



Pertemuan 6: Principles that Guide Practices

Meuthia Rachmaniah
Departemen Ilmu Komputer, FMIPA IPB

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

6. Principles that Guide Practices

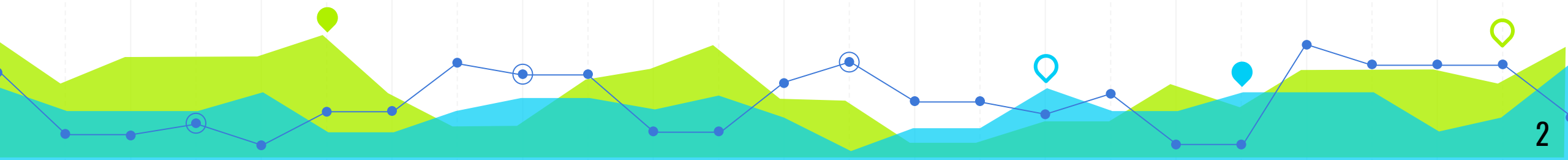
◎ 6.1 CORE PRINCIPLES

- ◎ 6.1.1 Principles that Guide Process (Prinsip yang Memandu Proses)
- ◎ 6.1.2 Principles that Guide Practices (Prinsip yang Memandu Praktik)

◎ 6.2 PRINCIPLES THAT GUIDE EACH FRAMEWORK ACTIVITY

- ◎ 6.2.1 Communication Principles (Prinsip Komunikasi)
- ◎ 6.2.2 Planning Principles (Prinsip Perencanaan)
- ◎ 6.2.3 Modeling Principles (Prinsip Pemodelan)
- ◎ 6.2.4 Construction Principles (Prinsip Konstruksi)
- ◎ 6.2.5 Deployment Principles (Prinsip Deployment)

◎ 6.3 RANGKUMAN



Quick Look

WHAT IS IT?



- Praktik RPL ialah array dari prinsip-prinsip, konsep-konsep, metode-metode, dan perangkat bantu (tools) yang harus Anda pertimbangkan ketika PL direncanakan dan dikembangkan
- Prinsip-prinsip yang memandu praktik membentuk fondasi untuk melakukan RPL

WHO DOES IT



- Praktisi (Software Engineer) dan manajer melakukan berbagai tugas RPL

WHY IS IT IMPORTANT?



- Proses PL memberi semua orang yang terlibat dalam pembuatan sistem atau produk berbasis komputer dengan peta jalan (road map) untuk mencapai tujuan yang sukses
- Praktik PL memberi Anda detail yang Anda perlukan seperti untuk mengemudi di sepanjang jalan
- Proses PL membantu untuk memahami konsep-konsep dan prinsip-prinsip
- Dalam konteks RPL, praktik adalah apa yang Anda lakukan hari demi hari saat PL berkembang dari ide menjadi kenyataan



WHAT ARE THE STEPS

- Empat elemen praktik berlaku terlepas dari model proses yang dipilih:
 - Prinsip, konsep, metode, dan perangkat bantu (tools)
 - Perangkat bantu (tools) mendukung penerapan metode



WHAT IS THE WORK PRODUCT


- Praktik meliputi kegiatan teknis yang menghasilkan semua produk kerja (work product) yang didefinisikan oleh model proses PL yang telah dipilih



HOW DO I ENSURE THAT I'VE DONE IT RIGHT

- Pertama, memiliki pemahaman yang kuat tentang prinsip-prinsip yang berlaku untuk pekerjaan (misalnya, desain) yang Anda lakukan saat ini
- Kemudian, pastikan bahwa Anda telah memilih metode yang sesuai untuk pekerjaan itu, pastikan bahwa Anda memahami cara menerapkan metode tersebut, menggunakan perangkat bantu (tools) otomatis saat sesuai untuk tugas tersebut, dan bersikeras tentang perlunya teknik untuk memastikan kualitas produk kerja yang dihasilkan
- Anda juga harus cukup gesit (agile) untuk membuat perubahan pada rencana dan metode Anda sesuai kebutuhan

What is Software Engineering “Practice”



Praktik adalah koleksi konsep, prinsip, metode, dan tools yang digunakan seorang software engineer setiap hari

Praktik memungkinkan manajer untuk mengelola PL dan software engineer untuk membangun program komputer

Praktik mengisi model proses PL dengan cara teknis dan manajemen yang diperlukan untuk menyelesaikan pekerjaan

Praktik mengubah pendekatan tidak fokus yang serampangan menjadi sesuatu yang lebih terorganisir, lebih efektif, dan lebih mungkin untuk mencapai kesuksesan



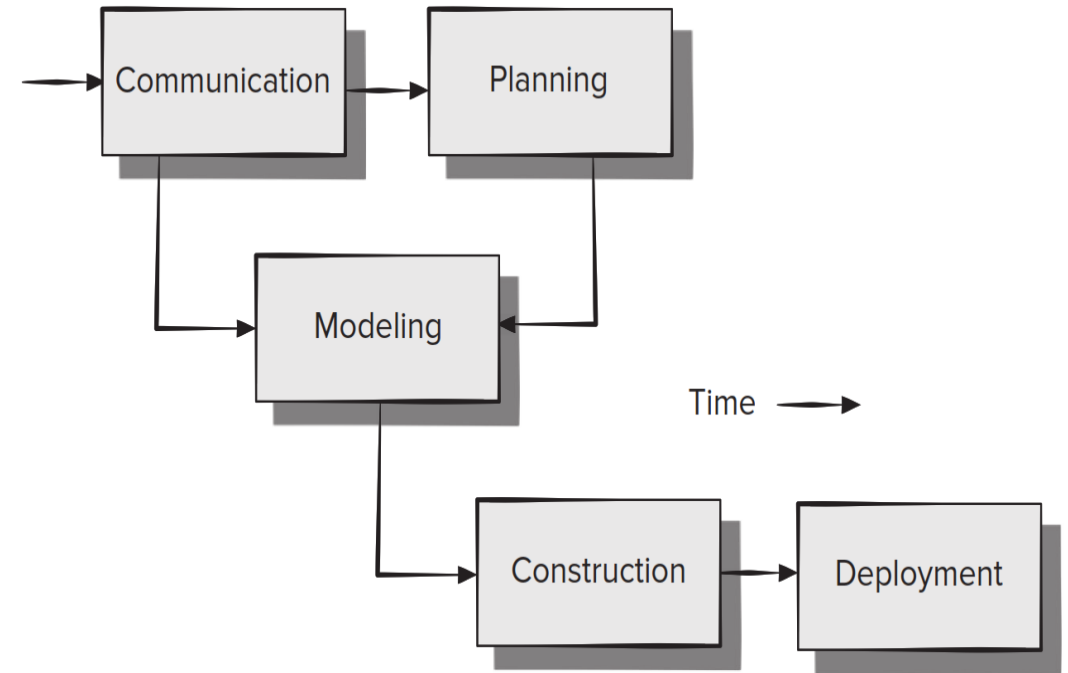
6.1 CORE PRINCIPLES

6.1.1 Principles that Guide Process (Prinsip yang Memandu Proses)

6.1.2 Principles that Guide Practices (Prinsip yang Memandu Praktik)

6.1.1 Principles that Guide PROCESS (Prinsip yang Memandu PROSES)

- Setiap proyek itu unik, dan setiap tim itu unik
 - Itu berarti Anda harus menyesuaikan proses Anda agar sesuai dengan kebutuhan Anda.
 - Terlepas dari model proses yang diadopsi tim Anda, proses berisi elemen dari framework proses generik
- Framework proses yang disederhanakan ialah seperti gambar di kanan ini



Framework Proses yang
Disederhanakan

Delapan prinsip inti dapat diterapkan pada FRAMEWORK PROSES dan untuk setiap PROSES PERANGKAT LUNAK

● Prinsip 1. Harus Agile

- Apakah model proses yang Anda pilih bersifat preskriptif atau agile, prinsip dasar pengembangan agile harus mengatur pendekatan Anda
- Setiap aspek pekerjaan yang Anda lakukan harus menekankan tindakan ekonomi—buat pendekatan teknis Anda sesederhana mungkin, pertahankan produk kerja yang Anda hasilkan sesingkat mungkin, dan buat keputusan secara lokal bila memungkinkan

● Prinsip 2. Fokus pada Kualitas di Setiap langkah

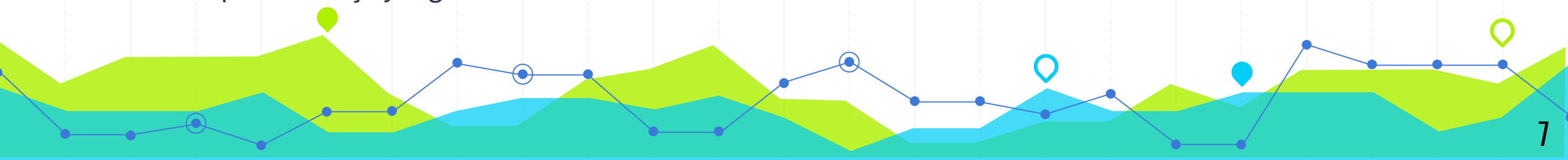
- Kondisi keluar untuk setiap aktivitas proses, tindakan, dan tugas harus fokus pada kualitas produk kerja yang telah dihasilkan

● Prinsip 3. Selalu Siap Beradaptasi

- Proses bukanlah pengalaman religius, dan dogma tidak memiliki tempat di dalamnya
- Bila perlu, sesuaikan pendekatan Anda dengan kendala yang ditimbulkan oleh problem, orang-orang, dan proyek itu sendiri

● Prinsip 4. Bangun Tim yang Efektif

- Proses dan praktik rekayasa perangkat lunak itu penting, tetapi intinya adalah manusia
- Bangun tim yang mengatur diri sendiri yang saling percaya dan menghormati



Delapan prinsip inti dapat diterapkan pada framework proses dan untuk setiap proses perangkat lunak

● **Prinsip 5. Bangun mekanisme komunikasi & koordinasi**

- Proyek gagal karena informasi penting tidak sampai dan/atau stakeholder gagal mengoordinasikan upaya mereka untuk menciptakan produk akhir yang sukses
- Ini adalah masalah manajemen, dan mereka harus ditangani

● **Prinsip 6. Kelola perubahan**

- Pendekatannya bisa formal atau informal, tetapi mekanisme harus ditetapkan untuk mengelola cara perubahan diminta, dinilai, disetujui, dan diimplementasikan

● **Prinsip 7. Kaji risiko**

- Banyak hal bisa salah ketika PL sedang dikembangkan
- Sangat penting bagi Anda untuk membuat rencana darurat
- Beberapa dari rencana darurat ini akan menjadi dasar untuk tugas-tugas rekayasa keamanan

● **Prinsip 8. Ciptakan Work Product dan pertimbangkan orang lain**

- Buat hanya produk kerja yang memberikan nilai untuk aktivitas, tindakan, atau tugas proses lainnya.
- Setiap produk kerja yang dihasilkan sebagai bagian dari praktik rekayasa perangkat lunak akan diteruskan ke orang lain.
- Pastikan bahwa produk kerja memberikan informasi yang diperlukan tanpa ambiguitas atau kelalaian

6.1.2 Principles that Guide PRACTICES (Prinsip yang Memandu PRAKTIK)

- Praktik RPL memiliki satu tujuan utama:
 - untuk memberikan PL operasional yang tepat waktu, berkualitas tinggi, yang berisi fungsi dan fitur yang memenuhi kebutuhan semua stakeholder
- Untuk mencapai tujuan ini, Anda harus mengadopsi seperangkat prinsip inti yang memandu pekerjaan teknis Anda
 - Prinsip-prinsip ini memiliki manfaat terlepas dari metode analisis dan desain yang Anda terapkan, teknik konstruksi (misalnya, bahasa pemrograman, perangkat bantu otomatis) yang Anda gunakan, atau pendekatan verifikasi dan validasi yang Anda pilih

Delapan Prinsip Inti Sangat Mendasar bagi PRAKTIK RPL



Prinsip 1: Divide and conquer

- Dinyatakan dengan cara yang lebih teknis, analisis dan desain harus selalu menekankan pemisahan perhatian/kekhawatiran (*Separation of Concerns-SoCs*)
- Masalah besar lebih mudah dipecahkan jika dibagi lagi menjadi kumpulan elemen (atau masalah/*concerns*)



Prinsip 2: Memahami Penggunaan Abstraksi

- Abstraksi adalah penyederhanaan beberapa elemen kompleks dari sistem yang digunakan untuk mengkomunikasikan makna dalam satu frase
- Saat menggunakan spreadsheet abstraksi, pahami apa itu spreadsheet, struktur umum konten yang disajikan spreadsheet, dan fungsi umum yang dapat diterapkan
- Menggunakan banyak tingkat abstraksi yang berbeda, masing-masing menyampaikan atau menyiratkan makna yang harus dikomunikasikan
- Dalam pekerjaan analisis dan desain, tim PL biasanya memulai dengan model yang mewakili abstraksi tingkat tinggi (misalnya, spreadsheet) dan perlahan menyempurnakan model tersebut ke tingkat abstraksi yang lebih rendah (misalnya, kolom atau fungsi SUM)



Prinsip 3: Berusaha untuk konsistensi

- Ketika membuat model analisis, mengembangkan desain PL, menghasilkan kode sumber, atau membuat kasus uji, prinsip konsistensi menunjukkan bahwa konteks yang familiar membuat PL lebih mudah digunakan
- Sebagai contoh, pertimbangkan desain antarmuka pengguna untuk aplikasi seluler:
 - Penempatan opsi menu yang konsisten, penggunaan skema warna yang konsisten, dan penggunaan ikon yang dapat dikenali secara konsisten membantu menciptakan pengalaman pengguna (*user experience*) yang sangat efektif



Prinsip 4: Fokus pada Transfer Informasi

- PL adalah tentang transfer informasi—from database ke pengguna akhir, dari sistem warisan ke WebApp, dari pengguna akhir ke antarmuka pengguna grafis (GUI), dari sistem operasi ke aplikasi, dari satu komponen PL ke komponen lainnya—daftarnya hampir tidak ada habisnya
- Dalam setiap kasus, informasi mengalir melintasi antarmuka, dan ini berarti ada peluang untuk kesalahan, kelalaian, atau ambiguitas
- Implikasi dari prinsip ini adalah Anda harus memberikan perhatian khusus pada analisis, desain, konstruksi, dan pengujian antarmuka

Delapan Prinsip Inti Sangat Mendasar bagi PRAKTIK RPL



Prinsip 5: Bangun PL yang Menunjukkan Modularitas Efektif

- Pemisahan masalah (Prinsip 1) menetapkan filosofi untuk PL
- Modularitas menyediakan mekanisme untuk mewujudkan filosofi
- Sistem kompleks apa pun dapat dibagi menjadi modul (komponen), tetapi praktik RPL yang baik menuntut lebih banyak
- Modularitas harus efektif. Artinya, setiap modul harus fokus secara eksklusif pada satu aspek sistem yang dibatasi dengan baik
- Modul harus saling berhubungan dengan cara yang relatif sederhana ke modul lain, ke sumber data, dan aspek lingkungan lainnya



Prinsip 6: Carilah Pola

- Software Engineer menggunakan pola sebagai sarana untuk membuat katalog dan menggunakan kembali solusi untuk problem yang mereka temui di masa lalu
- Penggunaan pola desain ini dapat diterapkan pada rekayasa sistem dan masalah integrasi sistem yang lebih luas, dengan memungkinkan komponen dalam sistem yang kompleks berkembang secara independen



Prinsip 7: Jika Memungkinkan, Tunjukkan Problem dan Solusinya dari Beberapa Perspektif yang Berbeda

- Ketika sebuah problem dan solusinya diperiksa dari perspektif yang berbeda, kemungkinan besar wawasan yang lebih besar akan tercapai dan kesalahan serta kelalaian akan terungkap
- Unified Modeling Language (UML) menyediakan sarana untuk menggambarkan solusi problem dari berbagai sudut pandang



Prinsip 8: Ingatlah Bahwa Seseorang Akan Memelihara PL

- Dalam jangka panjang, PL akan dikoreksi saat cacat ditemukan, disesuaikan dengan perubahan lingkungannya, dan ditingkatkan saat stakeholder meminta lebih banyak kemampuan
- Kegiatan pemeliharaan ini dapat difasilitasi jika praktik RPL yang solid diterapkan di seluruh proses PL

6.2 PRINCIPLES THAT GUIDE EACH FRAMEWORK ACTIVITY

6.2.1 Communication Principles (Prinsip Komunikasi)

6.2.2 Planning Principles (Prinsip Perencanaan)

6.2.3 Modeling Principles (Prinsip Pemodelan)

6.2.4 Construction Principles (Prinsip Konstruksi)

6.2.5 Deployment Principles (Prinsip Deployment)

6.2.1 Communication Principles (Prinsip KOMUNIKASI)

- Perbedaan Customer dan End User
- Dalam beberapa kasus, customer (pelanggan) dan pengguna akhir (end user) mungkin satu dan sama, tetapi untuk banyak proyek tidak demikian
- Customers adalah seseorang/grup yang :
 - 1) awalnya meminta PL yang akan dibangun,
 - 2) mendefinisikan tujuan bisnis keseluruhan untuk PL,
 - 3) menyediakan persyaratan produk dasar, dan
 - 4) mengoordinasikan pendanaan untuk proyek
- End Users adalah seseorang/grup yang:
 - 1) Akan benar-benar menggunakan PL yang dibuat untuk mencapai beberapa tujuan bisnis
 - 2) Akan menentukan detail operasional PL sehingga tujuan bisnis dapat tercapai

6.2.1 Communication Principles (Prinsip KOMUNIKASI)

- Persyaratan pelanggan harus dikumpulkan melalui aktivitas komunikasi
- Pelanggan memiliki problem yang mungkin dapat diselesaikan dengan solusi berbasis komputer
- Anda menanggapi permintaan bantuan pelanggan
- Komunikasi telah dimulai. Tapi jalan dari komunikasi menuju pemahaman seringkali berliku



Sepuluh Prinsip KOMUNIKASI



Prinsip 1. Dengarkan

- Pastikan Anda memahami sudut pandang pihak lain, tahu sedikit tentang kebutuhannya, dan kemudian mendengarkan
- Cobalah untuk fokus pada kata-kata pembicara, daripada merumuskan respons Anda terhadap kata-kata itu
- Mintalah klarifikasi jika ada sesuatu yang tidak jelas, dan hindari interupsi terus-menerus
- Jangan pernah menjadi kontroversial dalam kata-kata atau tindakan Anda (misalnya, memutar mata atau menggelengkan kepala) saat seseorang berbicara



Prinsip 2. Persiapkan Diri sebelum Anda berkomunikasi

- Luangkan waktu untuk memahami problem sebelum Anda bertemu dengan orang lain
- Jika perlu, lakukan riset untuk memahami jargon domain bisnis
- Jika Anda memiliki tanggung jawab untuk mengadakan rapat, siapkan agenda sebelum rapat



Prinsip 3. Seseorang Harus Memfasilitasi Kegiatan

- Setiap pertemuan komunikasi harus memiliki seorang pemimpin (fasilitator) untuk:
 - 1) Menjaga percakapan bergerak ke arah yang produktif,
 - 2) Menengahi setiap konflik yang terjadi, dan
 - 3) Memastikan bahwa prinsip-prinsip lain diikuti.



Prinsip 4. Komunikasi Face-to-Face adalah Terbaik

- Bekerja lebih baik ketika beberapa representasi lain dari informasi yang relevan hadir.
- Mis., seorang peserta dapat membuat gambar atau dokumen "strawman" yang berfungsi sebagai fokus diskusi



Prinsip 5. Buat Catatan dan Dokumentasikan Keputusan

- Hal-hal memiliki cara untuk jatuh ke dalam celah
- Seseorang yang berpartisipasi dalam komunikasi harus berfungsi sebagai "perekam" dan menuliskan semua poin dan keputusan penting

Sepuluh Prinsip KOMUNIKASI



Prinsip 6. Upayakan Kolaborasi

- Kolaborasi dan konsensus terjadi ketika pengetahuan kolektif anggota tim digunakan untuk menggambarkan fungsi atau fitur produk atau system
- Setiap kolaborasi kecil berfungsi untuk membangun kepercayaan di antara anggota tim dan menciptakan tujuan bersama untuk tim



Prinsip 7. Tetap Fokus; Diskusi Dimodularisasi

- Semakin banyak orang yang terlibat dalam komunikasi apa pun, semakin besar kemungkinan diskusi akan terpental dari satu topik ke topik berikutnya
- Fasilitator harus menjaga percakapan tetap modular, meninggalkan satu topik hanya setelah diselesaikan (namun, lihat Prinsip 9)



Prinsip 8. Jika Ada yang Tidak Jelas, Buatlah Gambar

- Komunikasi verbal hanya berjalan sejauh ini.
- Sebuah sketsa atau gambar seringkali dapat memberikan kejelasan ketika kata-kata gagal untuk melakukan pekerjaan itu



Prinsip 9. (a) Setelah Anda menyetujui sesuatu, lanjutkan. (b) Jika Anda tidak dapat menyetujui sesuatu, lanjutkan. (c) Jika fitur atau fungsi tidak jelas dan tidak dapat diklarifikasi sekarang, lanjutkan

- Komunikasi, seperti aktivitas rekayasa perangkat lunak lainnya, membutuhkan waktu
- Daripada mengulangi tanpa henti, orang-orang yang berpartisipasi harus menyadari bahwa banyak topik memerlukan diskusi (lihat Prinsip 2) dan bahwa "move on" terkadang merupakan cara terbaik untuk mencapai komunikasi yang agile

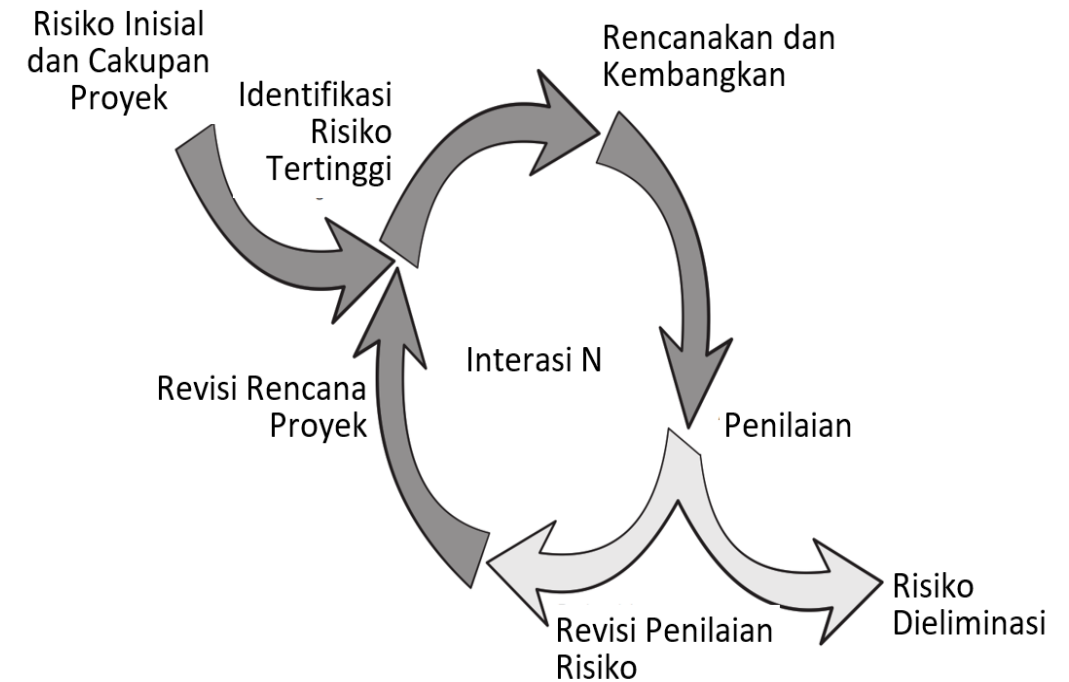


Prinsip 10. Negosiasi bukanlah kontes atau 'game'. Ini bekerja paling baik ketika kedua belah pihak menang

- Ada banyak contoh di mana Anda dan stakeholder lainnya harus menegosiasikan fungsi dan fitur, prioritas, dan tanggal pengiriman
- Jika tim telah bekerja sama dengan baik, semua pihak memiliki tujuan yang sama
- Namun, negosiasi akan menuntut kompromi dari semua pihak

6.2.2 Planning Principles (Prinsip PERENCANAAN)

- Aktivitas perencanaan mencakup serangkaian praktik manajemen dan teknis yang memungkinkan tim PL untuk menentukan peta jalan saat bergerak menuju tujuan strategis dan tujuan taktisnya
- Tim PL yang baik harus merencanakan pendekatannya. Seringkali perencanaan dilakukan berulang
- Dua Filosofi Perencanaan:
 - **Minimalis**. Perubahan sering meniadakan kebutuhan akan rencana terperinci
 - **Tradisional**. Rencana tersebut memberikan peta jalan yang efektif dan semakin detail, semakin kecil kemungkinan tim akan tersesat
- Seperti kebanyakan hal dalam hidup, perencanaan harus agile dan dilakukan secukupnya, cukup untuk memberikan panduan yang berguna bagi tim—tidak lebih, tidak kurang



Perencanaan Iteratif

Sepuluh Prinsip PERENCANAAN



Prinsip 1. Pahami Cakupan Proyek

- Tidak mungkin menggunakan peta jalan jika Anda tidak tahu ke mana Anda akan pergi
- Lingkup memberi tim perangkat lunak tujuan



Prinsip 2. Libatkan Stakeholder dalam Kegiatan Perencanaan

- Stakeholder menentukan prioritas dan menetapkan kendala proyek
- Untuk mengakomodasi kenyataan ini, Software Engineer harus sering menegosiasikan urutan pengiriman, garis waktu, dan masalah terkait proyek lainnya



Prinsip 3. Ketahuilah Bahwa Perencanaan Itu Berulang

- Rencana proyek tidak pernah terukir di atas batu. Saat pekerjaan dimulai, kemungkinan besar segalanya akan berubah. Rencananya perlu disesuaikan
- Model proses inkremental yang berulang mencakup waktu untuk merevisi rencana setelah pengiriman setiap peningkatan perangkat lunak berdasarkan umpan balik yang diterima dari pengguna



Prinsip 4. Estimasi Berdasarkan Apa Yang Anda Ketahui

- Maksud dari estimasi adalah untuk memberikan indikasi upaya, biaya, dan durasi tugas, berdasarkan pemahaman tim saat ini tentang pekerjaan yang harus dilakukan.
- Jika informasi tidak jelas atau tidak dapat diandalkan, perkiraan akan sama tidak dapat diandalkan



Prinsip 5. Pertimbangkan Risiko Saat Anda Menentukan Rencana

- Jika Anda telah mengidentifikasi risiko yang memiliki dampak tinggi dan probabilitas tinggi, perencanaan kontinjensi diperlukan
- Rencana proyek (termasuk jadwal) harus disesuaikan untuk mengakomodasi kemungkinan terjadinya satu atau beberapa risiko ini



Prinsip 6. Realistis

- Orang tidak bekerja 100 persen setiap hari. Perubahan akan terjadi
- Bahkan Software Engineer terbaik pun membuat kesalahan
- Ini dan kenyataan lainnya harus dipertimbangkan saat rencana proyek ditetapkan

Sepuluh Prinsip PERENCANAAN



Prinsip 7. Sesuaikan Granularitas Saat Menentukan Rencana

- Istilah granularitas mengacu pada detail yang dengannya beberapa elemen perencanaan diwakili atau dilakukan
- Rencana granularitas tinggi memberikan detail tugas kerja yang signifikan yang direncanakan dalam waktu yang relatif singkat (sehingga pelacakan dan kontrol sering terjadi)
- Rencana granularitas rendah memberikan tugas kerja yang lebih luas yang direncanakan dalam periode waktu yang lebih lama
- Secara umum, granularitas bergerak dari tinggi ke rendah saat garis waktu proyek menjauh dari tanggal saat ini
- Aktivitas yang tidak akan terjadi selama berbulan-bulan tidak memerlukan granularitas yang tinggi (terlalu banyak yang dapat berubah)



Prinsip 8. Tentukan Bagaimana Anda Ingin Memastikan Kualitas

- Rencana tersebut harus mengidentifikasi bagaimana tim PL bermaksud untuk memastikan kualitas
- Jika tinjauan teknis akan dilakukan, maka harus dijadwalkan
- Jika pair programming akan digunakan selama konstruksi, maka harus didefinisikan secara eksplisit dalam rencana



Prinsip 9. Jelaskan Bagaimana Anda Berniat untuk Mengakomodasi Perubahan

- Perubahan yang tidak terkendali dapat meniadakan bahkan perencanaan terbaik
- Anda harus mengidentifikasi bagaimana perubahan diakomodasi saat pekerjaan RPL berlangsung. Misalnya, dapatkah pelanggan meminta perubahan kapan saja?
- Jika diminta perubahan, apakah tim wajib segera menerapkannya?
- Bagaimana dampak dan biaya perubahan dinilai?

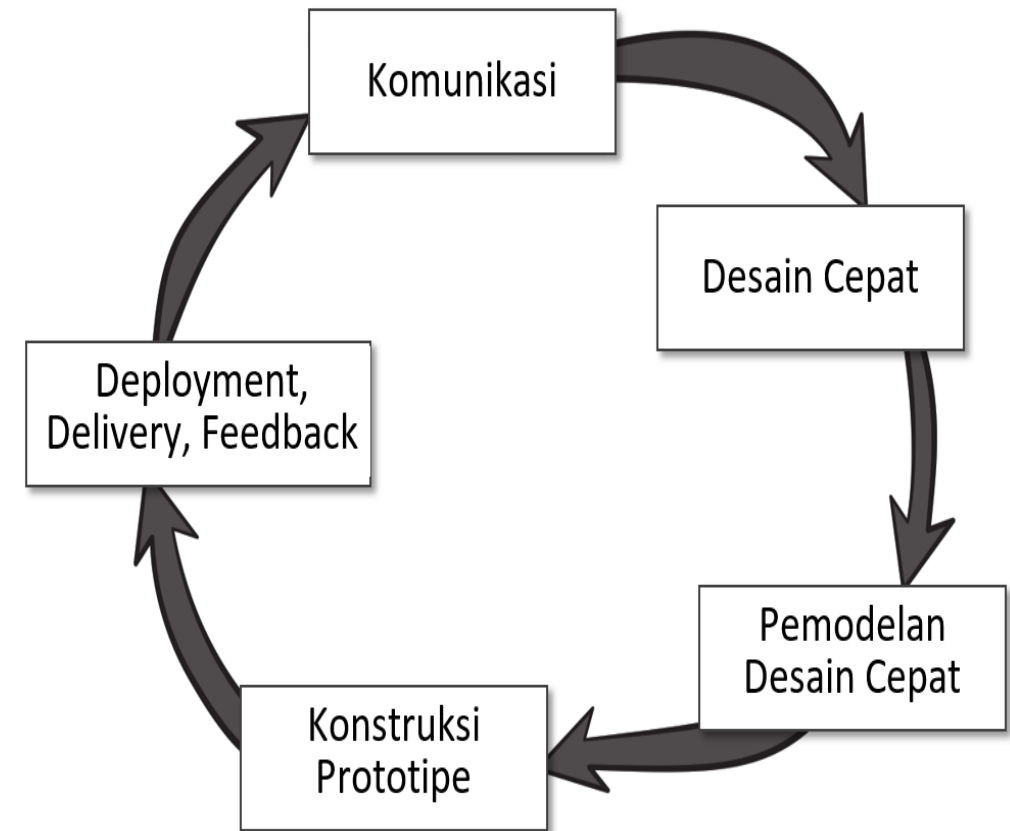


Prinsip 10. Lacak Rencana Sesering Mungkin, dan Buat Penyesuaian Sesuai Kebutuhan

- Proyek PL jatuh terlambat jadwal satu hari pada suatu waktu
- Oleh karena itu, masuk akal untuk melacak kemajuan setiap hari, mencari area masalah dan situasi di mana pekerjaan yang dijadwalkan tidak sesuai dengan pekerjaan yang sebenarnya dilakukan
- Saat terjadi selip jadwal, maka rencana akan disesuaikan

6.2.3 Modeling Principles (Prinsip PEMODELAN)

- Untuk mendapatkan pemahaman yang lebih baik tentang entitas sebenarnya yang akan dibangun
- Harus mampu mewakili informasi yang ditransformasikan PL, arsitektur dan fungsi yang memungkinkan transformasi terjadi, fitur yang diinginkan pengguna, dan perilaku sistem saat transformasi berlangsung
- Harus mencapai tujuan ini pada tingkat abstraksi yang berbeda—pertama menggambarkan PL dari sudut pandang pelanggan dan kemudian mewakili perangkat lunak pada tingkat yang lebih teknis
- Gambar di kanan menunjukkan bagaimana pemodelan dapat digunakan dalam desain PL agile



Peran Pemodelan Software

6.2.3 Modeling Principles (Prinsip PEMODELAN)

Requirements Models/ Analysis Models

Merepresentasikan kebutuhan pelanggan dengan menggambarkan perangkat lunak dalam tiga domain yang berbeda: domain informasi, domain fungsional, dan domain perilaku



Design Models

Merepresentasikan karakteristik perangkat lunak yang membantu praktisi untuk membangunnya secara efektif: arsitektur, antarmuka pengguna, dan detail tingkat komponen

Sepuluh Prinsip PEMODELAN



Prinsip 1. Tujuan utama tim perangkat lunak adalah membangun PL, bukan membuat model

- Agilitas berarti memberikan PL kepada pelanggan dalam waktu secepat mungkin
- Model yang mewujudkan hal ini layak untuk dibuat, tetapi model yang memperlambat proses atau memberikan sedikit wawasan baru harus dihindari



Prinsip 2. “Perjalanan Ringan” - Jangan Membuat Lebih Banyak Model Daripada yang Anda Butuhkan

- Setiap model yang dibuat harus selalu up to date saat terjadi perubahan
- Lebih penting lagi, setiap model baru membutuhkan waktu yang mungkin dihabiskan untuk konstruksi (pengkodean dan pengujian)
- Oleh karena itu, buat hanya model-model yang membuatnya lebih mudah dan lebih cepat untuk membangun perangkat lunak



Prinsip 3. Berusaha Keras untuk Menghasilkan Model Paling Sederhana yang Akan Menggambarkan Problem atau PL

- Jangan membangun PL secara berlebihan
- Dengan menjaga model tetap sederhana, PL yang dihasilkan juga akan sederhana
- Hasilnya adalah PL yang lebih mudah untuk diintegrasikan, lebih mudah untuk diuji, dan lebih mudah untuk dipelihara (diubah)
- Model sederhana lebih mudah dipahami dan dikritik oleh anggota tim PL, menghasilkan bentuk umpan balik berkelanjutan yang mengoptimalkan hasil akhir



Prinsip 4. Bangun model dengan Cara yang Membuatnya Dapat Diubah

- Asumsikan model Anda akan berubah, tetapi dalam membuat asumsi ini jangan asal-asalan (Hal ini karena model persyaratan mungkin tidak cukup lengkap, yang berarti desain (model desain) akan selalu kehilangan fungsi dan fitur penting)



Prinsip 5. Mampu Menyatakan secara Eksplisit Tujuan dari Setiap Model yang Dibuat

- Setiap kali Anda membuat model, tanyakan pada diri sendiri mengapa Anda melakukannya
- Jika Anda tidak dapat memberikan pembenaran yang kuat untuk keberadaan model, jangan habiskan waktu untuk itu

Sepuluh Prinsip PEMODELAN



Prinsip 6. Sesuaikan Model yang Dikembangkan dengan Sistem yang ada

- Mungkin perlu untuk mengadaptasi notasi atau aturan model ke aplikasi; misalnya, aplikasi video game mungkin memerlukan teknik pemodelan yang berbeda dari PL tertanam real-time untuk cruise control adaptif di mobil



Prinsip 7. Cobalah untuk Membuat Model yang Berguna, Lupakan Membangun Model yang Sempurna

- Saat membangun persyaratan dan model desain, seorang Software Engineer mencapai titik pengembalian yang semakin berkurang. Artinya, upaya yang diperlukan untuk membuat model yang lengkap dan konsisten secara internal mungkin tidak sebanding dengan manfaatnya
- Iterasi tanpa henti untuk membuat model "sempurna" tidak memenuhi kebutuhan akan agility



Prinsip 8. Jangan menjadi dogmatis tentang sintaks model. Jika konten berhasil Dikomunikasikan, representasi adalah hal yang kedua

- Karakteristik paling penting dari model ini adalah untuk mengkomunikasikan informasi yang memungkinkan tugas rekayasa perangkat lunak berikutnya.
- Jika model berhasil melakukan ini, sintaks yang salah dapat dimaafkan



Prinsip 9. Jika insting Anda memberi tahu Anda bahwa suatu model tidak benar meskipun tampaknya baik-baik saja di atas kertas, Anda mungkin punya alasan untuk khawatir

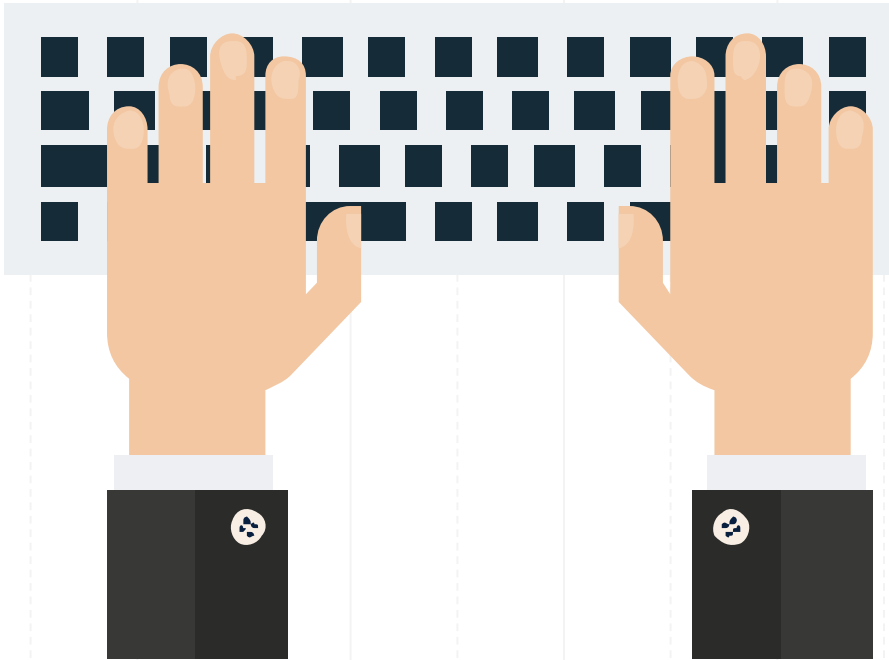
- Jika Anda seorang software engineer yang berpengalaman, percayalah pada insting Anda
- Pekerjaan PL mengajarkan banyak pelajaran - beberapa di antaranya di tingkat bawah sadar.
- Jika ada sesuatu yang memberi tahu Anda bahwa model desain pasti akan gagal (meskipun Anda tidak dapat membuktikannya secara eksplisit), Anda memiliki alasan untuk menghabiskan waktu tambahan untuk memeriksa model atau mengembangkan model yang berbeda



Prinsip 10. Dapatkan Umpan Balik Sesegera Mungkin

- Maksud dari model apapun adalah untuk mengkomunikasikan informasi. Ia harus berdiri sendiri.
- Asumsikan bahwa Anda tidak akan berada di sana untuk menjelaskan modelnya. Setiap model harus ditinjau oleh anggota tim perangkat lunak
- Maksud dari tinjauan ini adalah untuk memberikan umpan balik yang dapat digunakan untuk memperbaiki kesalahan pemodelan, mengubah salah tafsir, dan menambahkan fitur atau fungsi yang dihilangkan secara tidak sengaja.

6.2.4 Construction Principles (Prinsip KONSTRUKSI)

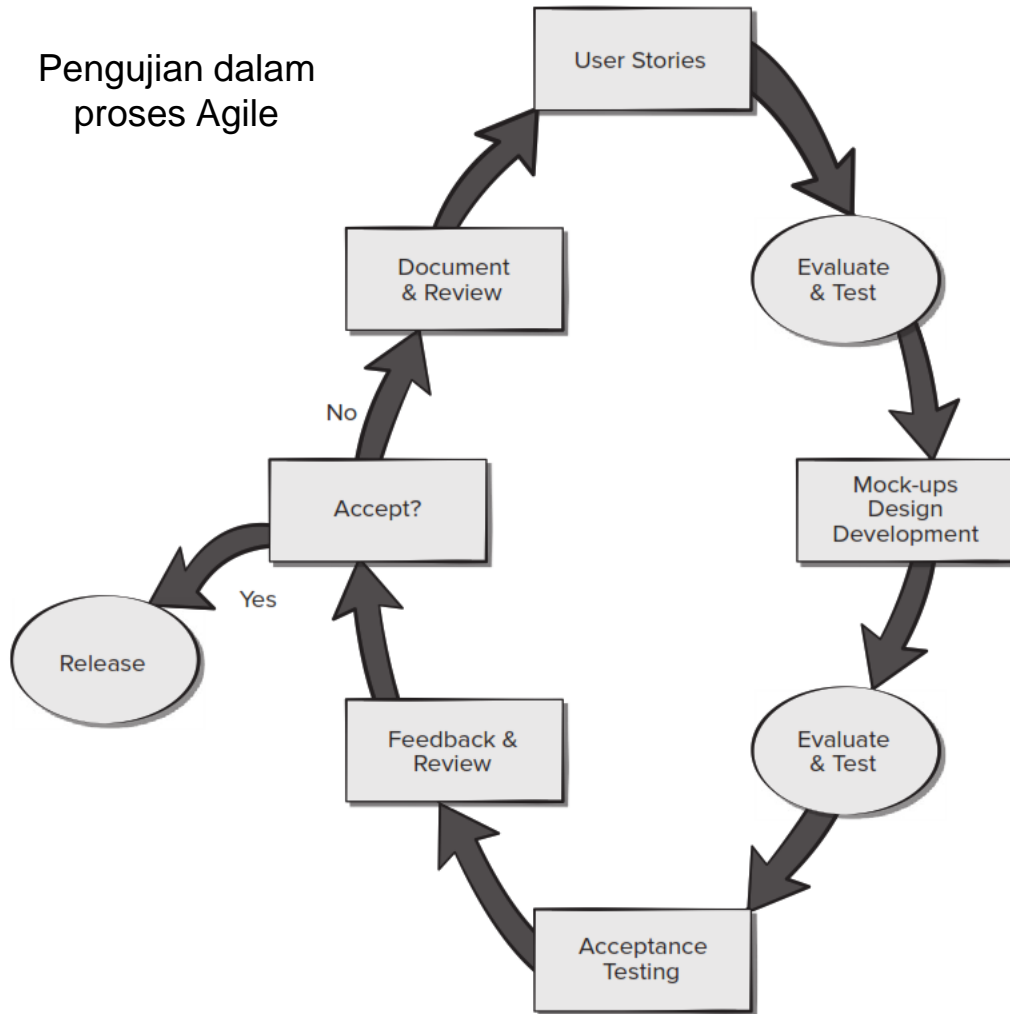


- Aktivitas konstruksi mencakup serangkaian tugas pengkodean dan pengujian yang mengarah ke PL operasional yang siap dikirim ke pelanggan atau pengguna akhir
- Prinsip Konstruksi juga mencakup Prinsip Testing
- Dalam pekerjaan RPL modern, pengkodean dapat berupa:
 - 1) Pembuatan langsung kode sumber (source code) bahasa pemrograman,
 - 2) Pembuatan source code secara otomatis menggunakan desain perantara seperti representasi komponen yang akan dibangun, atau
 - 3) Otomatisasi pembuatan koding yang dapat dieksekusi menggunakan bahasa pemrograman generasi keempat (mis., Unreal4 Blueprints)

Blueprints is a visual scripting tool created by Epic Games (<https://docs.unrealengine.com/latest/INT/Engine/Blueprints/>)

6.2.4 Construction Principles (Prinsip KONSTRUKSI)

Pengujian dalam proses Agile



● Jenis-jenis pengujian (testing)

- 1) **Unit Testing** (Pengujian Unit). Fokus awal pengujian adalah pada tingkat komponen,
- 2) **Integration Testing** (Pengujian Integrasi). Dilakukan saat sistem dibangun,
- 3) **Validation Testing** (Pengujian Validasi). Menilai apakah persyaratan telah dipenuhi untuk sistem yang lengkap (atau peningkatan PL
- 4) **Acceptance Testing** (Pengujian Penerimaan). Dilakukan oleh pelanggan dalam upaya untuk menjalankan semua fitur dan fungsi yang diperlukan

● Gambar di kiri menunjukkan pengujian dan desain kasus uji ditempatkan dalam proses yang agile

6.2.4.1 Prinsip-prinsip KODING

- Prinsip-prinsip yang memandu tugas pengkodean sangat selaras dengan gaya pemrograman, bahasa pemrograman, dan metode pemrograman
- Ada beberapa prinsip dasar yang dapat diikuti:
 - A. Prinsip Preparasi (Preparation Principles)
 - B. Prinsip Koding
 - C. Prinsip Validasi (Validation Principles)

A. Prinsip Preparasi (Preparation Principles)

Sebelum Anda menulis satu baris kode, pastikan Anda

- Prinsip 1.** Pahami masalah yang Anda coba selesaikan
- Prinsip 2.** Pahami prinsip dan konsep desain dasar
- Prinsip 3.** Pilih bahasa pemrograman yang memenuhi kebutuhan perangkat lunak yang akan dibangun dan lingkungan di mana ia akan beroperasi
- Prinsip 4.** Pilih lingkungan pemrograman yang menyediakan alat yang akan membuat pekerjaan Anda lebih mudah
- Prinsip 5.** Buat satu set unit test yang akan diterapkan setelah komponen yang Anda kodekan selesai

6.2.4.1 Prinsip-prinsip KODING

B. Prinsip Koding

Saat Anda mulai menulis kode, pastikan Anda:

- Prinsip 6.** Batasi algoritme Anda dengan mengikuti praktik pemrograman terstruktur
- Prinsip 7.** Pertimbangkan penggunaan pair programming
- Prinsip 8.** Pilih struktur data yang akan memenuhi kebutuhan desain
- Prinsip 9.** Pahami arsitektur PL dan buat antarmuka yang konsisten dengannya

C. Prinsip Validasi

Setelah menyelesaikan coding pass pertama, pastikan Anda:

- Prinsip 10.** Lakukan penelusuran koding bila perlu
- Prinsip 11.** Lakukan pengujian unit dan perbaiki kesalahan yang Anda temukan
- Prinsip 12.** Refactor koding untuk meningkatkan kualitasnya

6.2.4.1 Prinsip-prinsip TESTING

- Tujuan Testing menurut Glen Myers
 - 1) Testing adalah proses mengeksekusi program dengan maksud untuk menemukan error
 - 2) Kasus uji yang baik adalah kasus yang memiliki probabilitas tinggi untuk menemukan error yang belum ditemukan
 - 3) Tes yang berhasil adalah tes yang mengungkap error yang belum ditemukan
- Tujuan Anda adalah merancang tes yang secara sistematis mengungkap kelas error yang berbeda dan melakukannya dengan waktu dan usaha yang minimal
- Pengujian menunjukkan bahwa fungsi PL tampaknya bekerja sesuai dengan spesifikasi dan persyaratan perilaku dan kinerja tampaknya telah terpenuhi
- Data yang dikumpulkan saat pengujian dilakukan memberikan indikasi keandalan PL yang baik dan beberapa indikasi kualitas PL
- Testing tidak dapat menunjukkan tidak adanya kesalahan dan cacat; Testing hanya dapat menunjukkan bahwa ada kesalahan dan cacat perangkat lunak (Gambar 6.6)



6.2.4.1 Prinsip-prinsip TESTING



Prinsip 1. Semua tes harus dapat dilacak ke kebutuhan pelanggan

- Tujuan testing PL adalah untuk mengungkap error
- Oleh karena itu, cacat yang paling parah (dari sudut pandang pelanggan) adalah yang menyebabkan program gagal memenuhi persyaratannya



Prinsip 2. Testing harus direncanakan jauh sebelum testing dimulai

- Perencanaan testing dapat dimulai segera setelah model persyaratan (requirements model) selesai
- Definisi rinci dari kasus uji dapat dimulai segera setelah model desain telah solid
- Oleh karena itu, semua testing dapat direncanakan dan dirancang sebelum koding apa pun dibuat



Prinsip 3. Prinsip Pareto berlaku untuk testing perangkat lunak

- Dalam konteks ini, prinsip Pareto menyiratkan bahwa 80 persen dari semua kesalahan yang ditemukan selama pengujian kemungkinan akan dapat dilacak hingga 20 persen dari semua komponen program
- Masalahnya, tentu saja, adalah mengisolasi komponen yang dicurigai ini dan mengujinya secara menyeluruh



Prinsip 4. Testing harus dimulai "dari yang kecil" dan berlanjut ke testing "dalam skala besar"

- Tes pertama yang direncanakan dan dilaksanakan umumnya fokus pada komponen individu
- Saat testing berlangsung, fokus beralih ke mencari kesalahan dalam kelompok komponen yang terintegrasi dan akhirnya di seluruh sistem



Prinsip 5. Testing menyeluruh tidak mungkin dilakukan

Setiap kali Anda membuat model, tanyakan pada diri sendiri mengapa Anda melakukannya

- Jumlah permutasi jalur bahkan untuk program berukuran sedang sangat besar
- Untuk alasan ini, tidak mungkin untuk mengeksekusi setiap kombinasi jalur selama testing
- Walaupun demikian, hal ini dimungkinkan dilakukan semua, untuk cukup menutupi logika program dan untuk memastikan bahwa semua kondisi dalam desain tingkat komponen telah dilaksanakan

6.2.4.1 Prinsip-prinsip TESTING



Prinsip 6. Terapkan ke setiap modul dalam sistem upaya testing yang sepadan dengan ekspektasi densitas kesalahan

- Ini sering merupakan modul terbaru atau yang paling tidak dipahami oleh pengembang



Prinsip 7. Teknik testing statis dapat memberikan hasil yang tinggi

- Lebih dari 85 persen cacat PL berasal dari dokumentasi PL (persyaratan, spesifikasi, penelusuran kode, dan manual pengguna)
- Mungkin ada manfaatnya untuk menguji dokumentasi sistem



Prinsip 8. Lacak cacat dan cari pola cacat yang ditemukan dengan testing

- Cacat total yang ditemukan merupakan indikator kualitas PL yang baik
- Jenis cacat yang ditemukan dapat menjadi ukuran stabilitas PL yang baik
- Pola cacat yang ditemukan dari waktu ke waktu dapat memperkirakan jumlah cacat yang diharapkan



Prinsip 9. Sertakan kasus uji yang menunjukkan PL berperilaku dengan benar

- Saat komponen PL dipertahankan atau diadaptasi, interaksi tak terduga menyebabkan efek samping yang tidak diinginkan pada komponen lain
- Penting untuk memiliki satu set kasus uji regresi yang siap untuk memeriksa perilaku sistem setelah perubahan dilakukan pada produk PL

6.2.5 Deployment Principles (Prinsip Deployment)

- Aktivitas deployment terdiri atas: delivery, support, dan feedback
- Model proses PL modern bersifat evolusioner atau inkremental, sehingga deployment terjadi tidak hanya sekali, tetapi beberapa kali saat PL bergerak mengarah ke penyelesaian
- Setiap siklus pengiriman (delivery) memberi pelanggan dan pengguna akhir peningkatan PL operasional yang menyediakan fungsi dan fitur yang dapat digunakan
- Setiap siklus dukungan (support) menyediakan dokumentasi dan bantuan manusia untuk semua fungsi dan fitur yang diperkenalkan selama semua siklus deployment
- Setiap siklus umpan balik (feedback) menyediakan tim PL dengan panduan penting yang menghasilkan modifikasi pada fungsi, fitur, dan pendekatan yang diambil untuk increment berikutnya
- Pengiriman increment PL merupakan tonggak penting untuk setiap proyek perangkat lunak
- Aksi deployment tipikal diilustrasikan pada Gambar 6.7



Gambar 6.7 Deployment Actions

6.2.5 Deployment Principles (Prinsip Deployment)



Prinsip 1. Ekspektasi pelanggan untuk perangkat lunak harus dikelola

- Terlalu sering, pelanggan mengharapkan lebih dari yang dijanjikan tim untuk delivery, dan kekecewaan segera terjadi
- Ini menghasilkan umpan balik yang tidak produktif dan merusak moral tim
- Dalam bukunya tentang mengelola ekspektasi, Naomi Karten menyatakan:
 - "Titik awal untuk mengelola ekspektasi adalah menjadi lebih teliti tentang apa yang Anda komunikasikan dan caranya."
 - Seorang software engineer harus berhati-hati dalam mengirim pesan yang bertentangan kepada pelanggan (mis., menjanjikan lebih dari yang dapat Anda berikan secara wajar dalam periode waktu yang disediakan atau memberikan lebih dari yang Anda janjikan untuk satu increment perangkat lunak dan kemudian kurang dari yang dijanjikan untuk yang berikutnya)



Prinsip 2. Paket pengiriman (delivery) lengkap harus dirakit dan diuji

- Semua PL yang dapat dijalankan, file data dukungan, dokumen dukungan, dan informasi relevan lainnya harus dirakit dan diuji beta secara menyeluruh dengan pengguna sebenarnya
- Semua skrip penginstalan dan fitur operasional lainnya harus dijalankan secara menyeluruh di semua konfigurasi komputasi yang mungkin (yaitu, perangkat keras, sistem operasi, perangkat periferal, pengaturan jaringan)



Prinsip 3. Regimen dukungan harus ditetapkan sebelum PL dikirimkan

- Pengguna akhir mengharapkan respon dan informasi yang akurat ketika pertanyaan atau masalah muncul
- Jika dukungan bersifat ad hoc, atau lebih buruk lagi, tidak ada, pelanggan akan langsung merasa tidak puas
- Dukungan harus direncanakan, bahan pendukung harus disiapkan, dan mekanisme pencatatan yang tepat harus ditetapkan sehingga tim PL dapat melakukan penilaian kategoris dari jenis dukungan yang diminta

6.2.5 Deployment Principles (Prinsip Deployment)



Prinsip 4. Instruksional yang memadai harus disediakan untuk pengguna akhir

- Tim PL memberikan lebih dari PL itu sendiri
- Alat bantu pelatihan (training aids) yang sesuai (jika diperlukan) harus dikembangkan; pedoman pemecahan masalah (troubleshooting) harus disediakan
- Bila perlu, deskripsi "apa yang berbeda tentang increment perangkat lunak ini" harus dipublikasikan/disampaikan



Prinsip 5. Software buggy harus diperbaiki dulu, dikirim nanti

- Di bawah tekanan waktu, beberapa organisasi perangkat lunak memberikan peningkatan kualitas rendah dengan peringatan kepada pelanggan bahwa bug "akan diperbaiki pada rilis berikutnya." Ini adalah kesalahan
- Ada pepatah dalam bisnis PL: *"Pelanggan akan lupa bahwa Anda mengirimkan produk berkualitas tinggi beberapa hari terlambat, tetapi mereka tidak akan pernah melupakan masalah yang disebabkan oleh produk berkualitas rendah. Perangkat lunak berkualitas rendah mengingatkan mereka setiap hari."*

The background of the slide features a scenic view of a mountain range during sunset. The sky is a gradient of warm colors, from light orange at the top to a deeper orange near the horizon. The mountains are silhouetted against the sky, with the foreground mountains in a darker blue and the distant peaks in a lighter blue. A series of vertical dashed white lines are overlaid on the entire image, creating a grid-like effect.

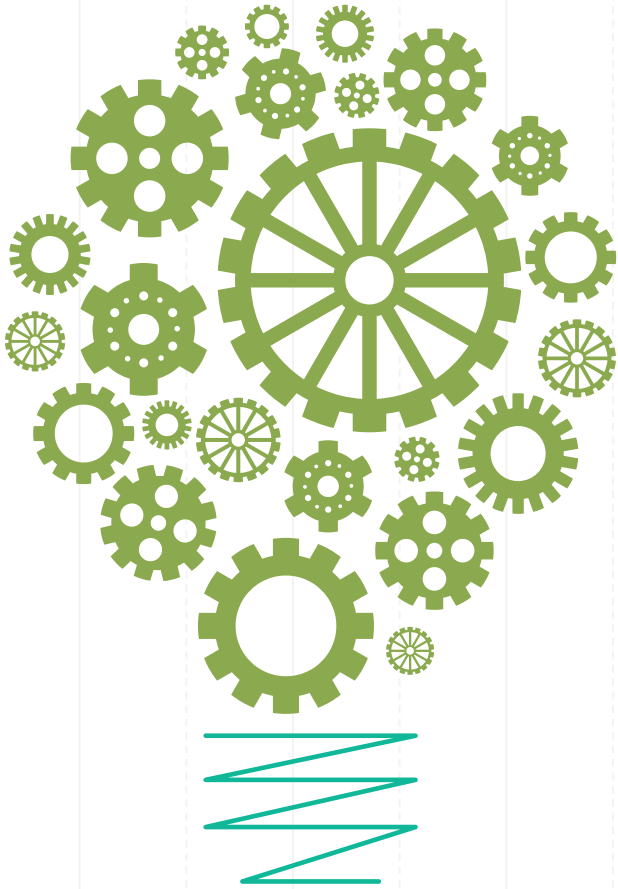
6.3 Rangkuman

Rangkuman

- **Praktik RPL** mencakup prinsip, konsep, metode, dan alat yang diterapkan oleh insinyur perangkat lunak selama proses perangkat lunak.
- **Setiap proyek RPL berbeda.**
- Namun, **seperangkat prinsip umum** berlaku untuk proses dan praktik setiap framework aktivitas terlepas dari proyek atau produknya
- **Serangkaian prinsip inti** membantu dalam penerapan proses PL yang bermakna dan pelaksanaan metode RPL yang efektif.
- Pada tingkat proses, prinsip-prinsip inti membangun landasan filosofis yang memandu tim PL saat menavigasi melalui proses PL.
- Pada tingkat praktik, prinsip inti menetapkan kumpulan nilai dan aturan yang berfungsi sebagai panduan saat Anda menganalisis problem, merancang solusi, menerapkan dan menguji solusi, dan akhirnya menerapkan perangkat lunak di komunitas pengguna.
- **Prinsip-prinsip komunikasi** berfokus pada kebutuhan untuk mengurangi kebisingan dan meningkatkan bandwidth saat percakapan antara pengembang dan pelanggan berlangsung.
- Kedua belah pihak harus berkolaborasi agar komunikasi yang terbaik terjadi

Rangkuman...

- **Prinsip-prinsip perencanaan** memberikan pedoman untuk membangun peta terbaik untuk perjalanan ke sistem atau produk yang telah selesai.
- Rencana tersebut dapat dirancang hanya untuk peningkatan PL tunggal, atau dapat ditentukan untuk keseluruhan proyek.
- Apapun, itu harus membahas apa yang akan dilakukan, siapa yang akan melakukannya, dan kapan pekerjaan itu akan selesai
- **Prinsip pemodelan** berfungsi sebagai dasar untuk metode dan notasi yang digunakan untuk membuat PL.
- Pemodelan meliputi analisis dan desain, menggambarkan representasi PL yang semakin menjadi lebih rinci.
- Maksud dari model ini adalah untuk memperkuat pemahaman tentang pekerjaan yang harus dilakukan dan untuk memberikan bimbingan teknis kepada mereka yang akan mengimplementasikan PL
- **Konstruksi** menggabungkan siklus pengkodean dan pengujian di mana kode sumber untuk komponen dihasilkan dan diuji.
- Prinsip pengkodean mendefinisikan tindakan umum yang harus terjadi sebelum kode ditulis, saat dibuat, dan setelah selesai.
- Meskipun ada banyak prinsip pengujian, hanya satu yang dominan: Pengujian adalah proses mengeksekusi program dengan maksud untuk menemukan error



Rangkuman...

- **Deployment** terjadi karena setiap increment PL disajikan kepada pelanggan dan mencakup pengiriman (delivery), dukungan (support), dan umpan balik (feedback).
- Prinsip-prinsip utama untuk pengiriman mempertimbangkan pengelolaan harapan pelanggan dan menyediakan pelanggan dengan informasi dukungan yang sesuai untuk perangkat lunak.
- Dukungan menuntut persiapan terlebih dahulu.
- Umpan balik memungkinkan pelanggan untuk menyarankan perubahan yang memiliki nilai bisnis dan memberikan masukan kepada pengembang untuk siklus RPL berulang berikutnya



Pertemuan 6: Principles that Guide Practices

Meuthia Rachmaniah
Departemen Ilmu Komputer, FMIPA IPB

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