PRAKTIKUM 01: PENGANTAR REKAYASA PERANGKAT LUNAK DAN METHODOLOGY PENGEMBANGAN SISTEM

by Nurul Firdaus, S.Kom., M.Inf.

TUJUAN PEMBELAJARAN

1. Mahasiswa memahami konsep rekayasa perangkat lunak
2. Mahasiswa dapat memahami definisi dan konsep dasar dari ketiga methodology pengembangan sistem berikut ini :
   a. *System Development Life Cycle (SDLC)* atau siklus hidup pengembangan sistem
   b. *Agile approach* atau pendekatan agile
   c. *Object-oriented systems analysis and design* atau analisis dan desain sistem berorientasi objek

REFERENSI

Roger S. Pressman., 2000. Software engineering: a practitioner's approach. *5th ed* : *McGraw-Hill series in computer science*

Kendall, K.E. and Kendall, J.E., 2014. Systems Analysis and Design, Global Edition 9e.

MATERI

REKAYASA PERANGKAT LUNAK (*SOFTWARE ENGINEERING*)

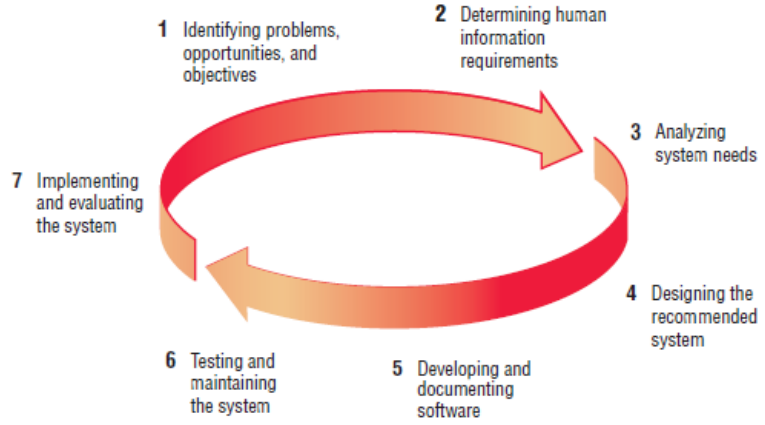
A. DEFINISI REKAYASA PERANGKAT LUNAK

Rekayasa perangkat lunak adalah instruksi (program komputer) yang bila dieksekusi dapat menjalankan fungsi tertentu, struktur data yang dapat membuat program memanipulasi informasi serta dokumen yang menjelaskan operasi dan penggunaan program.

METHODOLOGY PENGEMBANGAN SISTEM

B. *SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)*

System Development Life Cycle (SDLC) adalah pendekatan yang dilakukan secara bertahap dalam hal melakukan analisa dan membangun rancangan sistem dengan menggunakan siklus-siklus secara spesifik terhadap kegiatan penggunanya.



The seven phases of the systems development life cycle (SDLC).

Fase-fase di dalam *System Development Life Cycle* (SDLC) :

1. Mengidentifikasi masalah, peluang, dan tujuan

   Dalam fase ini penganalisis menentukan dengan tepat masalah-masalah dalam bisnis mereka, mengukur peluang guna mencapai sisi kompetitif atau menyusun standar-standar industri, dan tujuan-tujuan yang harus dicapai.

2. Menentukan kebutuhan informasi

   Dalam fase ini, penganalisis berusaha untuk memahami informasi apa yang dibutuhkan pemakai agar bisa digunakan dalam pekerjaan mereka. Orang-orang yang terlibat adalah penganalisis dan pemakai, manajer operasi dan pegawai operasional. Penganalisis sistem perlu mengetahui fungsi-fungsi sistem yang ada secara detail yaitu: siapa, apa, dimana, kapan dan bagaimana dari bisnis yang sedang dipelajari.

3. Menganalisa kebutuhan sistem

   Dalam fase ini, penganalisis menganalisis keputusan terstruktur yang dibuat. Penganalisis juga menyiapkan suatu proposal sistem yang berisikan ringkasan apa saja yang ditemukan, analisis biaya keuntungan alternatif yang tersedia serta rekomendasi atas apa saja yang harus dilakukan.

4. Merancang sistem yang direkomendasikan

   Dalam fase ini, penganalisis merancang *data-entry* sedemikian rupa sehingga data yang dimasukkan ke dalam sistem informasi benar-benar akurat. Penganalisis juga merancang *database* yang menyimpan data-data yang diperlukan oleh pembuat keputusan. Penganalisis bekerja sama dengan pemakai

untuk merancang *output*. Terakhir penganalisis juga merancang prosedur-prosedur *back-up* dan kontrol untuk melindungi sistem dan data serta membuat paket-paket spesifikasi program bagi *programmer*.

5. Mengembangkan dan mendokumentasikan perangkat lunak atau *software*

   Dalam fase ini, penganalisis bekerja sama dengan *programmer* mengembangkan suatu perangkat lunak awal yang diperlukan. Penganalisis juga bekerja sama dengan pemakai untuk mengembangkan dokumentasi perangkat lunak yang efektif, mencakup manual prosedur (*procedure manuals*), bantuan *online* dan *website* yang menampilkan *FAQs* atau file *Read me* yang termuat dalam software baru.

6. Menguji dan me-*maintenance* sistem

   Dalam fase ini, sistem yang telah dibuat harus dilakukan pengujian terlebih dahulu. Sebagian pengujian dilakukan oleh *programmer* sendiri dan lainnya dilakukan oleh penganalisis sistem.

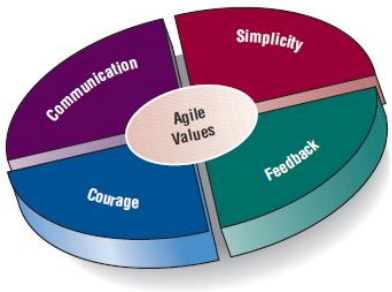
7. Mengimplementasikan dan mengevaluasi sistem

   Fase ini merupakan tahap terakhir yang melibatkan pelatihan bagi pemakai untuk mengendalikan sistem. Pelatihan dilakukan oleh *vendor*, namun kesalahan pelatihan merupakan tanggung jawab penganalisis sistem. Proses ini mencakup pengubahan file-file dari format lama ke format baru atau membangun suatu basis data, menginstall peralatan, dan membawa sistem baru untuk diproduksi.

C. *AGILE APPROACH*

Agile approach adalah pendekatan pengembangan *software* yang berdasarkan pada nilai-nilai, prinsip, dan praktik mendalam. Nilai-nilainya adalah komunikasi, kesederhanaan, umpan balik, dan keberanian.

Values are crucial to the agile approach.

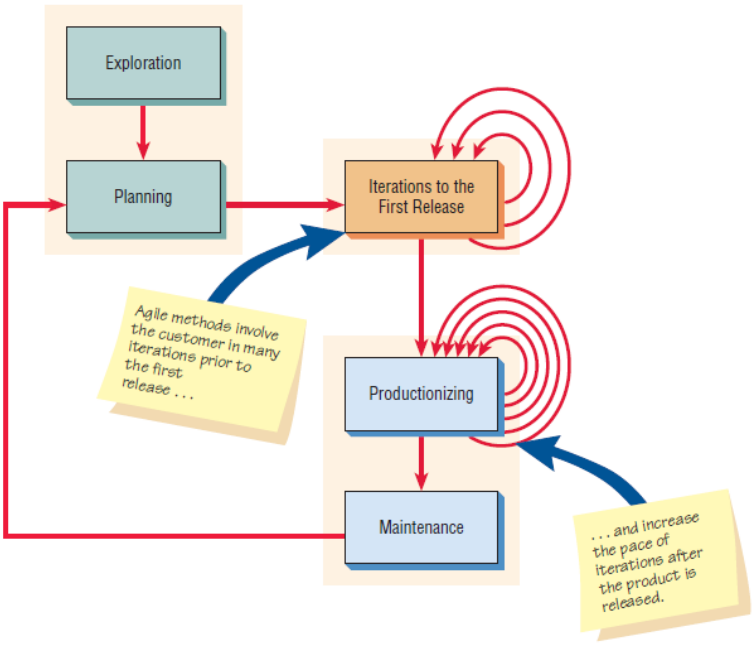


1. Komunikasi (*Communication*)

Tugas utama developer dalam membangun suatu sistem perangkat lunak adalah mengkomunikasikan kebutuhan sistem kepada pengembang perangkat lunak. Komunikasi dalam *agile modelling* dibangun dengan melakukan pemrograman berpasangan (*pair programming*). Developer didampingi oleh pihak klien dalam melakukan *coding* dan *unit testing* sehingga klien bisa terlibat langsung dalam pemrograman sambil berkomunikasi dengan developer. Tujuannya untuk memberikan pandangan pengembang sesuai dengan pandangan pengguna sistem.

2. Kesederhanaan (*Simplicity*)
   Agile modelling mencoba untuk mencari solusi paling sederhana dan praktis. Perbedaan metode ini dengan metodologi pengembangan sistem konvensional lainnya terletak pada proses desain dan *coding* yang terfokus pada kebutuhan saat ini daripada kebutuhan besok, seminggu lagi atau sebulan lagi. Lebih baik melakukan hal yang sederhana dan mengembangkannya besok jika diperlukan.
3. Umpan Balik (*Feedback*)
   Hal ini diperlukan untuk mengetahui kemajuan dari proses dan kualitas dari aplikasi yang dibangun. Informasi ini harus dikumpulkan setiap interval waktu yang singkat secara konsisten. Ini dimaksudkan agar hal-hal yang menjadi masalah dalam proses pengembangan dapat diketahui sedini mungkin. Setiap *feedback* ditanggapi dengan melakukan *test*, *unit test* atau *system integration* dan jangan menunda karena biaya akan membengkak (uang, tenaga, waktu).
4. Keberanian (*Courage*)
   Berani mencoba ide baru. Berani mengerjakan kembali dan setiap kali kesalahan ditemukan, langsung diperbaiki. Contoh dari *courage* adalah komitmen untuk selalu melakukan design dan *coding* untuk saat ini dan bukan untuk esok. Ketika ada kode yang terlalu rumit, sulit dibaca dan dipahami, tidak sesuai dengan kemauan pelanggan, dll maka seharusnya kode program seperti itu di *refactor* (kalau perlu dibangun ulang). Hal ini menjadikan pengembang merasa nyaman dengan *refactoring program* ketika diperlukan.

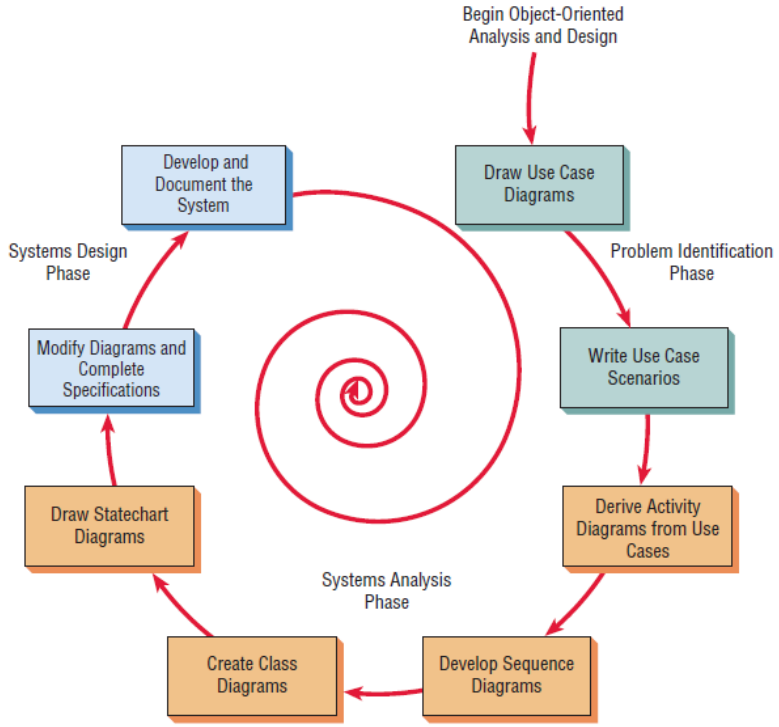
The five stages of the agile modeling development process show that frequent iterations are essential to successful system development.



D. *OBJECT-ORIENTED (O-O) SYSTEMS ANALYSIS AND DESIGN*

Analisis dan desain sistem berorientasi objek merupakan pendekatan yang dimaksudkan untuk memfasilitasi pengembangan sistem yang harus berubah dengan cepat sebagai respons terhadap lingkungan bisnis yang dinamis. Metode ini bekerja dengan baik dalam situasi dimana sistem informasi yang rumit sedang menjalani pemeliharaan, adaptasi, dan desain ulang yang berkelanjutan. Pendekatan berorientasi objek menggunakan standar industri untuk pemodelan *object-oriented (o-o) systems*, yang disebut *Unified Modeling Language* (UML), untuk memecah suatu sistem menjadi model use case.

The steps in the UML development process.



TUGAS DAN RESPONSI PRAKTIKUM 01

Carilah contoh konkret pengembangan sistem menggunakan metodology SDLC yang sesuai dengan fase-fase diatas. Jelaskan secara ringkas! (Cantumkan referensi/sumber dibaris paling akhir)

Simpan file dengan format **.pdf** dengan nama file **kelas_nim_nama_prak01-1_prpl**