji temu yang telah disepakati.

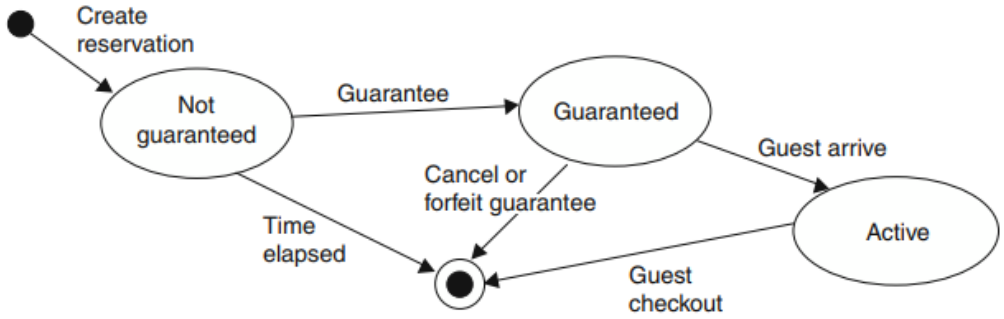
9. Gambarlah diagram keadaan untuk permasalahan B&B secara keseluruhan (lihat Soal 4.3).

Jawab:



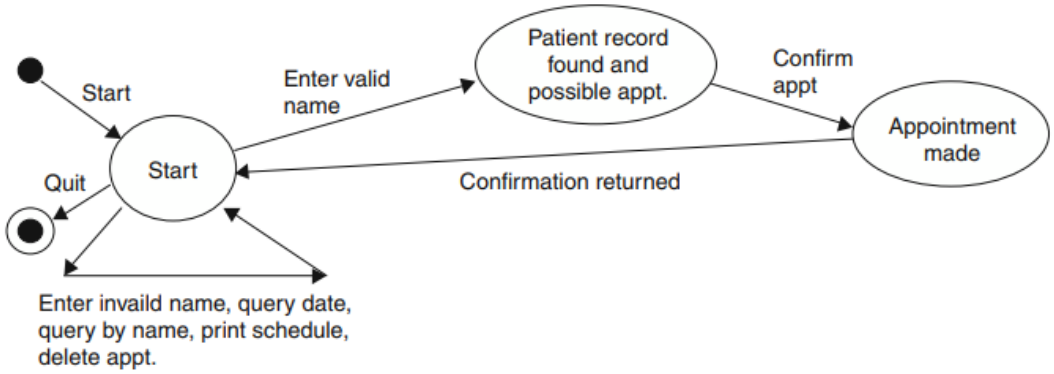
10. Gambarkan diagram keadaan untuk reservasi item data pada permasalahan B&B (lihat Soal 4.3).

Jawab:



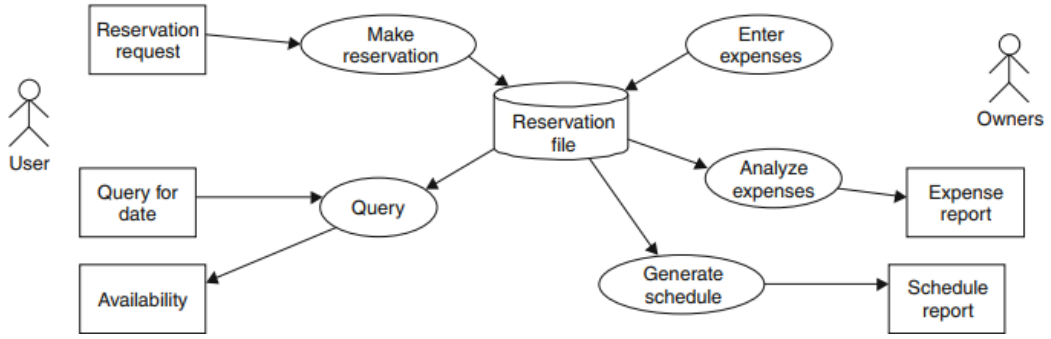
11. Gambarlah diagram keadaan untuk permasalahan kantor gigi (lihat Soal 4.2).

Jawab:



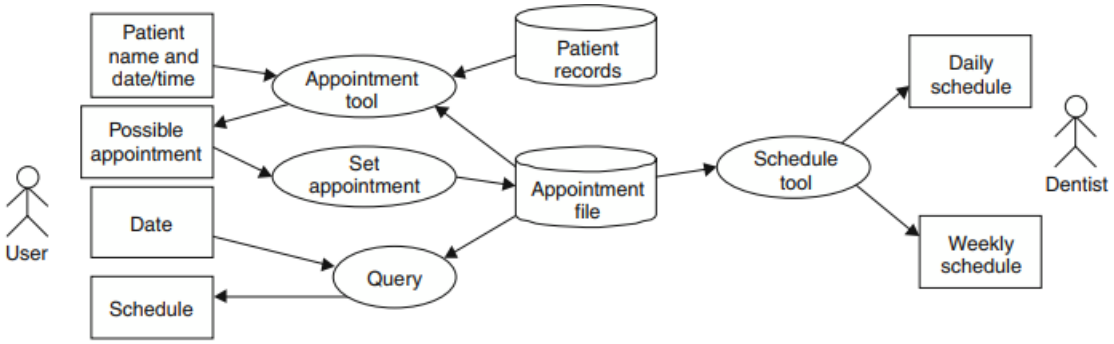
12. Gambarlah diagram sistem untuk sistem B&B (lihat Soal 4.3).

Jawab:



13. Gambarlah diagram sistem untuk sistem kantor gigi (lihat Soal 4.2).

Jawab:



BAB 9

LATIHAN SOAL

1. Dengan adanya desain berikut, tunjukkan ide penggandengan, kohesi, dan abstraksi berikut apakah diinginkan memiliki nilai tinggi atau rendah dan bagaimana desain ini mencontohkan istilah tersebut.

```
Class college
     student* stulist [MAX]
     course* courselist [MAX]
public:
```

```
        addStudent (char* studentname)
        addStudentToCourse (char* studentname, char*
        coursename)
        void displayStudent (char* studentname)
        void displayStudentsInCourse (char* coursename)
    Class course
        student* classroll [MAX]
    public:
        void displayStudents ()
    Class student
        char* name
    public:
        void displayname ()
```

Jawab:

Kopling—Kopling rendah diinginkan, dan desain ini memiliki kopling rendah, karena kelas perguruan tinggi tidak memiliki pengetahuan tentang internal kelas lain dan kelas lain tidak perlu mengetahui tentang perguruan tinggi.

Kohesi—Kohesi yang tinggi diinginkan, dan ini memiliki kohesi yang tinggi, karena setiap kelas hanya berurusan dengan atributnya sendiri.

Abstraksi—Desain menunjukkan abstraksi yang baik. Misalnya, metode tampilan di perguruan tinggi tidak menyertakan rincian apa pun tentang metode tampilan tingkat rendah

2. Mengapa Gunter membatasi istilah/kejadian yang dapat digunakan dalam suatu spesifikasi? Apa perbedaan antara kebutuhan pengguna dan spesifikasi?

   Jawab:

   Gunter mengatakan bahwa kebutuhan pengguna harus ditentukan dalam istilah/peristiwa yang diketahui oleh pengguna dan dapat mencakup istilah/peristiwa yang tersembunyi di mesin, sedangkan spesifikasi dirancang untuk menjadi dasar implementasi dan harus ditentukan hanya dalam istilah yang terlihat oleh mesin dan dunia.

3. Sistem yang diusulkan adalah pengenalan wajah berbasis pengolahan citra. Sistem tersebut akan memiliki kamera dan dimaksudkan untuk mencegah non-karyawan memasuki fasilitas rahasia perusahaan dengan mengontrol kunci pintu. Ketika seseorang mencoba memutar pegangan pintu, sistem mengambil gambar dan membandingkannya dengan sekumpulan gambar karyawan saat ini.

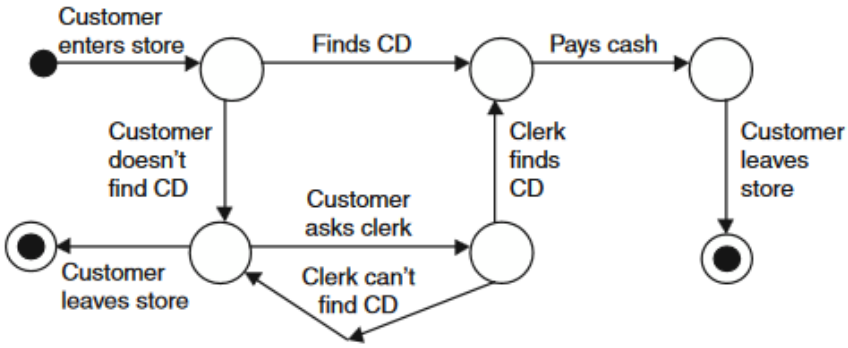
   Klasifikasikan masing-masing kejadian berikut, apakah kejadian tersebut berada di lingkungan atau di dalam sistem dan apakah kejadian tersebut tersembunyi atau terlihat:

   1. Seseorang mencoba memutar gagang pintu. **Lingkungan Terlihat**
   2. Pintu tidak terkunci oleh sistem. **Sistem Terlihat**

3. Seorang karyawan membiarkan seorang non-karyawan melewati pintu. **Lingkungan Tersembunyi**
4. Seorang karyawan mempunyai saudara kembar identik. **Lingkungan Tersembunyi**
5. Suatu gambar memiliki jumlah kesamaan minimal untuk algoritma pencocokan. **Sistem Tersembunyi**

Latihan Soal Tambahan (Diskusi)

1. Gambarkan skenario interaksi antara pelanggan yang mencoba membeli CD musik tertentu dengan uang tunai dan petugas di toko musik. Pastikan untuk mencakup semua kemungkinan. Gunakan model mesin negara dengan kejadian sebagai busur.
   Jawab:

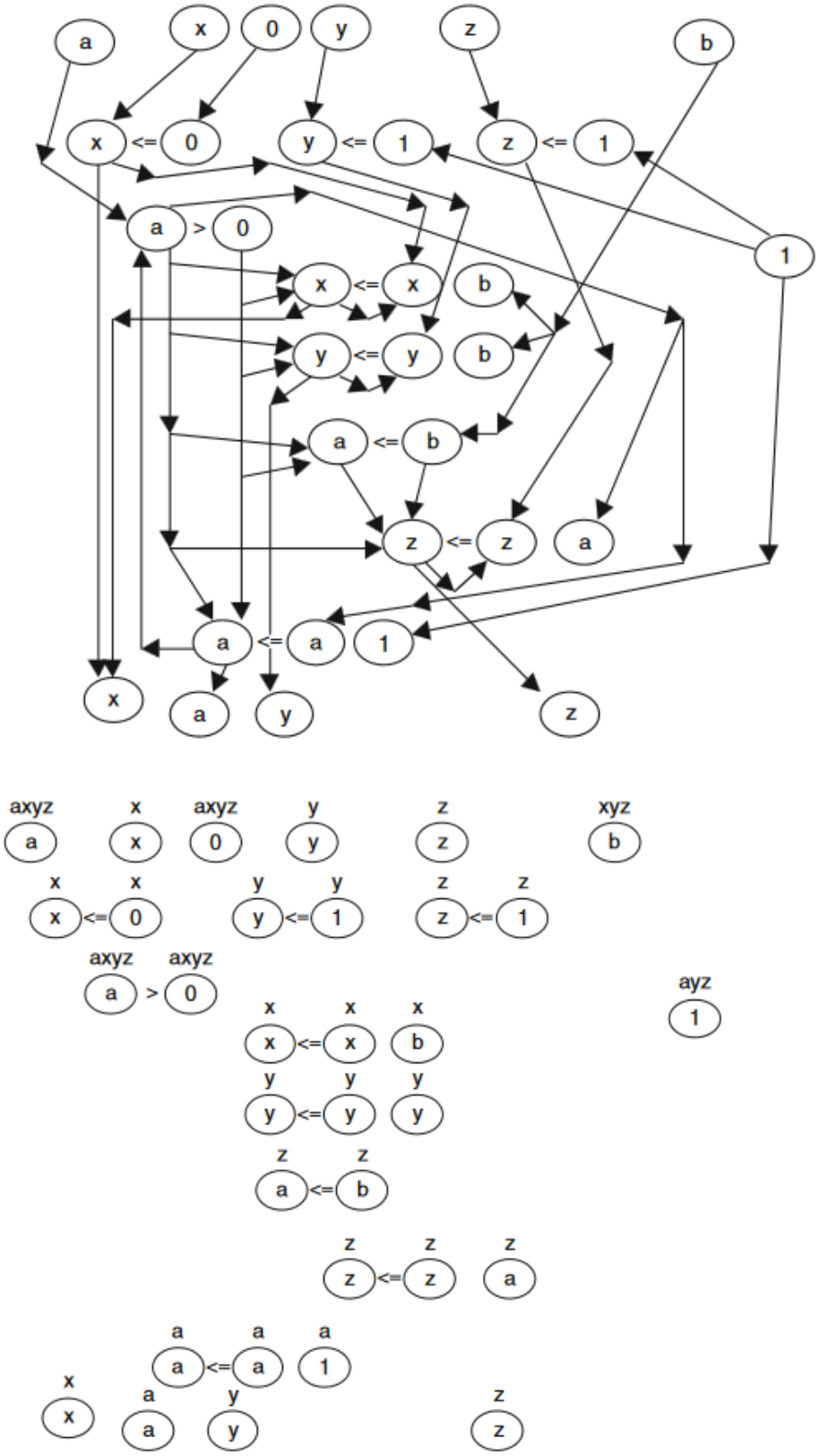


2. Hitung metrik kohesi fungsional Bieman dan Ott untuk segmen kode berikut. Gambarlah grafik berarah dan tunjukkan alirannya.

```
cin >> a >> b;
int x,y,z;
x=0; y=1; z=1;
while (a > 0) {
    x=x+b;
    z=z*b;
    if (a>b) {
        y=y*a;
    }
    a=a-1;
}
cout << x << a << z << y;
}
```

   Jawab:

   Lihat Gambar dibawah ini (Grafik aliran kontrol)



Ada 33 token. Empat adalah lem super. Enam (termasuk token lem super) adalah token lem. Kohesi fungsional lemah (WFC) adalah 6/33 = 18.2 persen. Kohesi fungsional kuat (SFC) adalah 4/33 = 12.1 persen. Daya rekatnya adalah (4 * 1 + 2 * 0.7 + 5 + 27 x 0.25/33 = 12.25/33 = 37.1 persen.

Program ini menghitung tiga besaran terpisah. Tidak mengherankan jika skornya rendah pada metrik kohesi.

BAB 10

LATIHAN SOAL

No	Soal	Jawaban
1	Apa saja permasalahan mendasar dalam pengujian perangkat lunak?	Permasalahan mendasarnya adalah bagaimana memilih kasus uji dan kapan harus menghentikan pengujian.
2	Mengapa diperlukan spesifikasi untuk melakukan pengujian?	Suatu spesifikasi diperlukan untuk memutuskan kapan perilaku aktual itu benar atau salah.
3	Mengapa pengujian jalur biasanya tidak praktis?	Sebagian besar program memiliki kemungkinan jalur yang tidak terbatas dalam program tersebut.
4	Apakah pengujian jalur mencakup cakupan pernyataan?	Ya, setiap pernyataan ada pada jalur tertentu. Jadi mencakup setiap jalur akan mencakup setiap pernyataan.
5	Penguji perangkat lunak terkadang mengatakan ''kesalahan terjadi di sudut-sudut''. Apa artinya ini?	Kesalahan cenderung lebih banyak terjadi pada batasan-batasan. Artinya, kesalahan dalam kode sumber sering kali mempengaruhi beberapa keputusan dan dengan demikian menghasilkan kesalahan pada batasannya.
6	Cakupan setiap pernyataan bukan merupakan kriteria pengujian subdomain. Apa pentingnya hal ini?	Saat menggunakan kriteria pengujian subdomain, Anda dapat dengan mudah meningkatkan cakupannya dengan memilih beberapa kasus pengujian dari setiap subdomain. Hal ini juga mudah untuk dianalisis. Hal ini tidak terjadi pada setiap liputan pernyataan.
7	Apa perbedaan profil operasional terminal tempat penjualan di toko diskon dengan terminal tempat penjualan di toko mewah?	Di toko diskon, akan ada lebih banyak item yang ditandai dengan harga kurang dari Rp. 100.000. Di toko yang mahal, akan ada harga yang lebih besar. Harga juga dapat dibulatkan ke jumlah berikutnya.
8	Pengembang perangkat lunak mungkin secara tidak sadar tidak menguji perangkat lunaknya secara menyeluruh. Mengapa kriteria cakupan pengujian dapat membantu?	Kriteria cakupan pengujian membantu memaksa penguji untuk menguji banyak bagian perangkat lunak yang berbeda.

Latihan Soal Tambahan (Diskusi)

1. Jika suatu program memiliki dua masukan bilangan bulat dan masing-masing dapat berupa bilangan bulat 32-bit, berapa banyak kemungkinan masukan yang dimiliki program ini?

   Jawab:

Setiap bilangan bulat 32-bit memiliki 2^{32} kemungkinan nilai. Jadi, sebuah program dengan dua masukan bilangan bulat akan memiliki 2^{64}kemungkinan masukan.

2. Jika suatu program mempunyai 2^{64} kemungkinan masukan dan satu pengujian dapat dijalankan setiap milidetik, berapa lama waktu yang diperlukan untuk mengeksekusi semua kemungkinan masukan tersebut?

   Jawab:

   Itu berarti 10^6 tes per detik, atau 8,64 10^{10} tes per hari. Itu setara dengan 3.139 * 10^{13} tes per tahun. Karena 2^{10} = 1024 kira-kira sama dengan 1000 = 10^3 , 2^{64} = $(2^{10})^{6.4}$ dan kira-kira sama dengan $(10^3)^{6.4}$ = $10^{19.2}$. Membagi $10^{19.2}$ dengan 10^{13} memberikan nilai lebih dari 10^5. Oleh karena itu, diperlukan waktu setidaknya 10^5 tahun untuk melakukan semua tes tersebut.

3. Program penggajian akan menghitung gaji kotor mingguan dengan memasukkan jumlah jam kerja dan upah saat ini. Seorang pekerja tidak boleh bekerja lebih dari 80 jam per minggu, dan upah maksimum adalah Rp. 50.000.000 per jam. Membangun tes fungsional.

   Jawab:

   Pengujian fungsional harus mencakup pengujian gaji normal dan lembur serta pengujian kondisi kesalahan.

Jam	Gaji	Keluaran yang diharapkan
30	Rp. 400.000	Rp. 12.000.000
60	Rp. 500.000	Rp. 35.000.000 (Asumsi lembur)
81	Rp. 500.000	Jam tidak Valid
20	Rp. 600.000	Gaji tidak Sah

4. Sebuah program menghitung luas segitiga. Inputnya adalah tiga set koordinat (x, y). Membangun tes fungsional.

   Jawab:

   Tes fungsional harus mencakup segitiga benar, non-segitiga, kondisi kesalahan, dan orientasi segitiga yang jelas.

Poin 1	Poin 2	Poin 3	Area yang diharapkan
1,1	1,5	5,1	8
1,1	1,5	1,10	Bukan segitiga
10,10	0,10	10,0	50
0,0	0,10	10,10	50
0,0	0,0	0,0	0

5. Sebuah program menerima dua waktu (dalam format 12 jam) dan mengeluarkan jumlah menit yang telah berlalu. Membangun tes fungsional.

   Jawab:

Uji fungsional harus mencakup pengujian yang berdurasi kurang dari 1 jam, lebih dari 1 jam, satu pengujian lebih dari 12 jam, pengujian yang membutuhkan waktu berjam-jam, dan pengujian dengan waktu terbalik.

Waktu mulai	Waktu Berhenti	Waktu yang diharapkan berlalu
10:00	10:40	0:40
21:57	23:40	1:43
03:00	21:15	18:15
01:50	03:40	1:50
03:00	07:24	4:24
17:00	04:00	Kesalahan

6. Rutinitas pencarian biner mencari daftar nama dalam urutan abjad dan mengembalikan nilai true jika nama tersebut ada dalam daftar dan mengembalikan nilai false jika tidak. Membangun tes fungsional.

   Jawab:

   Tes fungsional harus mencakup tes berikut ini:

   Nama depan dalam daftar
   Nama belakang dalam daftar
   Nama sebelum nama depan
   Nama setelah nama belakang
   Nama di tengah.
   Nama yang tidak ada dalam daftar tepat setelah nama depan
   Nama yang tidak ada dalam daftar tepat sebelum nama belakang

7. Untuk program penggajian pada Soal 10.3, kenali kondisinya dan buatlah matriks pengujian.

   Jawab:

Kondisi						
0 < jam < = 40	T	F	F	F	T	F
40 < jam < = 80	F	T	F	T	F	F
0 < Gaji < = 50	T	T	T	F	F	F
Jam	30	50	90	50	30	- 5
Gaji	50	30	30	60	70	- 5
Output yang diharapkan	1500	1650	Error	Error	Error	Error

8. Untuk perhitungan luas segitiga pada Soal 10.4, kenali kondisinya dan buatlah matriks ujinya.

   Jawab:

Kondisi		
Poin Kolinear	F	T
Poin 1	10,0	0,0
Poin 2	10,10	0,5

Poin 3	0,10	0,10
Daerah yang diharapkan	50	Error

9. Untuk kalkulator waktu berlalu pada Soal 10.5, identifikasi kondisinya dan buatlah matriks pengujian.
   Jawab:
   Tidak ada ketentuan yang ditentukan pada saat itu. Satu-satunya syarat adalah waktu tersebut adalah waktu yang valid.

10. Untuk pencarian biner rutin pada Soal 10.6, kenali kondisinya dan buatlah matriks uji.
    Jawab:
    Tidak ada kondisi yang ditentukan pada pencarian.

11. Temukan himpunan kasus uji minimal yang dapat mencapai cakupan C0 dan C1 dari kodesemu masalah segitiga dari Contoh 10.3.
    Jawab:
    C0 dapat dicapai dengan satu kasus uji dengan tiga nilai yang sama kurang dari atau sama dengan nol, misalnya 0,0,0.
    C1 dapat dicapai dengan dua kasus uji: 0,0,0 dan skala tak sama panjang, misalnya 3,4,5.

12. Pseudocode berikut mengimplementasikan soal waktu berlalu pada Soal 10.5 jika waktu berlalu kurang dari 24 jam. Pilih kasus uji hingga cakupan setiap pernyataan tercapai. Pilih kasus uji tambahan untuk mencapai cakupan setiap cabang.

```
read hr1 min1 AmOrPm1
read hr2 min2 AmOrPm2
if (hr1 == 12)
     hr1=0
if (hr2 == 12)
     hr2 = 0
if (AmOrPm1 == pm)
     hr1 = hr1 + 12
if (AmOrPm2 == pm)
     hr2 = hr2 + 12
if (min2 <min1)
     min2 = min2 + 1
     hr2=hr2-1
if(hr2 <hr1)
     hr2 = hr2 + 24
elapsed =min2-min1 + 60* (hr2-hr1)
print elapsed
```

Jawab:

C0-Waktu Mulai	Waktu Berhenti	Waktu yang diharapkan berlalu
12:00	12:40 Malam	0:40
21:57	23:40	1:43
17:00	04:00	12:00

C1-Waktu Mulai	Waktu Berhenti	Waktu yang diharapkan berlalu
Tes diatas ditambah yang berikut		
08:00	12:40 Malam	04:40

13. Untuk pseudocode Soal 10.12, temukan himpunan minimal kasus uji yang akan mencapai C0 dan himpunan minimal kasus uji yang akan mencapai C1.

Jawab:

Set pengujian minimal untuk C0 memerlukan setidaknya dua kasus uji. Jam 1 harus jam 12 pada satu tes, jam 2 harus jam 12 pada satu tes, jam 1 harus sore. pada satu tes, jam 2 harus sore. dalam satu tes, menit ke-2 harus kurang dari menit ke-1 pada satu tes dan jam ke-2 harus kurang dari jam 1 pada satu tes. Hal ini dapat dilakukan pada dua kasus uji.

Min C0 – Waktu Mulai	Waktu Berhenti	Waktu yang diharapkan Berlalu
12:00	10:40	22:40
09:57	12:40	2:43

Hal ini juga mencapai cakupan C1 karena masing-masing kondisi tersebut juga salah pada salah satu pengujian ini.

14. Untuk kode berikut, identifikasi semua jalur yang layak, pengujian jalur, dan pengujian aliran data:

```
cin >> a >> b >> c; // node A
x=5; y=7;
if ( a > b && b > c) {
     a = a + 1; // node B
     x = x + 6;
     if (a = 10||b>20) {
          b = b+1; // node C
          x = y+4;
          }
     if (a<10 | | c = 20) { // node D
          b = b + 2; // node E
          y = 4
          }
     a = a + b + 1; // node F
     y = x + y;
     }
```

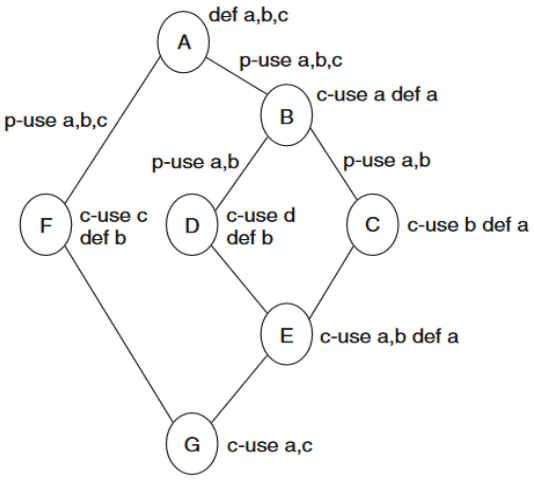
```
if (a>5 || c<10) { // node G
     b = c + 5; // node H
     x=x+1;
     }
cout >> x >> y; // node I
```

Jawab:

Path	Kebenaran	a,b,c
AGI	FxxF	4,8,12
AGHI	FxxT	4,8,8
ABDFGI	TFFF	Tidak mungkin
ABDFGHI	TFFT	12,8,6
ABCDGI	TTFF	Tidak mungkin
ABCDGHI	TTFT	24,22,8
ABCDEFGI	TTTF	Tidak mungkin
ABCDEFGHI	TTTT	24,22,20
ABDEFGI	TFTF	Tidak mungkin
ABDEFGHI	TFTT	6,4,2

Node	def	c-use	p-use
a	a,b,c,x,y		
ab,ag			a,b,c
b	a,x	a,x	
bc,bd			a,b
c	b,x	b,y	
d			
de,df			a,c
e	b,y	b	
f	a,y	a,b,x,y	
g			
gh,gi			a,c
h	b,x	c,x	
i		x,y	

Lihat gambar berikut ini:



15. Dengan kode berikut, gambarkan CFG dan hasilkan satu set kasus uji minimal untuk setiap kriteria berikut: C0, C1, dpu, dan dcu.

```
cin>> a >> b // node A
if (b>a) {
     x= b; // node B
     if (b>20) {
          x= x + 9; // node C
     }
     else {
          x=x+1; // node D
     }
     x=x+1; // node E
}
else {
     x = a // node F
     if (a> 20_{
          x=x+15; // node G
     }
     x=x-5; // node H
}
if (b>a+20) // node I
{
     x=20; // node J
}
cout << x; // node K
```

Jawab:

Jalur (yang memungkinkan diberi nomor)

1. `abceik TTF`
2. `abceijk TTT`
3. `abdeik TFF`
4. `abdeijk TTT`
5. `afhik FTF`
   `afhijk` tidak mungkin dilakukan
6. `afghik FFF`
   `afghijk` tidak mungkin dilakukan

Minimal Test

C0	Input	Output
2. abceijk	10,30	20
3. abdeik	10,20	22
6. afghik	20,15	30

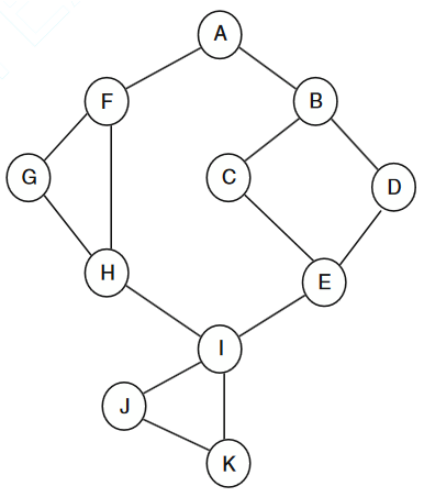
(harus menyertakan 6, tetapi bisa juga 1 dan 4)

C1	Input	Output
6. afghik	20,15	30

(harus menyertakan 6, tetapi bisa berupa jalur 1 dan 4)

dpu: Penggunaan p hanya untuk variabel a dan b. Defs satu-satunya untuk a dan b ada di node A. Set pengujian minimal setara dengan C1. Itu harus mencakup sebagian jalur AB, AF, BC, BD, FH, FG, IJ, dan IK. Hal ini dapat dilakukan dengan jalur 1,4,5,6 atau jalur 2,3,5,6

dcu: Defs variabel x ada di node B,C,D,E,F,G,H, dan J. Penggunaan c untuk variabel x ada di node C,D,E,G,H, dan K. Defs variabel a dan b ada di node A dan c-penggunaan variabel b ada di node B dan c-penggunaan variabel a ada di node F. Set pengujian minimal harus mencakup sebagian jalur BC atau BD, CE, DE, FG atau F..K, GH, H..J atau H..K, JK, AB, dan AF. Ini dapat dilakukan dengan set pengujian C1 apa pun. Perhatikan gambar disamping.



BAB 11

LATIHAN SOAL

1. Mengapa mobil Ford merupakan spesialisasi dari mobil dan mesinnya bukan?
   Jawab: *Mobil Ford akan memiliki atribut dan fungsi yang sama dengan kelas dasar mobil. Dengan demikian, mobil Ford dapat diturunkan dari kelas dasar mobil tersebut. Namun, mesin adalah bagian dari sebuah mobil dan bukan spesialisasi dari sebuah mobil. Ada banyak fungsi dan atribut mobil yang tidak dimiliki mesin. Dengan demikian, suatu mesin tidak bisa diturunkan dari mobil kelas dasar.*

2. Apa perbedaan antara objek dan atribut?
   Jawab: *Objek adalah suatu entitas, sedangkan atribut adalah ciri dari objek tersebut. Misalnya, seseorang akan menjadi objek dan tinggi badan orang tersebut akan menjadi atribut. Terkadang membedakannya mungkin sulit. Pada contoh orang/tinggi badan, orang mungkin merupakan kelas dasar dan mungkin terdapat kelas turunan untuk orang tinggi, orang pendek, dan orang dengan tinggi sedang.*

3. Apa tujuan analisis domain?
   Jawab: *Tujuan dari analisis domain adalah untuk mengidentifikasi bagian-bagian yang terbaik untuk digunakan kembali dalam sistem masa depan. Pendekatannya adalah*

untuk menemukan kesamaan di antara sistem yang mungkin ada dalam domain masalah.

Latihan soal tambahan (Diskusi)

1. Identifikasi objek dari pernyataan masalah B&B berikut:
   Tom dan Sue memulai penginapan dan sarapan di kota kecil di New England. Mereka akan memiliki tiga kamar tidur untuk para tamu. Mereka menginginkan sebuah sistem untuk mengelola pemesanan dan memantau pengeluaran dan keuntungan. Ketika calon pelanggan menelepon untuk melakukan reservasi, mereka akan memeriksa kalender, dan jika ada lowongan, mereka akan memasukkan nama pelanggan, alamat, nomor telepon, tanggal, harga yang disepakati, nomor kartu kredit, dan nomor kamar. Reservasi harus dijamin dengan pembayaran 1 hari.
   Reservasi akan dilakukan tanpa jaminan untuk waktu yang telah disepakati. Jika tidak ada jaminan pada tanggal tersebut, reservasi akan dibatalkan.
   Jawab:

   Objects

   ```
   bed-and-breakfast
   bedroom
   reservation
   calendar
   customer
   guarantee
   payment
   expense
   ```

2. Identifikasi objek dari pernyataan masalah kantor gigi berikut:
   Tom memulai praktik dokter gigi di kota kecil. Dia akan memiliki asisten gigi, ahli kesehatan gigi, dan resepsionis. Dia menginginkan sistem untuk mengatur janji temu.
   Ketika seorang pasien meminta janji temu, resepsionis akan memeriksa kalender dan akan mencoba menjadwalkan pasien sedini mungkin untuk mengisi lowongan. Jika pasien puas dengan janji temu yang diusulkan, resepsionis akan memasukkan nama pasien dan tujuan janji temu. Sistem akan memverifikasi nama pasien dan memberikan rincian yang diperlukan dari catatan pasien termasuk nomor ID pasien. Setelah setiap pemeriksaan atau pembersihan, ahli kesehatan atau asisten akan menandai janji temu telah selesai, menambahkan komentar, dan kemudian menjadwalkan pasien untuk kunjungan berikutnya jika diperlukan.
   Sistem akan menjawab pertanyaan berdasarkan nama pasien dan tanggal. Rincian pendukung dari catatan pasien ditampilkan bersama dengan informasi janji temu. Resepsionis dapat membatalkan janji temu. Resepsionis dapat mencetak daftar notifikasi untuk melakukan panggilan pengingat 2 hari sebelum janji temu. Sistem

menyertakan nomor telepon pasien dari catatan pasien. Resepsionis juga dapat mencetak jadwal kerja harian dan mingguan dengan seluruh pasien.

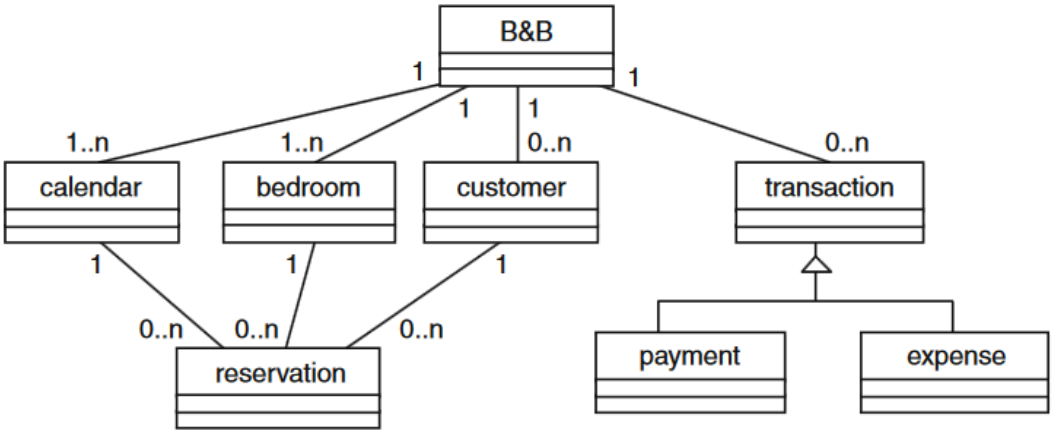
Jawab:

Objects

```
dental office
patients
appointments
calendar
patient records
notification list
daily work schedule
weekly work schedule
```

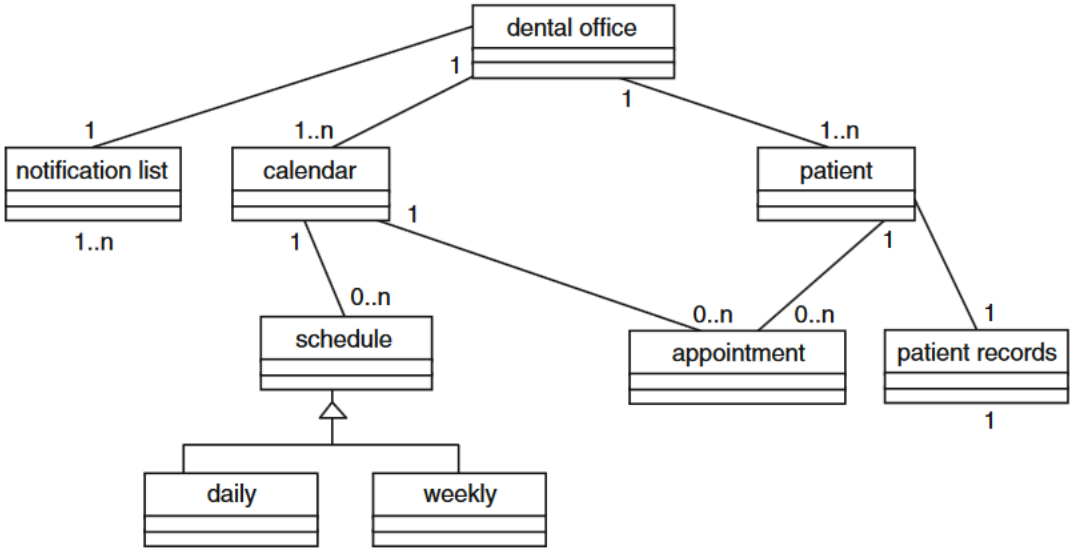
3. Gambarkan model objek untuk permasalahan B&B (Soal 11.1).

Jawab:



4. Gambarlah model objek untuk permasalahan kantor gigi (Soal 11.2).

Jawab:



BAB 12
LATIHAN SOAL

1. Mengapa bilangan siklomatik McCabe dan ilmu perangkat lunak Halstead tidak mudah diterapkan pada perangkat lunak berorientasi objek?
   Jawab: Kedua metrik ini didasarkan pada ukuran dan kompleksitas algoritma yang ditulis sebagai satu fungsi. Fungsi berorientasi objek biasanya tersebar di sejumlah metode, seringkali di kelas yang berbeda. Setiap fungsi berorientasi objek seringkali berukuran kecil dan relatif sederhana. Dengan demikian, kedua metrik ini mungkin tidak akan memberikan ukuran yang baik mengenai kompleksitas sistem berorientasi objek.

2. Abstraksi apa yang tersedia dalam desain berorientasi objek untuk digunakan sebagai dasar metrik berorientasi objek?
   Jawab: Abstraksi standar adalah diagram UML: model objek, diagram use case, model keadaan, dan diagram urutan. Tak satu pun dari hal ini yang menangkap gagasan penting tentang kompleksitas dalam perangkat lunak berorientasi objek.

3. Kapan metrik untuk keseluruhan sistem harus berbeda dengan jumlah atau rata-rata metrik yang dihitung untuk setiap kelas?
   Jawab: Jika metrik untuk masing-masing kelas pada dasarnya adalah metrik ukuran, seperti LOC atau jumlah anak, maka masuk akal untuk menjumlahkan nilai metrik individual tersebut untuk mendapatkan nilai metrik untuk keseluruhan sistem atau ukuran rata-rata per kelas. Jika metrik kelas individual adalah rata-rata, maka rata-rata dari rata-rata tersebut mungkin masuk akal—misalnya, jumlah rata-rata parameter per fungsi. Namun, baik jumlah maupun rata-rata tidak akan menjadi metrik yang baik untuk interaksi antar kelas.

4. Apakah LCOM yang tinggi baik atau buruk?
   Jawab: Buruk, karena ini menunjukkan kurangnya kohesi.

5. Ada yang berpendapat bahwa dalam LCOM hanya menggunakan selisih antara ukuran P dan Q saja. Artinya, penggunaan maksimum nol dan selisih ini tidak efektif. Apa dampak dari perubahan ini?
   Jawab: Hal ini akan memungkinkan adanya diskriminasi di antara kelas-kelas yang lebih kohesif. Sekarang kelas yang kohesif hanya dipetakan ke nol.

Latihan Soal Tambahan

1. Hitung metrik Chidamber untuk kode berikut yang mengelola array orang/siswa:

```
class person{
  char* name;
  char* ssn;
```

```
   public:
     person () {name = new char [NAMELENGTH] ; ssn = new char
[SSNLENGTH] ; }
     ~person () {delete name; delete ssn;}
     void addName (char* newname) {strcpy (name, newname)
     void addSsn (char* newssn) {strcpy(ssn, newssn) ;}
     char* getName () {return name; }
     void virtual display () {cout << "the person's name i" <<
name; }
   };
   class student public person {
     float gpa;
   public:
     void addGpa (float newgpa) {gpa = newgpa; }
     void display () {cout << "the student's name is"
     << getName () << " and gpa is " << gpa; }

   };
   class personlist {
     person* list [MAX] ;
     int listIndex;
   public:
     personlist () {listIndex=0;}
     void addPerson (char* newname, char*
     newssn) {list [listIndex] =new person;
       list [listIndex]->addName (newname); list[listIndex]
       ->addSsn (newssn) ;
       listIndex++; }
     void addStudent (char* newname, char* newssn, float gpa)
       {student* temp = new student;
       temp->addName (newname); temp->addSsn (newssn) ;
       temp->addGpa (newgpa) ; list[listIndex++]=temp; }
     void display () {int j; for(j=0; j<listIndex; j++) list[j]
     ->display();}
   };
```

Jawab:

Metrik 1: Metode Tertimbang per Kelas

Class	# Metode
Point	3
Rectangle	3
Linklistnode	4
rectanglelist	3

WMC = 12/3 = 4 methods/class

Metrik 2: Depth of Inheritance Tree (DIT)

Class	DIT
person	0

student	1
personlist	0

Metric 3: Number of Children (NOC)

Class	NOC
person	1
student	0
personlist	0

Metric 4: Coupling between Object Classes (CBO)

Lihat gambar berikut ini:

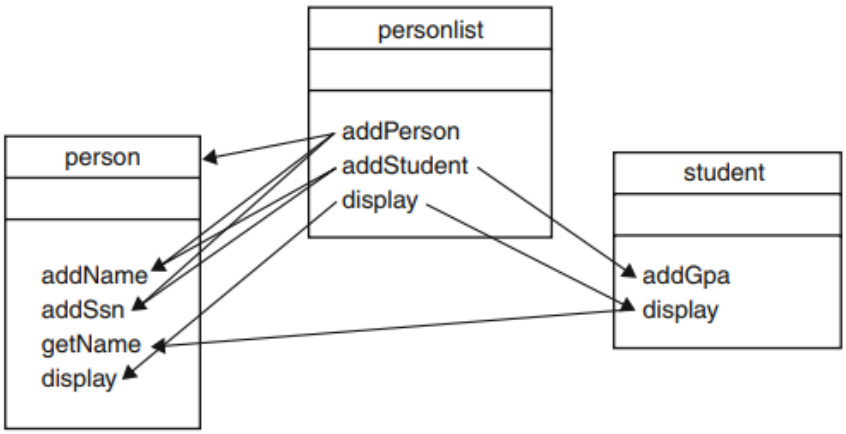


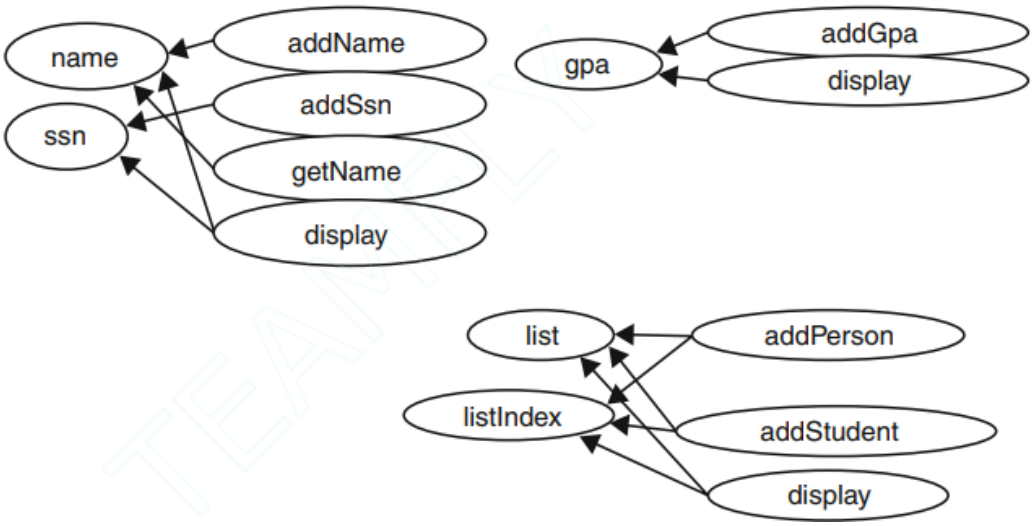
Diagram kelas dianotasi dengan panah untuk menunjukkan fungsi (atau konstruktor) mana yang dipanggil oleh setiap fungsi (hanya panggilan di kelas lain yang ditampilkan).

Class	Coupled Class	CBO
person	student, personlist	2
student	person, personlist	2
personlist	person, student	2

Metric 5: Response for a class (RFC)

Class	Coupled Class	RFC
person	person, addName, addSssm getName, display	2
student	student, addGpa, person, getName	2
personlist	personlist, addPerson, addStudent, addName, addSsn, addGpa, display	2

Metric 6: Lack of Cohesion in Methods (LCOM)



Class	LCOM
`person`	max(0,8 – 5) – 5) = 0
`student`	max(0,2 – 2) – 2) = 0
`personlist`	max(0,6 – 6) – 6) = 0

BAB 13

LATIHAN SOAL

No	Soal	Jawaban
1	Bagaimana pengujian fungsional perangkat lunak berorientasi objek dilakukan?	Pengujian fungsional perangkat lunak berorientasi objek tidak berbeda dengan pengujian fungsional perangkat lunak konvensional.
2	Apakah cakupan pernyataan perangkat lunak berorientasi objek bermanfaat?	Ya, cakupan pernyataan perangkat lunak berorientasi objek harus dilakukan. Ini mungkin merupakan cakupan paling minimal yang dapat diterima.
3	Apakah pengujian MM mencakup cakupan pernyataan? (Lihat Bagian 10.3)	Tidak, pengujian MM mengharuskan setiap pemanggilan metode diuji. Namun, bagian kode sumber tidak boleh berisi pemanggilan metode apa pun dan karenanya tidak diuji oleh serangkaian kasus uji yang mencapai pengujian MM.
4	Apa keuntungan dari cakupan pasangan fungsi?	Cakupan pasangan fungsi memastikan bahwa kombinasi panggilan dieksekusi. Dalam contoh tumpukan (13.3), jika tumpukan dipanggil oleh antarmuka pengguna, mungkin hanya ada satu panggilan untuk setiap fungsi. Jadi, urutan sederhana dari buat, dorong, dan pop mungkin mencapai pengujian MM. Cakupan pasangan fungsi mencakup pengujian MM.

Latihan Soal Tambahan

1. Dengan kode berikut, buat kasus uji yang memenuhi kriteria pengujian MM dan setiap pengujian pasangan fungsi:

```
class Threes {
  char cout;
public:
  Threes () { count = 'a' ; }
  void PlusOne () { if (count == 'a') count = 'b'; if (count ==
    'b')
  count='c';
    if (count == 'c') count = 'a'; }
  char* IsDiv(){if (count=='a'){return 'yes'; }else{return
  'no' ; } }
}
class Num {
  Threes* SumMod3; int last; int number;
  void Digit (int newnum) {int j; for (j=1; j <= newnum; j++)
  SumMod3->PlusOne ();}
public:
  Num () { SumMod3 = new Threes; }
  void Reduce() {while (number > 0) {last = number - (number/
    10)*10;
    Digit (last); number = number/10; }
  char* IsDivisibleBy3 (int newnum) {number = newnum; Reduce;
    return SumMod3->IsDiv (); }
}
Main () {
  Num* Test = new Num;
  int value;
  char* answer;
  cin >> value;
  answer = test->IsDivisibleBy3 (value);
  cout << answer;
};
```

Jawab:

Regular expressions of calls:

```
main: Num IsDivisibleBy3
Num: Threes
Reduce: Digit *
IsDivisibleBy3: Reduce IsDiv
Digit: PlusOne*
Threes:
PlusOne:
IsDiv
```

Pengujian MM:

Set pengujian harus menjalankan semua panggilan.

Satu kasus uji, masukan 5—keluaran no, harus mengeksekusi semua.

Setiap pasangan fungsi:

Set pengujian harus memiliki pengujian untuk setiap transisi `IsDiv`, sehingga input 6 dan 7 harus mencapai cakupan. Outputnya masing-masing adalah ya dan tidak.

BAB 14

LATIHAN SOAL

1. Pertanyaan seperti apa yang seharusnya dapat dijawab oleh spesifikasi?
   Jawab: *Biasanya pertanyaannya adalah tentang perilaku perangkat lunak yang diusulkan. Pengembang harus dapat menggunakan spesifikasi untuk menentukan dengan tepat apa yang harus dilakukan perangkat lunak dalam situasi tertentu.*

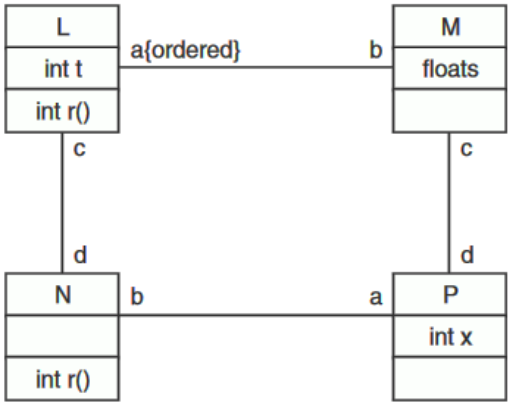
2. Mengapa ambiguitas bisa menjadi masalah?
   Jawab: *Jika ambiguitas berarti pengembang akan menafsirkan spesifikasi berbeda dari apa yang diinginkan pengguna, maka akan terjadi masalah.*

3. Mengapa gagasan matematika, seperti himpunan, merupakan dasar yang baik untuk spesifikasi?
   Jawab: *Gagasan matematika seperti himpunan merupakan dasar yang baik untuk spesifikasi karena himpunan dan operasi himpunan didefinisikan secara tepat. Misalnya, penyatuan dua himpunan telah dipahami dengan baik. Jika perilaku suatu fungsi dapat didefinisikan sebagai operasi pada himpunan tertentu, maka akan mudah untuk menentukan dengan tepat apa yang seharusnya dilakukan oleh fungsi tersebut.*

4. Apa perbedaan antara prakondisi, pascakondisi, dan invarian?
   Jawab: *Prasyarat adalah sesuatu yang harus benar sebelum suatu fungsi dapat dijalankan. Kondisi pasca adalah sesuatu yang harus benar pada penyelesaian fungsi. Invarian adalah sesuatu yang harus benar sepanjang eksekusi suatu fungsi. Sebenarnya, sebagian besar invarian berlaku di setiap operasi.*

Latihan Soal Tambahan

1. Mengingat model objek yang ditunjukkan pada Gambar dibawah ini, evaluasi setiap pernyataan OCL yang diberikan. Jika pernyataan tersebut salah, jelaskan apa yang salah dan tentukan koreksi yang paling sederhana.

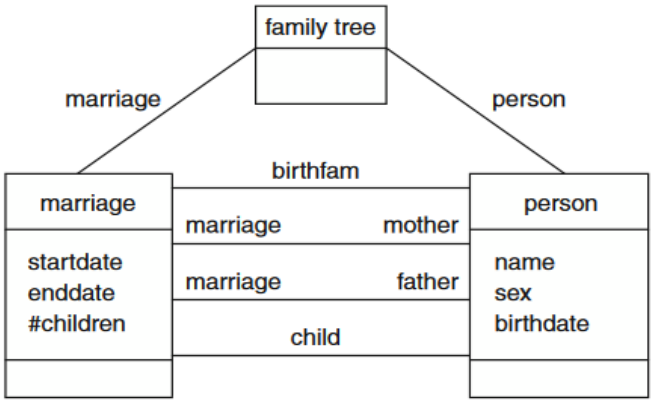



```
L
self.c->size= 10
self.a=self.c.b.d

L::r():int
pre: self.a.b=self.c.a
post: t = t@pre + 1
post: result = self.a->first.s

P
self.a.d->size > max
self.a.b=self.d.c

N::q():int
pre: self.b->isEmpty
pre: self.d->forall(1 |1.t <10)
post: result = self.d->size
```

Jawab:

Semua baik-baik saja kecuali `pre:self.a.b = self.c.a`, yang seharusnya `pre:self.a.b = self.c.d`.

2. Berdasarkan model objek yang ditunjukkan pada Gambar dibawah ini, jelaskan setiap pernyataan OCL. Apa yang ditentukannya? Apakah invarian OCL masuk akal? Apakah itu selalu benar?




```
familytree
  a) self.person = self.marriage.child
marriage
  b) self.child.birthfam =self
  c) self.husband.birthdate < self.wife.birthdate
person
  d) self.birthfam.child_include(self)
  e) self.marriage->size = 1
  f) self.marriage.wife.birthdate < self.birthdate
```

Jawab:

a) Invarian ini mengatakan bahwa himpunan orang-orang sama dengan himpunan semua anak atau setiap orang mempunyai perkawinan kelahirannya yang terdaftar. Ini adalah invarian yang masuk akal, dan benar jika datanya lengkap.

b) Invarian ini mengatakan bahwa setiap anak memiliki keluarga kandungnya yang terdaftar dan cocok dengan contoh yang menunjuk pada orang tersebut sebagai seorang anak. Ini masuk akal dan selalu benar.

c) Artinya setiap suami lebih tua dari istrinya. Invarian ini dapat dinyatakan, namun tidak sesuai dengan kenyataan.

d) Hal ini menyatakan bahwa himpunan anak (saudara kandung) yang dapat dijangkau dari keluarga kelahiran termasuk orang tersebut. Ini masuk akal dan selalu benar.

e) Hal ini menyatakan bahwa himpunan perkawinan bagi seseorang hanya ada satu. Itu tidak sesuai dengan kenyataan.

f) Entah tanggal lahir Anda sendiri (jika perempuan) atau tanggal lahir pasangan Anda (jika laki-laki) lebih kecil dari tanggal lahir Anda. Tidak masuk akal. Tidak selalu benar.

3. Tulis batasan OCL untuk restoran tanpa bagian merokok yang menempatkan pelanggan berdasarkan urutan kedatangan.

```
Class group
  Char* name
```

```
   Int number
   Int arrivalorder
 Class waitlist
   Group* list [MAX]
   Int listptr
   Void addtolist (group* newgroup)
   Group* seatnext ()
 Class restaurant
   Waitlist* waiting
   Void arrive(group* newgroup)
   Group* seat ()
```

Daftar Tunggu:

```
   Self.Listptr = self.list->size
Void waitlist :: addtolist (group* newgroup)
   Pre: self. listptr < MAX
   Post: forall (i | list [i] = list [i-1]@pre)
     List [0] = newgroup
     Listptr = listptr@pre+1
Group* waitlist: : seatnext ()
   Pre: self.listprt > 0
   Post: Result = list[listptr@pre]
     Self.listptr = self.listprt@pre - 1
Group* Restaurant: :seat
   Pre: waiting.waitlist->size > 0
   Post: waiting.waitlist->size = waiting. waitlist@pre->size
   - 1
     Result = waiting.seatnext ()
     and forall (x : group | waiting.seatnext () .arrivalorder
     <= x.arrivalorder)
```

Teori & Problem

RPL

(Rekayasa Perangkat Lunak)

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

BIODATA PENULIS

Dr. Joseph Teguh Santoso, M.Kom adalah pemimpin yang visioner dan praktisi industri berpengalaman, yang menjabat sebagai Rektor Universitas Sains dan Teknologi Komputer (Universitas STEKOM), salah satu universitas terkemuka di Jawa Tengah, Indonesia. Dengan pengalaman lebih dari 13 tahun di dunia bisnis dan praktisi industri di China, beliau membawa perspektif global dan inovasi yang signifikan ke dalam dunia akademis.Sebagai seorang entrepreneur, penulis adalah pencipta TopLoker.com, sebuah platform inovatif yang merevolusi cara mencari dan menawarkan pekerjaan. TopLoker.com adalah portal lowongan bursa kerja terbesar di Indonesia, khusus untuk pendidikan SMA/SMK sederajat. TopLoker.com telah mendapatkan penghargaan sebagai juara 1 Startup4industry 2022 oleh Kementerian Perindustrian Republik Indonesia. Kontribusi Dr. Joseph dalam menyediakan akses pekerjaan yang luas bagi lulusan SMA/SMK telah membantu banyak individu menemukan peluang kerja yang sesuai dengan keahlian mereka. Selain itu, Dr. Joseph Teguh Santoso, M.Kom adalah pendiri dari dua organisasi yaitu (1) organisasi guru/pendidik PTIC (Perkumpulan Teacherpreneur Indonesia Cerdas) yang bertujuan untuk meningkatkan kualitas pendidikan dan kesejahteraan guru/pendidik dengan wawasan entrepreneurship, serta (2) organisasi industri PERKIVI (Perkumpulan Komunitas Industri dan Vokasi Indonesia) yang berfokus pada pengembangan link and match antara industri dan dunia pendidikan. Sebagai Rektor, Dr. Joseph Teguh Santoso, M.Kom memiliki kepemimpinan yang berorientasi pada hasil, dan berkomitmen untuk mendorong kemajuan Universitas Sains dan Teknologi Komputer (Universitas STEKOM). Saat ini Universitas STEKOM telah mengalami transformasi positif dalam peningkatan kualitas pendidikan, perluasan fasilitas, serta penguatan kemitraan Perguruan Tinggi Nasional dan Internasional. Beliau memprioritaskan pengembangan sumber daya manusia dan penelitian, serta memastikan bahwa universitas berada di garis depan dalam inovasi dan teknologi untuk mencapai tujuan akhir, yaitu lulusan yang mampu bekerja dan sukses setelah lulus.Dr. Joseph Teguh Santoso, M.Kom sering diundang sebagai pembicara di berbagai konferensi nasional maupun internasional dan telah menerima berbagai penghargaan atas dedikasinya dalam bidang pendidikan, industri, dan kewirausahaan.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Teori & Problem

RPL

(Rekayasa Perangkat Lunak)

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8642-15-1 (PDF)
9 786238 642151