MODUL MATA KULIAH

REKAYASA PERANGKAT LUNAK 1

KP342 - 3 SKS

TIM PENYUSUN

Noni Juliasari, M.Kom

Bima Cahya Putra, M.Kom

Basuki Hari Prasetyo, M.Kom

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS BUDI LUHUR

JAKARTA

VERSI 1.0

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

PERTEMUAN 3
KONSEP RPL

Capaian Pembelajaran	:	Memahami sejarah dan konsep dasar rekayasa perangkat lunak
Sub Pokok Bahasan	:	1.1. Permasalahan seputar pengembangan perangkat lunak 1.2. Urgensi dan konsep rekayasa perangkat lunak 1.3. 7 Aktifitas dasar rekayasa perangkat lunak : • Requirements • Specification • Design • Code • Verification & Validation • Debug • Maintenance
Daftar Pustaka	:	1. Kung, David C., 2014. Object Oriented Software Engineering: An Agile Methodology, McGraw-Hill 2. Pressman, Roger S. 2010. *Software Engineering* : A Practitioner's Approach,7th, McGraw-Hill 3. Schach, Stephen R. 2010. Object Oriented and Classical *Software Engineering*, 8th, McGraw-Hill 4. Sommerville, Ian. 2010. *Software Engineering*, 9th, Pearson Education

		5. *Software Engineering* Body of Knowledge (SWEBOK). 2004 6. Andri Kristianto, Rekayasa Perangkat Lunak (Konsep Dasar), Gava Media Yogyakarta, 2004 7. https://www.iso.org/committee/45086/x/catalogue/ Maiocchi, M. (1991). *Software Engineering*. Future Generation Computer Systems, 7, 23-29.

SEJARAH DAN KONSEP RPL

1.1. Permasalahan seputar pengembangan Perangkat Lunak

Dalam proses pengembangan perangkat lunak yang ada, sebagian besarnya dinyatakan mengalami kegagalan. Beberapa permasalahan yang menjadi faktor penyebab kegagalan proyek pengembangan perangkat lunak antara lain :

1) Produk akhir tidak mampu memenuhi kebutuhan pelanggan secara penuh

   Tujuan utama dalam pengembangan perangkat lunak adalah menjadikan produk yang akan menyelesaikan permasalahan serta memenuhi kebutuhan pelanggan. Namun seringkali kegagalan pengembangan perangkat lunak terjadi karena kedangkalan dalam menganalisa masalah yang akan dituntaskan serta kebutuhan yang diinginkan. Sehingga muncul kekecewaan dari pelanggan saat produk akhir perangkat lunak mereka terima, fungsionalitas yang ada tidak sesuai dengan harapan mereka.

2) Sulit untuk dikembangkan atau diimprovisasi

   Keberhasilan dari produk perangkat lunak tidak hanya dilihat dari program yang bisa "jalan" saja, namun juga dari bagaimana produk tersebut memiliki kemampuan untuk dapat dikembangkan atau diubah di kemudian hari sesuai tuntutan perubahan kebutuhan. Fleksibilitas kemampuan perangkat lunak untuk dimodifikasi ini sangatlah ditunjang dengan rancangan yang baik pula.

3) Dokumentasi yang tidak lengkap

   Keberadaan dokumentasi dalam pengembangan perangkat lunak sangat penting. Dokumentasi akan menjadi catatan rekam jejak yang memudahkan pengembangan perangkat lunak di kemudian hari, entah karena perubahan kebutuhan pengguna atau perkembangan teknologi. Tanpa dokumentasi yang baik, penerus pengembangan sistem tersebut akan mengalami kesulitan untuk menelusuri *history* kebutuhan pengguna yang telah lalu. Dokumentasi juga dapat menjadi alat pertanggungjawaban bagi pengembang untuk menghadapi tuntutan fungsionalitas produk dari pengguna yang tidak sesuai dengan apa yang telah disepakati di awal.

4) Kualitas yang buruk

   Keterbatasan waktu pengembangan serta resiko yang mungkin dihadapi selama

pengembangan perangkat lunak dapat mengakibatkan produk akhir yang di deliver kepada pengguna masih memiliki kekurangan/cacat. Selain itu, analisa masalah serta penelusuran kebutuhan yang tidak sesuai dengan harapan pengguna juga dapat mengakibatkan ketidaklengkapan atau ketidakbenaran fungsionalitas, sehingga kualitas dari produk akhirnya dinilai buruk oleh pelanggan.

5) Melebihi alokasi waktu dan biaya yang direncanakan
   Estimasi kebutuhan sumber daya waktu dan biaya yang dialokasikan untuk pengembangan seringkali tidak terkelola dengan baik sehingga penyelesaian produk seringkali terlambat dan juga melebihi anggaran yang sudah ditetapkan.

Beberapa permasalahan yang seringkali menjadi faktor kegagalan pengembangan perangkat lunak kemudian dikenal dengan *software crisis*. Tanda-tanda yang paling terlihat dari *software crisis* ini adalah keterlambatan pengiriman perangkat lunak ke pelanggan, over budget/melebihi anggaran, perangkat lunak tidak memenuhi persyaratan yang ditentukan/tidak sesuai kebutuhan pelanggan hingga sampai membahayakan, dan dokumentasi yang tidak memadai.

Beberapa penyebab terjadinya *Software Crisis* diantaranya adalah komputer/*hardware* yang semakin canggih, kebutuhan perangkat lunak yang semakin besar dan kompleks, sehingga cara- cara konvensional tidak cukup efektif (secara biaya, waktu, dan kualitas). Oleh karena itu, diperlukan ilmu Rekayasa Perangkat Lunak (*Software Engineering*) yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi, desain, konstruksi, testing sampai pemeliharaan setelah digunakan.

1.2. Urgensi dan Konsep Rekayasa Perangkat Lunak

Dari semua permasalahan yang ditemukan saat pengembangan perangkat lunak tersebut, dapat disimpulkan bahwa membangun produk perangkat lunak tidak hanya cukup mengandalkan kemampuan programming tapi dibutuhkan konsep dan teknik rekayasa perangkat lunak.

Tujuan dari rekayasa perangkat lunak dapat diuraikan sebagai berikut :

1) Menghasilkan sebuah perangkat lunak yang berkualitas.

   Yang dimaksud dengan berkualitas dapat dilihat dari tiga sisi, sisi sponsor (individu atau organisasi yang telah mengeluarkan biaya dalam pembangunan perangkat lunak), sisi pemakai (siapapun yang menggunakan perangkat lunak tersebut), sisi *maintainer / modifier* (yang memelihara dan memodifikasi perangkat lunak tersebut).

   - Sisi Sponsor :

     Tujuan utama sponsor adalah menghasilkan dan atau menghemat uang. Sponsor ingin menggunakan perangkat lunak tersebut untuk meningkatkan produktivitas organisasi. Sponsor mengharapkan untuk dapat menghasilkan sebuah layanan dengan biaya yang rendah tetapi masuk akal. Karena itu sistem yang dibuat harus handal, fleksibel dan efisien. Selain itu biaya dari pemeliharaan, modifikasi dan peningkatan dari sistem tersebut harus serendah mungkin.
   - Sisi Pemakai :

     Bagi pemakai perangkat lunak adalah alat untuk membantu menyelesaikan tugas-tugasnya. Karena itu perangkat lunak harus menyediakan fungsi-fungsi yang dibutuhkan oleh pemakai. Perangkat lunak juga harus handal dan efisien, perangkat lunak harus dapat menghasilkan output yang konsisten. Selain itu pemakai harus merasa perangkat lunak yang dibuat mudah untuk dipelajari, mudah digunakan dan mudah untuk diingat.
   - Sisi *Maintainer/modifier* :

     Yang diinginkan oleh *maintainer/modifier* adalah perangkat lunak tersebut memiliki sangat sedikit error pada saat penginstallan pertama (catatan : sangat kecil kemungkinannya untuk menghasilkan perangkat lunak yang 100 % bebas dari bug). Selain itu perangkat lunak tersebut harus terdokumentasi dengan baik. *Source code* juga harus mudah dibaca, terstruktur dan dirancang dengan baik dan bersifat modular.

2) Menghasilkan perangkat lunak dengan biaya yang efisien dan mampu di *deliver* ke pengguna tepat waktu.

Dengan mempelajari prinsip rekayasa dalam pengembangan perangkat lunak, diharapkan pengembang mampu mengelola sumber daya waktu dan biaya dengan baik sehingga realisasi pengembangan sesuai dengan estimasi yang sudah ditetapkan.

3) Menghasilkan perangkat lunak yang benar dan tepat serta bermanfaat bagi pengguna

   Dengan mempelajari rekayasa perangkat lunak ini diharapkan mampu mengembangkan sebuah perangkat lunak yang berguna dan juga bermanfaat bagi usernya. Sebuah perangkat lunak tentu saja tidak akan digunakan oleh user apabila tidak memiliki fungsi yang spesifik. Karena itu dengan mempelajari rekayasa perangkat lunak, akan membuat seseorang menjadi lebih paham mengenai pengembangan perangkat lunak yang fungsional.

4) Agar mampu melakukan perawatan terhadap produk perangkat lunak serta mampu menjadikan perangkat lunak yang sudah ada menjadi lebih baik lagi

   Fungsi dari mereka yang mempelajari rekayasa perangkat lunak tidak hanya terpaku pada pembuatan dan juga pengembangan dari sistem perangkat lunak yang ada, namun juga berada pada level *maintenance* atau perawatan dari sebuah perangkat lunak yang ada. Setiap perangkat lunak tentu saja membutuhkan *maintenance*, terutam aketika perangkat lunak tersebut mengalami suatu gangguan atau kendala. Ada kalanya juga perangkat lunak yang sudah ada terkadang membutuhkan pembaruan, karena fungsinya yang mungkin sudah berkurang. Karena itu, dengan mempelajari rekayasa perangkat lunak seseorang akan mampu mengembangkan perangkat lunak yang sudah ada sebelumnya agar kemudian menjadi sebuah sistem perangkat lunak yang dapat berguna dan menjadi lebih baik lagi di kalangan *user*.

Marco Maiocchi dari University of Milan dalam jurnal Future Generation Computer Systems volume 7, menjelaskan bahwa setidaknya terdapat beberapa alasan mengapa *software engineering* itu sangat penting, diantaranya adalah:

1) Mengurangi Kompleksitas

   Software dengan skala besar dan kompleks sangat sulit untuk dikembangkan. Untuk mengatasi hal ini, *Software Engineering* dapat diterapkan sebagai sebuah

solusi untuk mengurangi kompleksitas dari setiap projek dengan cara membagi sekumpulan masalah besar menjadi beberapa masalah yang lebih kecil. Dengan demikian, masalah-masalah dapat diselesaikan dengan solusi-solusi yang dikerjakan satu persatu. Dengan teknik ini setiap permasalahan kecil dapat diselesaikan secara independen satu sama lain. Pada akhirnya, setiap masalah yang telah berhasil diselesaikan akan dikombinasikan satu sama lain untuk menghasilkan sebuah solusi akhir. Teknik ini disebut Problem Decomposition. Fokus dari teknik ini adalah memfokuskan penyelesaian masalah pada masalah yang paling relevan untuk diselesaikan saat ini dengan cara mengabaikan masalah-masalah yang dianggap tidak relevan. Dengan pola pikir ini, diharapkan masalah-masalah besar dapat diselesaikan dengan lebih mudah.

2) Untuk meminimalisir biaya perangkat lunak

   Perangkat lunak membutuhkan banyak sekali kerja keras dan disisi lain seorang *software engineer* merupakan tenaga ahli dengan biaya yang mahal. Kebanyakan orang berfokus untuk membangun perangkat lunak dengan jutaan baris kode program. Namun di dalam *software engineering*, programmer harus merencanakan semuanya dan mereduksi segala hal yang dianggap tidak penting. Hasilnya, biaya produksi dari pengembangan perangkat lunak menjadi dapat dikurangi dibandingkan pengembangan perangkat lunak lainnya yang tidak menggunakan pendekatan *software engineering*.

3) Untuk mengurangi waktu

   Segala yang tidak dibuat berdasarkan "Perencanaan" selalu menghabiskan waktu lebih banyak. Dan jika kita membuat sebuah *software* yang memiliki lingkup besar maka anda harus menjalankan banyak kode program untuk membuat kode dapat berjalan dengan maksimal. Hal ini sangat menghabiskan waktu. Jika tidak ditangani dengan manajemen yang baik maka hal ini malah akan menghabiskan banyak waktu.

4) Mampu Menangani Proyek Besar

   Projek besar tidak dibuat hanya dalam beberapa hari, dibutuhkan kesabaran, perencanaan dan manajemen untuk menjamin tercapainya tujuan. Andaikata terdapat sebuah projek yang harus diselesaikan dalam jangka waktu 5 tahun, dan terdapat 40 tugas yang harus dikerjakan dan dalam satu semester perusahaan

harus menyelesaikan setidaknya 4 tugas. Maka perusahaan juga harus memiliki komitmen agar jangan sampai dalam satu semester ternyata tugas yang hanya bisa diselesaikan 1 dari 4 tugas yang harusnya selesai. Oleh sebab itu untuk dapat sesuai dengan waktu yang ditentukan dibutuhkan perencanaan, arahan, pengujian dan proses maintenance yang dijalankan dengan disiplin.

5) Menjamin Kehandalan Perangkat Lunak

Software harus handal, handal berarti *software* yang dibangun haruslah bekerja sesuai dengan waktu yang telah ditetapkan. Dan jika terdapat kesalahan dalam pembuatan *software* maka perusahaan harus menyelesaikan permasalahan-permasalahan tersebut. Di dalam *software engineering*, terdapat teknik testing dan maintenance yang harus dilakukan untuk menjamin kehandalan perangkat lunak.

6) Menjamin Efektifitas

Efektifitas akan terjadi jika perangkat lunak telah berhasil dibangun berdasarkan sebuah standar yang berlaku. Di dalam *software engineering* terdapat berbagai standar yang dapat dijadikan acuan dalam pembangunan sebuah *software* yang efektif seperti ISO 3535:1977 tentang desain formulir dan bagan, ISO 5806/1984 tentang tabel dan information processing, ISO 5807:1985 tentang dokumentasi dan konvensi data, dan lain-lain.

7) Produktivitas

Dengan bantuan *software engineering*, setiap perusahaan dapat meningkatkan produktivitas. Setiap projek dapat dikurangi biayanya dan menghabiskan waktu pengerjaan yang lebih sedikit. Di dalam *software engineering*, produktivitas *software* dapat dihasilkan karena adanya sistem pengujian di dalam setiap bagian terkecil suatu proses. Jika tidak lolos uji, maka seorang *developer* harus memperbaikinya hingga akhirnya mencapai suatu standar yang berlaku.

1.3. Aktifitas Dasar Pada Rekayasa Perangkat Lunak

SDLC (*Systems Development Life Cycle*, Siklus Hidup Pengembangan Sistem) atau *Systems Life Cycle* (Siklus Hidup Sistem), dalam rekayasa sistem dan rekayasa perangkat lunak, adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. Konsep

ini umumnya merujuk pada sistem komputer atau informasi. SDLC juga merupakan pola yang diambil untuk mengembangkan sistem perangkat lunak. Dalam rekayasa perangkat lunak, konsep SDLC mendasari berbagai jenis metodologi pengembangan perangkat lunak. Metodologi-metodologi ini membentuk suatu kerangka kerja untuk perencanaan dan pengendalian pembuatan sistem informasi, yaitu proses pengembangan perangkat lunak.

Aktifitas dasar dalam SDLC antara lain sebagai berikut :

1) Requirement

   Tahapan dimana kebutuhan *customer* (*user*) dikumpulkan, ditangkap secara menyeluruh dan global untuk menjadi masukan buat rekayasa kebutuhan.

   - Mendeteksi tujuan (visi) dari pembuatan produk perangkat lunak : untuk apa sistem dibuat dan tujuan pembuatan sistem.
   - Mendata kebutuhan mengenai kemampuan (unjuk kerja) sistem yang diinginkan pengguna (*customer*)

2) Specification

   Pada tahapan ini kebutuhan yang sudah dikumpulkan pada tahapan *requirement* kemudian dianalisis dan dispesifikasikan agar diperoleh kebutuhan yang jelas, detil dan lengkap.

3) Design

   Tahap ini melakukan rancangan design sistem. Tahap ini memberikan rincian kinerja program dan interaksi antara user dengan program tersebut.

4) Code

   Tahap ini adalah spesifikasi design yang telah dibuat untuk diterjemahkan ke dalam program / instruksi yang ditulis dalam bahasa pemrograman.

5) Test

   Tahap ini semua program digabungkan dan diuji sebagai satu sistem yang lengkap untuk menjamin sumua berkerja dan memenuhi kebutuhan penanganan masalah yang dihadapi. Pada tahapan ini dilakukan verifikasi dan validasi terhadap requirement dan specification yang sudah ditangkap

6) Debug

   Apabila pada tahapan testing ditemukan beberapa kesalahan, maka pada tahapan

ini dilakukan perbaikan sebelum produk diujicoba di lingkungan pengguna

7) Maintenance

Tahapan akhir dari pengembangan perangkat lunak adalah kegiatan perawatan atau pemeliharaan system.

Rangkuman

Rekayasa perangkat lunak telah berkembang sejak pertama kali diciptakan pada tahun 1940-an hingga kini. Focus utama pengembangannya adalah untuk mengembangkan praktek dan teknologi untuk meningkatkan produktivitas para praktisi pengembang perangkat lunak dan kualitas aplikasi yang dapat digunakan oleh pemakai.

Latihan

Pilihlah jawaban yang menurut Anda benar dari pilihan jawaban yang diberikan

1. Apa yang dimaksud dengan rekayasa perangkat lunak/software engineering?
   a. Pengembangan perangkat lunak secara tim(kelompok)
   b. Pengembangan perangkat lunak secara perorangan
   c. Pengambangan perangkat lunak dengan menggunakan CASE
   d. Pengembangan perangkat lunak secara cepat
2. Apa yang dimaksud dengan rekayasa perangkat lunak/software engineering?
   a. Pengembangan PL dengan memanfaatkan metode daur hidup pengembangan sistem
   b. Pengembangan PL dengan memanfaatkan prinspi-prinsip rekayasa dalam pengembangan sistem
   c. Pengembangan PL dengan memanfaatkan CASE
   d. Pengembangan PL secara cepat
3. Apa yang dimaksud dengan rekayasa perangkat lunak/software engineering?
   a. Pengembangan PL dengan memanfaatkan prinsip-prinsip rekayasa
   b. Pengembangan PL dengan memanfaatkan prinsip-prinsip PL
   c. Pengembangan PL dengan memanfaatkan prinsip-prinsip prototipe
   d. Pengembangan sistem dengan memanfaatkan prinsip-prinsip prototipe
4. Apa tujuan rekayasa perangkat lunak?
   a. Menghasilkan PL yang baik walaupun mahal

b. Menghasilkan PL yang efektif
c. Menghasilkan PL yang baik dan murah
d. Menghasilkan PL yang efisien

5. Diantara berikut ini, mana yang tidak termasuk dalam ruang lingkup atau tujuan rekayasa perangkat lunak?
   a. Menghasilkan PL yang bebas dari kesalahan dan sesuai kebutuhan user
   b. Menghasilkan PL yang dapat dibuat tepat waktu
   c. Menghasilkan PL on budget (tepat dengan anggaran biaya yang sudah direncanakan)
   d. Menghasilkan PL yang kompleks dan memiliki fitur-fitur menarik
6. Pernyataan yang benar pada tahapan dasar pada Debug adalah…
   a. Mencari, menemukan dan memperbaiki kesalahan yang ada di dalam program
   b. Mencari dan menemukan kesalahan yang ada di dalam program
   c. Mencari dan memperbaiki kesalahan yang ada di dalam program
   d. Mencari, menemukan, menyelidiki dan memperbaiki kesalah yang ada di dalam program
7. Pada tahapan apa kita dapat membuat program?
   a. Test
   b. Code
   c. Design
   d. Debug
8. Validasi adalah pengujian…
   a. Statis dari sebuah program
   b. Yang dilakukan manusia
   c. Dengan mengeksekusi / menjalankan program
   d. Yang melibatkan programmer, user dan tester
9. Pilihlah jawaban yang bukan termasuk user dan dokumentasi sistem…
   a. Install
   b. Design
   c. Debug
   d. Maintain

10. Pada requirements testing yang diuji adalah…
    a. Kebutuhan system
    b. Output system
    c. Proses Input system
    d. Fungsionalitas system

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS BUDI LUHUR

Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan

Jakarta Selatan, 12260

Telp: 021-5853753 Fax : 021-5853752

http://fti.budiluhur.ac.id