Universitas Padjadjaran

Pengenalan Rekayasa Perangkat Lunak (*Software Engineering*)

Dr. Asep Sholahuddin, MT
Mira Suryani, S.Pd., M.Kom
Rahmatullah, Skom., MT.
S-1 Teknik Informatika

Tujuan Pembelajaran

Setelah mempelajari materi mengenai topik ini diharapkan mahasiswa mampu:

- Menjelaskan macam-macam jenis dan siklus proses pengembangan perangkat lunak.
- Mengetahui dasar-dasar menjaga kualitas dari perangkat lunak yang dikembangkan
- Mengetahui dasar-dasar bagaimana mengelola project pengembangan perangkat lunak.

Pokok Bahasan

- Pengenalan *Software*
- Pengertian *Software Engineering*
- Proses Generic *Software Engineering*
- Metodologi Pengembangan *Software*

Pengenalan Software

Apa itu Software?

Perangkat lunak (*software*) adalah:

1. Instruksi (program komputer) yang ketika dijalankan menyediakan fitur, fungsi, dan kinerja yang diinginkan;
2. Struktur data yang memungkinkan program untuk memanipulasi informasi secara memadai; dan
3. Informasi deskriptif baik dalam bentuk *hard copy* dan virtual yang menjelaskan pengoperasian dan penggunaan program.

Karakteristik Software

- Software dikembangkan atau direkayasa, tidak diproduksi dalam pengertian klasik.
- Software tidak “usang”
- Meskipun industri bergerak menuju konstruksi berbasis komponen, sebagian besar perangkat lunak terus dibuat khusus.

Jenis-jenis *Software*

Terdapat 7 jenis software berdasarkan domainnya:

- System Software
- Application Software
- Engineering/Scientific Software
- Embedded Software
- Product-line Software
- Web Applications
- Artificial Intelligence Software

Jenis Software

System Software

- Kumpulan program yang ditulis untuk melayani atau mengembangkan program lain.
- Beberapa proses perangkat lunak memiliki sistem rumit, tetapi mempunyai struktur informasi tertentu. (Contoh: compilers, editors, dan file management utilities)
- Terdapat software yang bertugas mengolah data tak tentu (Contoh: operating system components, drivers, networking software, telecommunications processors)
- Karakteristik *System Software*:
  1. Interaksi yang beratdengan perangkat keras komputer;
  2. Penggunaan berat oleh banyak pengguna;
  3. Operasi bersamaan yang membutuhkan penjadwalan,
  4. Berbagi sumber daya, dan
  5. Manajemen proses yang canggih;
  6. Struktur data yang kompleks;
  7. Beberapa antarmuka eksternal.

Jenis Software

Application Software

- Stand-alone program yang dibuat untuk memecahkan kebutuhan bisnis tertentu
- Aplikasi di area ini memproses data bisnis atau teknis dengan cara yang memfasilitasi operasi bisnis atau pengambilan keputusan manajemen/teknis.
- Application software digunakan untuk mengontrol fungsi bisnis secara *real time*
- Contoh: point-of-sale, transaction processing, real-time manufacturing process control

Jenis Software

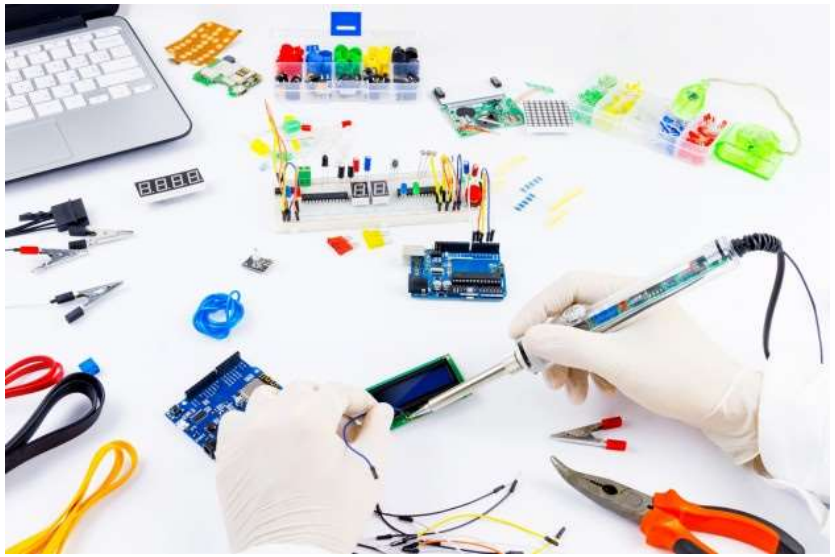
Engineering/Scientific Software

- Karakteristiknya berupa algoritma untuk “menghitung angka”
- Contoh: aplikasi di bidang molekular biologi, astronomi, vulkanologi, dll
- Computer-aided design, system simulation, dan aplikasi interaktif lainnya menjadi karakteristik utama dari software ini

Jenis Software

Embedded Software

- Software yang ditanamkan ke dalam produk atau sistem dan digunakan untuk mengimplementasikan dan mengontrol fitur dan fungsi untuk pengguna akhir dan untuk sistem itu sendiri
- Software ini dapat menjalankan fungsi terbatas dan esoteric (Contoh: kontrol tombol untuk oven microwave)
- Dapat juga memberikan fungsi yang signifikan dan kemampuan control (Contoh: fungsi digital dalam mobil seperti bahan bakar kontrol, tampilan dasbor, dan sistem pengereman)



Jenis Software

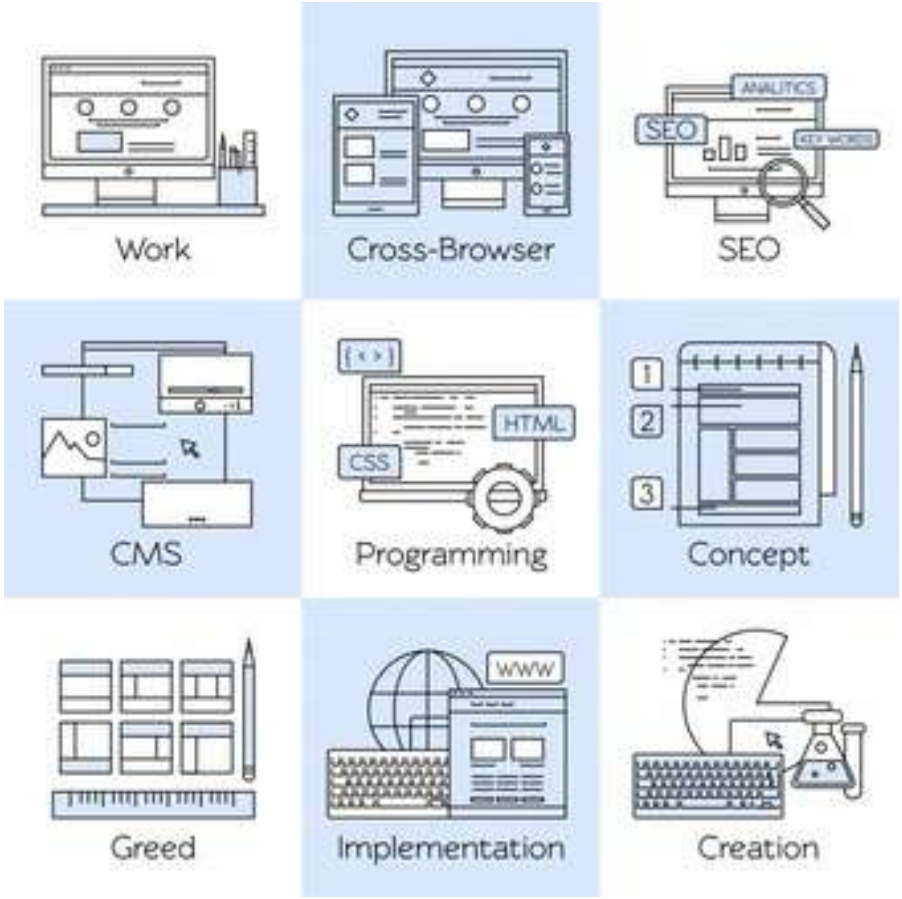
Product-line Software

- Dirancang untuk memberikan kemampuan khusus untuk digunakan oleh banyak pelanggan yang berbeda
- Software ini berfokus pada pasar yang terbatas dan esoteric (Contoh: produk pengendalian persediaan)
- Dapat juga membahas pasar konsumen massal (Contoh: word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, serta personal and business financial applications)

Jenis Software

Web Applications

- Kategori perangkat lunak yang berpusat pada jaringan ini mencakup beragam aplikasi
- WebApps berkembang menjadi lingkungan komputasi canggih, dapat menyediakan fitur yang berdiri sendiri, fungsi komputasi, dan konten kepada pengguna akhir
- WebApps juga terintegrasi dengan database perusahaan dan aplikasi bisnis.



Jenis Software

Artificial Intelligence Software

- AI Software memanfaatkan algoritma nonnumerik untuk memecahkan masalah kompleks yang tidak sesuai dengan komputasi atau analisis langsung.
- Aplikasi dalam area ini termasuk robotika, sistem pakar, pengenalan pola (gambar dan suara), jaringan saraf tiruan, pembuktian teorema, dan *game*.

Pengenalan Rekayasa Perangkat Lunak (*Software Engineering*)

Rekayasa Perangkat Lunak (RPL)

- Berdasarkan IEEE, Rekayasa perangkat lunak adalah pengaplikasian pendekatan yang sistematis, disiplin, dan terukur untuk pengembangan, pengoperasian, dan pemeliharaan perangkat lunak.
- RPL merupakan pembentukan dan penggunaan prinsip-prinsip rekayasa untuk mendapatkan perangkat lunak yang dapat diandalkan secara ekonomis dan bekerja secara efisien pada mesin nyata.

Rekayasa Perangkat Lunak (RPL)

Menurut Beekman, RPL adalah :

1. Proses pemecahan masalah dalam menyelidiki situasi;
2. Merancang solusi sistem untuk memperbaiki situasi;
3. Memperoleh sumber daya manusia, keuangan, dan teknologi untuk mengimplementasikan solusi; dan
4. Mengevaluasi keberhasilan solusi tersebut.

Pengembangan Software

- Setiap proyek baru membutuhkan orang, uang, dan sumber daya organisasi lainnya. Hal tersebut ditentukan oleh *steering committee* (SC).
- Tim proyek biasanya mencakup satu atau lebih end-user dan system analyst.
- *End-user* (Pengguna akhir) adalah orang yang menggunakan sistem informasi secara langsung atau menggunakan informasi yang dihasilkan oleh sistem
- *System analyst* adalah seorang profesional TI yang terutama bertanggung jawab untuk mengembangkan dan mengelola sistem.

Pentingnya Rekayasa Perangkat Lunak

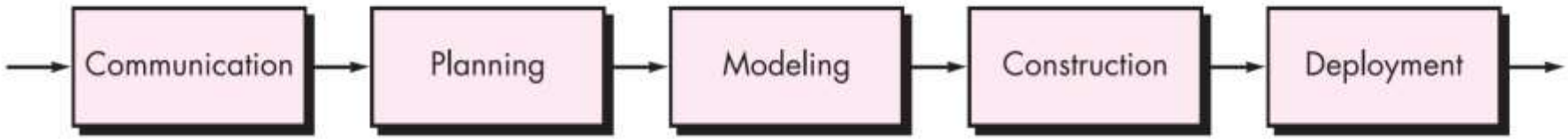
Rekayasa perangkat lunak semakin diperlukan karena hal-hal berikut :

- Ukuran software makin besar.
- Akan lebih mudah untuk menskalakan software yang sudah ada ketimbang membuat ulang.
- Menjaga agar harga software tidak mahal
- Membuat software punya kualitas yang lebih baik

Proses Pengembangan Perangkat Lunak

Proses Pengembangan Perangkat Lunak

- Dalam rekayasa perangkat lunak, diperlukan pembagian kerja menjadi beberapa fase atau tahap dalam aktivitasnya supaya dapat direncanakan dan di-manage dengan baik.
- Hal tersebut dinamakan software development methodology / *software development life cycle* (SDLC).
- Tahapan-tahapan generik dalam SDLC:



Sumber: Roger S. Pressman. Software Engineering: A Practitioner's Approach 7th Edition. Mc Graw Hill Higher Education. 2010.

- Tahapan *maintenance* dan *retirement* saat ini sudah menjadi tahapan generik melengkapi tahapan sebelumnya.

Tahap RPL

Communication/Investigation

- Tahap ini bertujuan:
  1. Mempelajari masalah atau peluang bisnis yang ada
  2. Menentukan apakah layak untuk mengembangkan sistem baru atau mendesain ulang yang ada.
- Tim melakukan studi kelayakan untuk mengidentifikasi sifat masalah
- Tim mencoba menjawab beberapa pertanyaan kelayakan:
  1. Kelayakan teknis (hardware, software, keamanan, akurasi, dll)
  2. Kelayakan ekonomi (investasi bagus/tidak, perbandingan biaya pengembangan dan keuntungan)
  3. Kelayakan operasional (sesuai kebutuhan, terjadwal baik)
  4. Kelayakan organisasi (aplikasi mendukung tujuan, aspek legal, dll)

Tahap RPL

Planning/Analysis

Selama fase perencanaan/analisis, analis sistem:

- Mengumpulkan dokumen,
- Mewawancarai pengguna sistem saat ini (jika ada),
- Mengamati sistem dalam tindakan, dan
- Mengumpulkan serta menganalisis data untuk memahami sistem saat ini dan mengidentifikasi persyaratan baru — fitur atau kemampuan yang harus disertakan dalam sistem untuk memenuhi kebutuhan pengguna.

Tahap RPL
Planning/Analysis

Kebutuhan yang perlu diidentifikasi dalam pengembangan software:

- **Kebutuhan input / output**. Karakteristik antarmuka pengguna, termasuk konten, format, dan persyaratan waktu untuk layar entri data dan laporan manajerial
- **Kebutuhan pemrosesan**. Perhitungan, aturan keputusan, kapasitas pemrosesan data, dan waktu respons yang dibutuhkan
- **Kebutuhan penyimpanan**. Isi catatan dan database serta prosedur pengambilan data
- **Kebutuhan kontrol**. Akurasi, validitas, dan keamanan sistem yang diinginkan; misalnya, untuk mencegah kesalahan entri data dan menjamin sistem yang mudah digunakan dan ramah pengguna

Tahap RPL

Design/Modeling

- Tahap desain/pemodelan berfokus pada bagaimana melakukan pengembangan
- Analis sistem mengembangkan spesifikasi yang menggambarkan bagaimana kebutuhan sistem, yang diidentifikasi dalam fase analisis, bisa dipenuhi.
- Terdapat 3 desain yang perlu dibuat:
  1. *User Interface Design*
  2. *Database Design*
  3. *Process Design*
- Pada tahap ini, sistem analis bisa membuat prototipe
- Prototipe adalah sistem kerja terbatas yang memberi pengguna dan manajemen gambaran tentang bagaimana sistem yang sudah selesai akan bekerja.

Tahap RPL

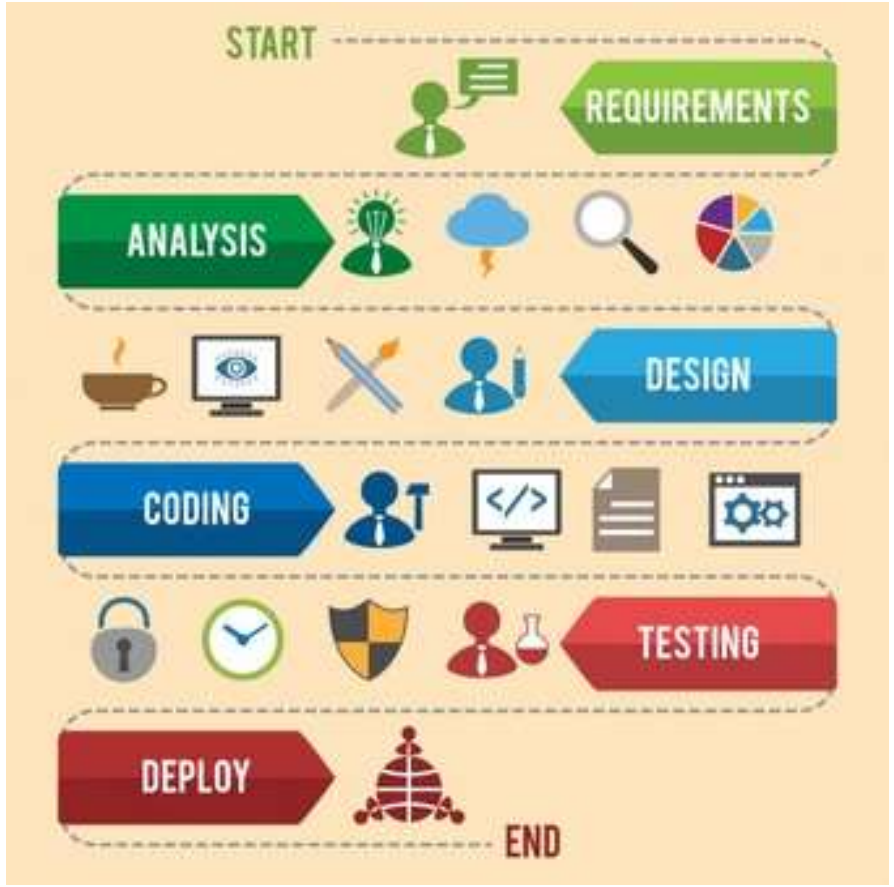
Construction/Development

- Tahap konstruksi/pengembangan merupakan proses pengubahan spesifikasi desain menjadi sistem kerja nyata.
- Pengembangan mencakup campuran kompleks dari penjadwalan, pembelian, instalasi, dokumentasi, dan pemrograman
- Untuk proyek besar, tahap pengembangan melibatkan tim programmer, penulis teknis, dan orang-orang administrasi di bawah pengawasan analis sistem.
- Sebagian besar dari jadwal pengembangan dikhususkan untuk menguji sistem.
- Anggota tim pengembangan sistem melakukan pengujian awal untuk menemukan dan menghilangkan bug (alpha testing).
- Tes berikutnya melibatkan potensial end-user (beta testing).

Tahap RPL

Deployment/Implementation

- Tahap implementasi terjadi ketika fase pengujian selesai dan sistem baru siap menggantikan yang lama.
- Pada software komersial, tahap ini biasanya melibatkan pelatihan ekstensif dan dukungan teknis untuk melengkapi upaya penjualan dan pemasaran.
- Untuk sistem kustom yang besar, implementasi mencakup pendidikan dan pelatihan pengguna akhir, penggantian peralatan, konversi file, dan pemantauan masalah untuk sistem baru.



Tahap RPL

Maintenance

- Tahap pemeliharaan (*maintenance*) melibatkan pemantauan, evaluasi, perbaikan, dan peningkatan sistem selama masa pakai sistem
- Permasalahan software biasanya muncul ketika sistem telah digunakan beberapa saat atau kebutuhan organisasi berubah
- Evaluasi merupakan aspek penting dari perawatan
- Sistem dievaluasi secara berkala untuk menentukan apakah memberikan manfaat dan sesuai dengan kebutuhan organisasi

Tahap RPL

Retirement

- Di beberapa titik dalam kehidupan suatu sistem, pemeliharaan berkelanjutan tidaklah cukup
- Penyebabnya: perubahan kebutuhan organisasi, ekspektasi pengguna, perubahan teknologi, peningkatan biaya pemeliharaan, dan faktor lainnya
- Pada tahapan ini software yang digunakan perlu diberhentikan
- Saatnya untuk meluncurkan penyelidikan untuk sistem yang lebih baru, memulai kembali project RPL.

Metodologi Pengembangan Perangkat Lunak

Model/Metodologi Pengembangan Perangkat Lunak

Dalam SDLC, setidaknya terdapat beberapa 3 kelompok metodologi yang berkembang yaitu prescriptive (plan-driven), agile (value-driven), dan specialized (formal method).

Perbedaan Karakteristik Kelompok Metodologi SDLC

Value-driven methods	Plan-driven methods	Formal methods
Tingkat Kritis: Rendah	Tingkat Kritis: Tinggi	Tingkat Kritis: Extreme
Pengembang Senior	Pengembang Junior	Pengembang Senior
Kebutuhan sering berubah	Kebutuhan tidak sering berubah	Kebutuhan dan fitur terbatas
Jumlah developer sedikit	Jumlah developer banyak	Kebutuhan dapat dimodelkan
Budaya yang merespon perubahan	Budaya berdasarkan permintaan pesanan	Kualitas Extreme

Presciptive Methogology

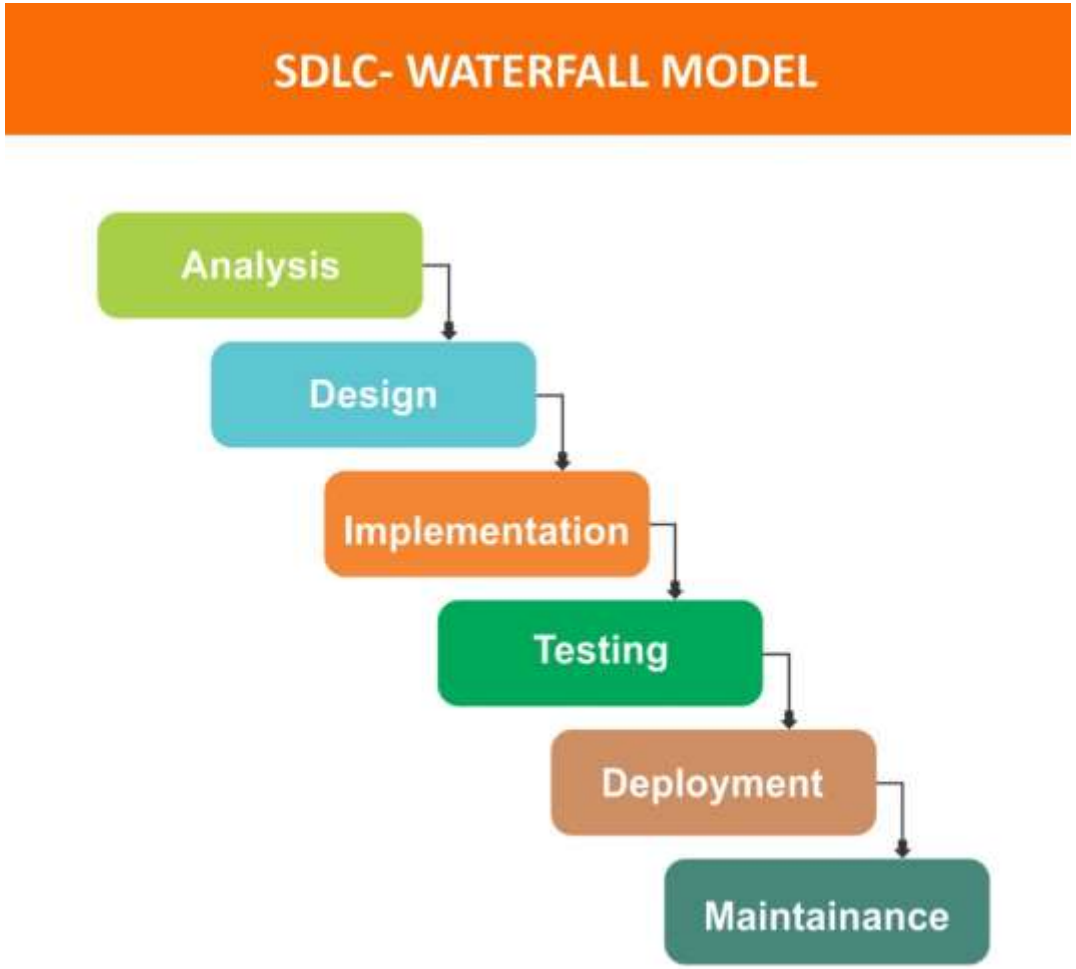
Pada kelompok prescriptive, terdapat beberapa metode yang umum digunakan dalam pembuatan perangkat lunak antara lain:

1. Waterfall
2. Iterative
3. Spiral
4. Rapid Application Development

Metodologi RPL

Waterfall

- Metode waterfall merupakan metode pengembangan software secara berurutan tiap langkahnya.
- Setiap langkah harus diselesaikan terlebih dahulu untuk maju ke tahap berikutnya.

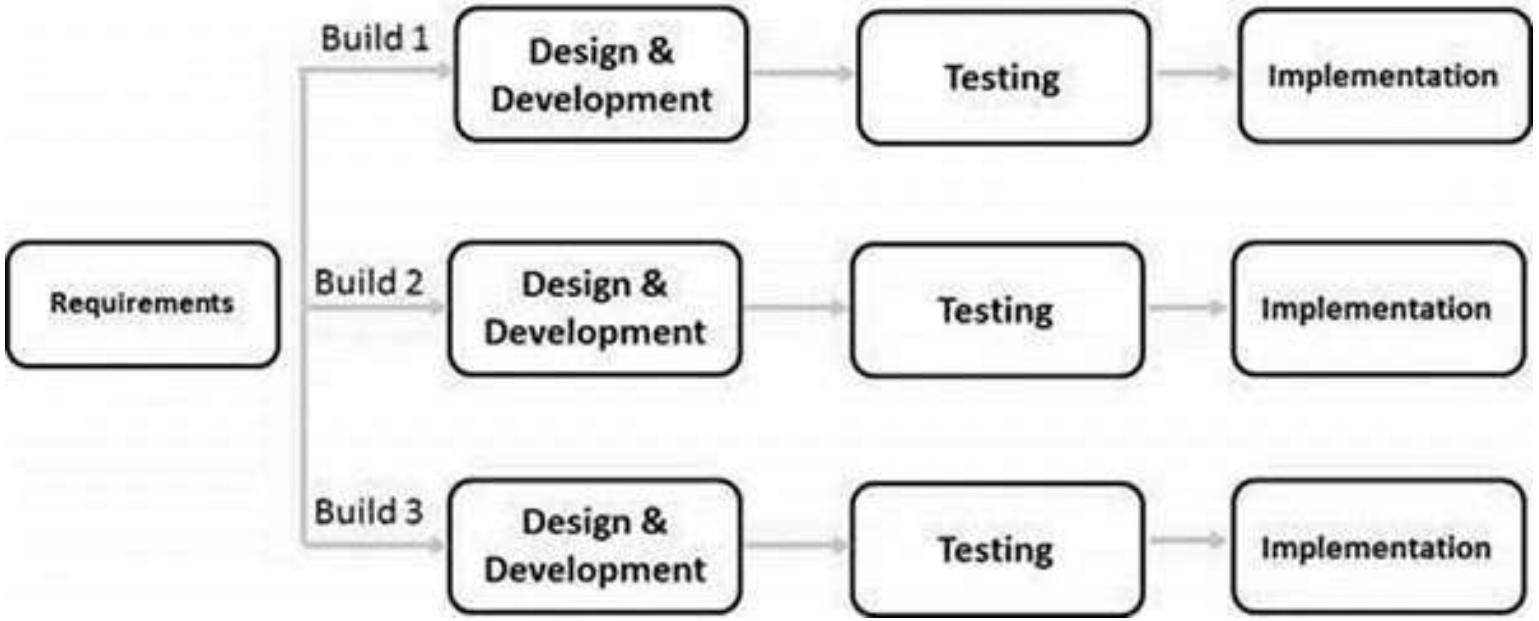


Sumber: medium.com

Metodologi RPL

Iterative

Dalam metode ini, proses dimulai dengan implementasi sederhana dari serangkaian kecil dari *software requirements* dan secara bertahap dikembangkan sampai sistem lengkap diimplementasikan dan siap untuk digunakan

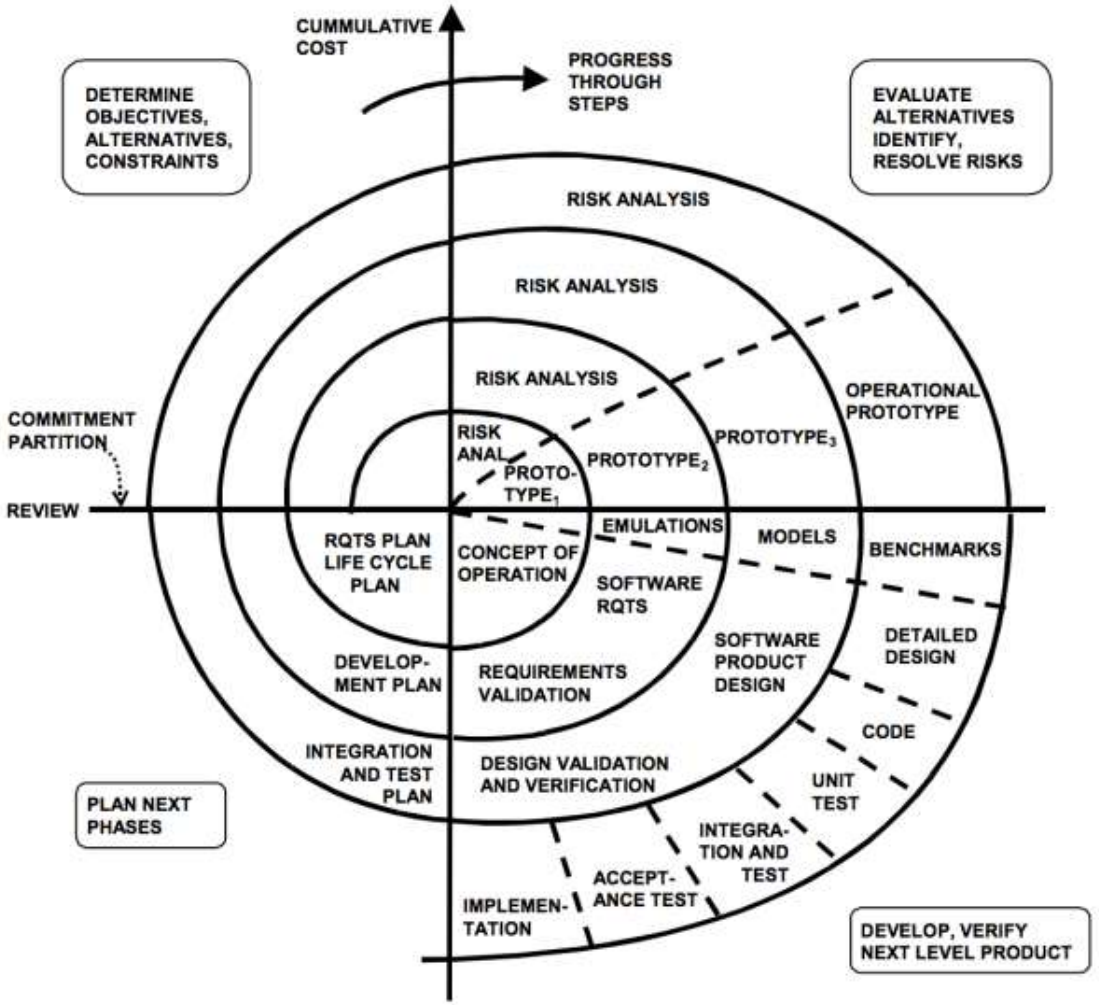


Sumber : https://www.tutorialspoint.com/sdlc/images/sdlc_iterative_model.jpg

Metodologi RPL

Iterative

Metode spiral merupakan kombinasi antara metode *iterative* dengan metode *waterfall* dengan penekanan yang tinggi terhadap analisis resiko.

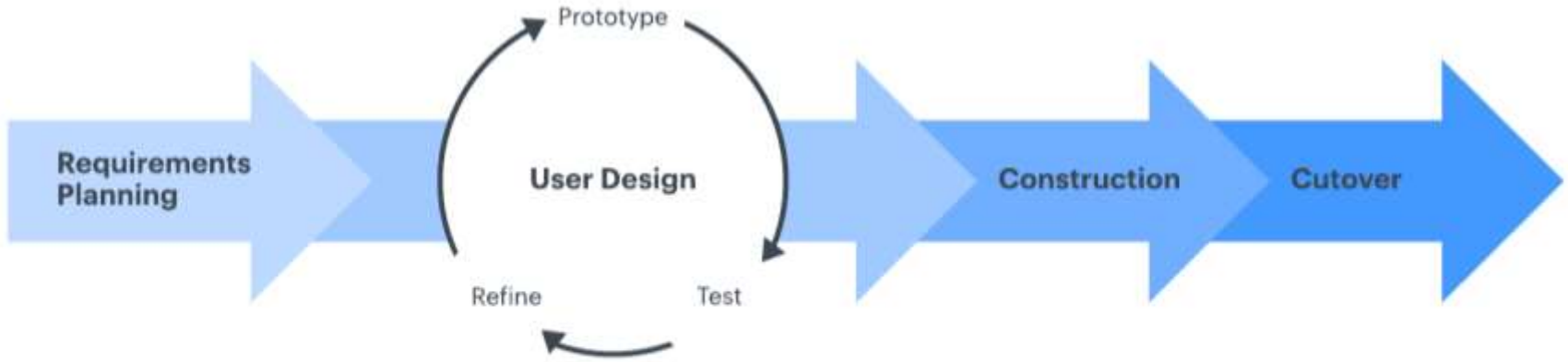


Sumber : *Barry Boehm, & Wilfred J. Hansen http://www.sei.cmu.edu/reports/00sr008.pdf*

Metodologi RPL

Rapid Application Development

RAD: Metode RPL yang menggunakan metode iterative dan pembangunan prototipe yang cepat dan terus-menerus tanpa perencanaan mendetail.



Agile Development

- Agile adalah kemampuan untuk membuat dan merespons perubahan. Ini adalah cara untuk menghadapi, dan pada akhirnya berhasil, dalam lingkungan yang tidak pasti dan bergejolak.
- Pengembangan perangkat lunak Agile adalah istilah umum untuk serangkaian kerangka kerja dan praktik berdasarkan nilai dan prinsip yang diungkapkan dalam manifesto untuk Pengembangan Perangkat Lunak Agile dan 12 Prinsip di belakangnya.
- Satu hal yang membedakan Agile dari pendekatan lain untuk pengembangan perangkat lunak adalah fokus pada orang yang melakukan pekerjaan dan cara mereka bekerja sama.

Manifesto Agile Development

Manifesto untuk Pengembangan Perangkat Lunak Agile didasarkan pada dua belas prinsip:

1. Kepuasan pelanggan dengan pengiriman perangkat lunak berharga secara dini dan berkelanjutan.
2. Menyambut persyaratan yang berubah, bahkan dalam pengembangan yang terlambat.
3. Sering mengirimkan perangkat lunak yang berfungsi (beberapa minggu, bukan bulan)
4. Keeratan, kerja sama keseharian antara pebisnis dan developer
5. Proyek dibangun di sekitar individu yang termotivasi, yang harus dipercaya
6. Percakapan tatap muka (face-to-face) adalah bentuk komunikasi terbaik (lokasi bersama)

Manifesto Agile Development

7. Pengerjaan software merupakan ukuran keberhasilan yang utama
8. Pembangunan berkelanjutan, mampu mempertahankan kecepatan yang konstan
9. Perhatian terus menerus terhadap keunggulan teknis dan desain yang baik
10. Kesederhanaan — seni memaksimalkan jumlah pekerjaan yang belum diselesaikan — sangat penting
11. Arsitektur, kebutuhan, dan desain terbaik muncul dari tim yang mengatur dirinya sendiri
12. Secara teratur, tim merefleksikan bagaimana menjadi lebih efektif, dan menyesuaikannya

Metode SDLC Agile

Framework	Main contributor(s)
Adaptive software development (ASD)	Jim Highsmith, Sam Bayer
Agile modeling	Scott Ambler, Robert Cecil Martin
Agile unified process (AUP)	Scott Ambler
Disciplined agile delivery	Scott Ambler
Dynamic systems development method (DSDM)	
Extreme programming (XP)	Kent Beck, Robert Cecil Martin
Feature-driven development (FDD)	Jeff De Luca
Lean software development	Mary Poppendieck, Tom Poppendieck
Lean startup	Eric Ries
Kanban	Taiichi Ohno
Rapid application development (RAD)	James Martin
Scrum	Ken Schwaber, Jeff Sutherland
Scrumban	
Scaled Agile Framework - SAFe	Scaled Agile, Inc.

Sumber: https://en.wikipedia.org/wiki/Agile_software_development

Kesalahan Umum dalam Implementasi Metodologi Agile

Seringkali *agile development* gagal diaplikasikan dalam project, hal ini disebabkan:

- Kurangnya desain produk secara keseluruhan
- Menambahkan stories ke iterasi yang sedang berlangsung
- Kurangnya dukungan sponsor
- Pelatihan yang tidak memadai
- Peran pemilik produk (*product owner*) tidak diisi dengan benar
- Tim tidak fokus
- Persiapan / perencanaan yang berlebihan
- Pemecahan masalah dalam standup harian
- Menetapkan tugas
- Scrum master sebagai contributor
- Kurangnya otomatisasi pengujian
- Pengembang kelelahan, dll

Alat & Teknik Pengembangan Software

- Analis sistem menggunakan berbagai alat dan teknik sepanjang siklus hidup pengembangan sistem.
- Beberapa digunakan untuk mengumpulkan data;
- Beberapa digunakan untuk mendeskripsikan atau merancang sistem, dan
- Beberapa digunakan untuk mendokumentasikan sistem dalam laporan.

Teknik Pengumpulan Data

Teknik pengumpulan data dapat digunakan selama setiap tahap SDLC. Teknik tersebut, antara lain:

- Review dokumen
- Interview
- Kuisioner
- Observasi
- Sampling

Alat untuk Modeling/Design

- Alat pemodelan adalah representasi grafis dari suatu sistem.
- Banyak alat seperti itu tersedia, tetapi alat pemodelan yang paling banyak digunakan oleh analis sistem adalah:

1. Diagram alir sistem (*flowchart*),
2. Diagram aliran data (data flow diagram),
3. Kamus data (data dictionary), dan
4. Tabel keputusan (decision table).

Computer-Aided Systems Engineering (CASE)

- Saat ini banyak alat dan teknik pengembangan sistem disertakan dalam paket software yang tersedia secara komersial disebut Computer-Aided System Engineering (CASE)
- CASE aplikasi biasanya terdiri dari:
  1. Alat untuk menggambar diagram alir sistem dan diagram aliran data
  2. Kamus data terpusat yang berisi informasi rinci tentang semua komponen sistem
  3. Generator antarmuka pengguna untuk membuat dan mengevaluasi berbagai desain antarmuka
  4. Generator kode yang mengotomatiskan banyak pemrograman komputer untuk membuat sistem atau aplikasi baru
- Contoh: Microsoft Visual Studio

Daftar Pustaka

- AgileAlliance. Agile 101. Retrieved 16 August 2020. Online: https://www.agilealliance.org/agile101/
- Boehm, B.; R. Turner (2004). Balancing Agility and Discipline: A Guide for the Perplexed. Boston, MA: Addison-Wesley. pp. 55–57. ISBN 978-0-321-18612-6.
- George Beekman & Ben Beekman. Digital Planet, Tomorrow's Technology and You 10th Edition. Prentice Hall. 2012
- Kent Beck; James Grenning; Robert C. Martin; Mike Beedle; Jim Highsmith; Steve Mellor; Arie van Bennekum; Andrew Hunt; Ken Schwaber; Alistair Cockburn; Ron Jeffries; Jeff Sutherland; Ward Cunningham; Jon Kern; Dave Thomas; Martin Fowler; Brian Marick (2001). "Principles behind the Agile Manifesto". Agile Alliance. Archived from the original on 14 June 2010. Retrieved 16 August 2020.
- Roger S. Pressman. Software Engineering: A Practitioner's Approach 7th Edition. Mc Graw Hill Higher Education. 2010.


ANY
QUESTIONS?

Sesi Berakhir
TERIMA KASIH