SIKLUS HIDUP PERANGKAT LUNAK

SWDLC (SOFTWARE DEVELOPMENT LIFE CYCLE)

Di Susun Oleh :
ABDUL KARIM, M.TI

FAKULTAS TEKNIK
PROGRAM STUDY TEKNIK INFORMATIKA
UNIVERSITAS ALWASHLIYAH
LABUHANBATU 2016

A. SIKLUS HIDUP PERANGKAT LUNAK

Jika ditinjau dari sisi definisi, siklus hidup memiliki beberapa definisi sebagai berikut:

1) Dari Gustafson :

"The Software life cycle is the sequence of different activities that take place during software development".

Definisi ini menyatakan bahwa siklus hidup adalah urutan dari kegiatan yang ada di dalam sebuah pengembangan perangkat lunak. Dalam penjelasnnya bahwa urutan tersebut tidaklah harus benar-benar urut, tetapi dapat mengikuti dengan jenis siklus hidup yang dianut oleh pengembang perangkat lunak itu sendiri.

2) Dari Keyes :

"A system has life of its own. It starts out as an idea and progresses until this idea germinates and then is born".

Dalam bukunya, Jennifer keyes lebih menekankan bahwa sebuah perangkat lunak bias saja mengalami sebuah siklus hidup bergantung dari proses pengembangannya mulai dari ide dasar hingga saat lahirnya perangkat lunak itu sendiri.

3) Dari IEEE :

"T***he life cycle of a software system is normally defined as the period of time that starts when a software product is conceived and ends when the product is no longer available for use".***

Dari standard IEEE 1016, ditekankan bahwa siklus hidup adalah segala sesuatu yang lebih berdasar kepada urutan waktu dibandingkan proses yang terjadi. Sedangkan dari referensi IEEE lain disebutkan bahwa yang dimaksud dengan siklus hidup dalam konteks perangkat lunak life cycle lebih fokus kepada siklus hidup suatu data dalam perangkat lunak.

Dari ketiga definisi tersebut, sekilas terlihat bahwa yang dimaksud dengan siklus hidup dalam konteks RPL berarti tidak sama dengan definisi dari proses perangkat lunak. Secara umum dapat disimpulkan bahwa siklus hidup perangkat lunak adalah urutan hidup sebuah perangkat lunak berdasarkan perkembangan perangkat lunak yang ditentukan oleh pengembang perangkat lunak itu sendiri. Sehingga dapat ditentukan usia fungsional dari sebuah perangkat lunak, apakah akan menjadi using dan mati, ataukah lahir kembali dalam bentuk berbeda menggunakan model proses tertentu.

Bagi pihak pengembang, tentu saja hal tersebut akan menguntungkan secara financial, tetapi secara etika seharusnya hal tersebu tidak terjadi. Jika pemahaman mengenai konsep dasar RPL dapat dijelaskan ke pelanggan atau pengguna, maka semua unsure pelaku RPL akan mampu mengatasi masalah dengan solusi yang lebih efisien.

System Life Cycle (SLC) adalah proses evolusi yang diikuti oleh pelaksanaan system informasi dasar-dasar atau subsistem. Telah ada pendekatan implementasi tradisional sepanjang era komputer, dan ada perjanjian umum antara ahli-ahli komputer sehubungan dengan tugas-tugas yang dilaksanakan.

Berbagai metodologi SLC telah dikembangkan untuk memandu proses yang terlibat termasuk model air terjun (asli metode SLC), pengembangan aplikasi cepat (RAD), pengembangan aplikasi bersama (JAD), maka air mancur model dan spiral model. Umumnya, beberapa model digabungkan ke dalam beberapa jenis hibrida metodologi. Dokumentasi sangat penting berapapun jenis model dipilih atau dibuat untuk setiap aplikasi, dan biasanya dilakukan bersamaan dengan proses pembangunan. Beberapa metode kerja lebih spesifik untuk jenis proyek, tetapi dalam analisis terakhir, faktor yang paling penting bagi keberhasilan suatu proyek dapat seberapa dekat rencana tertentu diikuti.

Beberapa SLC terdapat dalam perusahaan yang menggunakan komputer, mungkin ada seratus atau lebih. Pada kenyataannya SLC adalah sarana yang digunakan oleh manajemen untuk melaksanakan rencana strategis. Konsep life cycle menjadikan segala sesuatu yang tumbuh, menjadi dewasa setiap waktu dan akhirnya mati. Pola ini digunakan untuk sistem dasar komputer seperti subsistem pemrosesan data atau SSD.

Dalam pengembangan software ada beberapa tahapan utnuk mencapai kualitas pembuatan/ siklus hidup perangkat lunak. Siklus hidup perangkat lunak atau tahap pengembangan software adalah sebagai berikut :

a. Requirements Analysis (Analisa Kebutuhan)

Tahap ini menganalisa masalah dan kebutuhan yang harus diselesaikan dengan sistem komputer yang akan dibuat. Tahap ini berakhir dengan pembuatan laporan kelayakan yang mengidentifikasi kebutuhan sistem yang baru dan merekomendasikan apakah kebutuhan atau masalah tersebut dapat diselesaikan dengan sistem komputer yang ada.

b. System and Software Design (Perancangan Sistem dan Software)

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

c. Implementation (Implementasi)

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

d. System Testing (Pengujian Sistem)

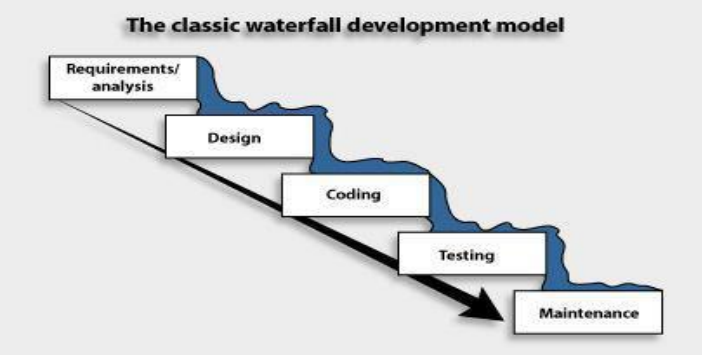
Tahap ini semua program digabungkan dan diuji sebagai satu sistem yang lengkap untuk menjamin sumua berkerja dan memenuhi kebutuhan penanganan masalah yang dihadapi.

e. Operation and Maintenance (Pengoperasian dan Pemeliharaan)

Tahap ini merupakan pengaplikasian program yang telah dibuat untuk digunakan secara utuh dan masalah baru yang muncul sebagai bahan masukan untuk memperbaiki sistem program yang baru.

B. MODEL SIKLUS HIDUP PERANGKAT LUNAK

1) Waterfall Model



Menurut Pressman, (Pressman, 2005, page 79), dalam rekayasa perangkat lunak, terdapat suatu pendekatan yang disebut Waterfall model. Nama model ini sebenarnya adalah “Linear Sequential Model”. Model ini sering disebut dengan “classic life cycle” atau model waterfall. Model ini adalah model yang muncul pertama kali yaitu sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai didalam Software Engineering (SE).

Model ini melakukan pendekatan secara sistematis dan urut mulai dari level kebutuhan sistem lalu menuju ketahap analisis, desain, coding, testing dan maintenance. Model ini merupakan model yang paling banyak dipakai oleh para pengembang software. Ada lima tahap dalam model waterfall, yaitu: Requirement Analysis, System Design, Implementation, Integration & Testing, Operations & Maintenance.

Sesuai dengan namanya waterfall (air terjun) maka tahapan dalam model ini disusun bertingkat, setiap tahap dalam model ini dilakukan berurutan, satu sebelum yang lainnya (lihat tanda anak panah). Selain itu dari satu tahap kita dapat kembali ketahap sebelumnya.

Model ini biasanya digunakan untuk membuat sebuah software dalam skala besar dan yang akan dipakai dalam waktu yang lama.

Tahap – Tahap Dalam Model Waterfall:

1. *Requirement Analysis*

Seluruh kebutuhan software harus bisa didapatkan dalam fase ini, termasuk didalamnya kegunaan software yang diharapkan pengguna dan batasan software. Informasi ini biasanya dapat diperoleh melalui wawancara, survey atau diskusi. Informasi tersebut dianalisis untuk mendapatkan dokumentasi kebutuhan pengguna untuk digunakan pada tahap selanjutnya.

2. *System Design*

Tahap ini dilakukan sebelum melakukan coding. Tahap ini bertujuan untuk memberikan gambaran apa yang seharusnya dikerjakan dan bagaimana tampilannya. Tahap ini membantu dalam menspesifikasikan kebutuhan hardware dan sistem serta mendefinisikan arsitektur sistem secara keseluruhan.

3. *Implementation*

Dalam tahap ini dilakukan pemrograman. Pembuatan software dipecah menjadi modul-modul kecil yang nantinya akan digabungkan dalam tahap berikutnya. Selain itu dalam tahap ini juga dilakukan pemeriksaaan terhadap modul yang dibuat, apakah sudah memenuhi fungsi yang diinginkan atau belum.

4. *Integration & Testing*

Di tahap ini dilakukan penggabungan modul-modul yang sudah dibuat dan dilakukan pengujian ini dilakukan untuk mengetahui apakah software yang dibuat telah sesuai dengan desainnya dan masih terdapat kesalahan atau tidak.

5. *Operation & Maintenance*

Ini merupakan tahap terakhir dalam model waterfall. Software yang sudah jadi dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

Kelebihan Waterfall Model:

Ketika semua kebutuhan sistem dapat didefinisikan secara utuh, eksplisit, dan benar diawal proyek, maka software engineering dapat berjalan dengan baik dan tanpa masalah.

Kekurangan Waterfall Model:

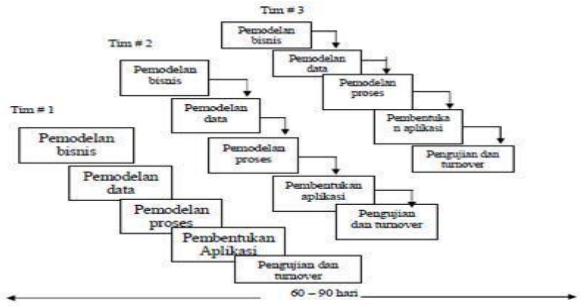
Ketika problem muncul, maka proses berhenti, karena tidak dapat menuju ketahapan selanjutnya. Bahkan jika kemungkinan problem tersebut muncul akibat kesalahan dari

tahapan sebelumnya, maka proses harus membenahi tahapan sebelumnya agar problem ini tidak muncul. Hal-hal seperti ini yang dapat membuang waktu pengerjaan software engineering.

Karena pendekatannya secara sequential, maka setiap tahap harus menunggu hasil dari tahap sebelumnya. Hal itu tentu membuang waktu yang cukup lama, artinya bagian lain tidak dapat mengerjakan hal lain selain hanya menunggu hasil dari tahap sebelumnya. Oleh karena itu, seringkali model ini berlangsung lama pengerjaannya.

Pada setiap tahap proses tentunya dipekerjakan sesuai spesialisasinya masing-masing. Oleh karena itu, ketika tahap tersebut sudah tidak dikerjakan, maka sumber dayanya juga tidak terpakai lagi. Oleh karena itu, seringkali pada model proses ini dibutuhkan seseorang yang "multi-skilled", sehingga minimal dapat membantu pengerjaan untuk tahapan berikutnya.

2) RAD (Rapid Application Development) Model



Rapid Application Development (RAD) atau Rapid Prototyping adalah model proses pembangunan perangkat lunak yang tergolong dalam teknik incremental (bertingkat). RAD menekankan pada siklus pembangunan pendek, singkat, dan cepat. Waktu yang singkat adalah batasan yang penting untuk model ini.

Rapid application development menggunakan metode interatif (berulang) dalam mengembangkan sistem dimana working model (model bekerja) sistem dikonstruksikan di awal tahap pengembangan dengan tujuan menetapkan kebutuhan (requirement) user dan selanjutnya disingkirkan. Working model digunakan kadang-kadang saja sebagai basis desain dan implementasi sistem final.

Tahap – Tahap Rekayasa Software Dalam RAD Model :

1. *Business modeling*

Pada tahap ini, aliran informasi (information flow) pada fungsi-fungsi bisnis dimodelkan untuk mengetahui informasi apa yang mengendalikan proses bisnis, informasi

apa yang hasilkan, siapa yang membuat informasi itu, kemana saja informasi mengalir, dan siapa yang mengolahnya.

2. *Data modeling*

Aliran informasi yang didefinisikan dari business modeling, disaring lagi agar bisa dijadikan bagian-bagian dari objek data yang dibutuhkan untuk mendukung bisnis tersebut. Karakteristik (atribut) setiap objek ditentukan beserta relasi antar objeknya.

3. *Process modelling*

Objek-objek data yang didefinisikan sebelumnya diubah agar bisa menghasilkan aliran informasi untuk diimplementasikan menjadi fungsi bisnis. Pengolahan deskripsi dibuat untuk menambah, merubah, menghapus, atau mengambil kembali objek data.

4. *Application generation*

RAD bekerja dengan menggunakan fourth generation techniques (4GT). Sehingga pada tahap ini sangat jarang digunakan pemrograman konvensional menggunakan bahasa pemrograman generasi ketiga (third generation programming languages), tetapi lebih ditekankan pada reuse komponen-komponen (jika ada) atau membuat komponen baru (jika perlu). Dalam semua kasus, alat bantu untuk otomatisasi digunakan untuk memfasilitasi pembuatan perangkat lunak

5. *Testing and Turnover*

Karena menekankan pada penggunaan kembali komponen yang telah ada (reuse), sebagian komponen-komponen tersebut sudah diuji sebelumnya. Sehingga mengurangi waktu testing secara keseluruhan. Kecuali untuk komponen-komponen baru.

<u>Kelebihan RAD Model :</u>

RAD memang lebih cepat dari Waterfall. Jika kebutuhan dan batasan proyek sudah diketahui dengan baik. Juga jika proyek memungkinkan untuk dimodularisasi.

<u>Kekurangan RAD Model :</u>

Tidak semua proyek bisa dipecah (dimodularisasi), sehingga belum tentu RAD dipakai pada semua proyek. Karena proyek dipecah menjadi beberapa bagian, maka dibutuhkan banyak orang untuk membentuk suatu tim yang mengerjakan tiap bagian tersebut.

Membutuhkan komitmen antara pengemang dengan pelanggan.

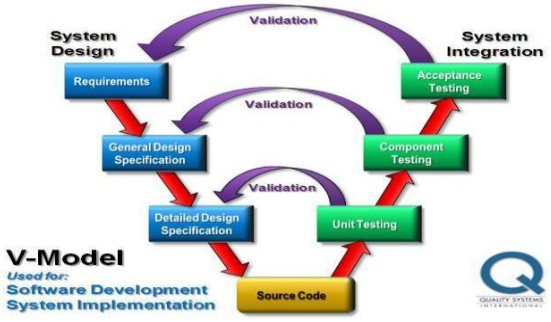
Model RAD memerlukan sumber daya yang cukup besar, terutama untuk proyek dengan skala besar.

Resiko teknis yang tinggi kurang cocok untuk model ini.

Sistem yang tidak bisa dimodularisasi tidak cocok untuk model ini.

Karena dibuat dengan reuse komponen-komponen yang sudah ada, fasilitas-fasilitas pada tiap komponen belum tentu digunakan seluruhnya oleh program yang me-reuse-nya sehingga kualitas program.

3) Model V



Model ini merupakan perluasan dari model waterfall. Disebut sebagai perluasan karena tahap-tahapnya mirip dengan yang terdapat dalam model waterfall. Jika dalam model waterfall proses dijalankan secara linear, maka dalam model V proses dilakukan bercabang. Dalam model V ini digambarkan hubungan antara tahap pengembangan software dengan tahap pengujiannya.

Berikut penjelasan masing-masing tahap beserta tahap pengujiannya:

1. Requirement Analysis & Acceptance Testing

Tahap Requirement Analysis sama seperti yang terdapat dalam model waterfall. Keluaran dari tahap ini adalah dokumentasi kebutuhan pengguna.

Acceptance Testing merupakan tahap yang akan mengkaji apakah dokumentasi yang dihasilkan tersebut dapat diterima oleh para pengguna atau tidak.

2. System Design & System Testing

Dalam tahap ini analis sistem mulai merancang sistem dengan mengacu pada dokumentasi kebutuhan pengguna yang sudah dibuat pada tahap sebelumnya. Keluaran dari tahap ini adalah spesifikasi software yang meliputi organisasi sistem secara umum, struktur data, dan yang lain. Selain itu tahap ini juga menghasilkan contoh tampilan window dan juga dokumentasi teknik yang lain seperti Entity Diagram dan Data Dictionary.

3. Architecture Design & Integration Testing

Sering juga disebut High Level Design. Dasar dari pemilihan arsitektur yang akan digunakan berdasar kepada beberapa hal seperti: pemakaian kembali tiap modul, ketergantungan tabel dalam basis data, hubungan antar interface, detail teknologi yang dipakai.

4. Module Design & Unit Testing

Sering juga disebut sebagai Low Level Design. Perancangan dipecah menjadi modul-modul yang lebih kecil. Setiap modul tersebut diberi penjelasan yang cukup untuk memudahkan programmer melakukan coding. Tahap ini menghasilkan spesifikasi program seperti: fungsi dan logika tiap modul, pesan kesalahan, proses input-output untuk tiap modul, dan lain-lain.

5. Coding

Dalam tahap ini dilakukan pemrograman terhadap setiap modul yang sudah dibentuk. V Model memiliki beberapa kelebihan. Kelebihan-kelebihan tersebut secara garis besar dapat dijelaskan seperti berikut:

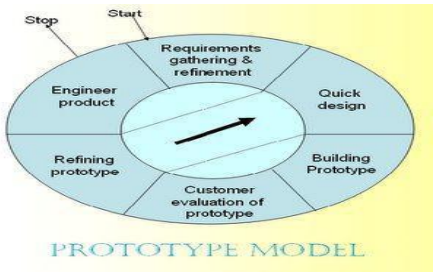
V Model sangat fleksibel. V Model mendukung project tailoring dan penambahan dan pengurangan method dan tool secara dinamik. Akibatnya sangat mudah untuk melakukan tailoring pada V Model agar sesuai dengan suatu proyek tertentu dan sangat mudah untuk menambahkan method dan tool baru atau menghilangkan method dan tool yang dianggap sudah obsolete.

V Model dikembangkan dan di-maintain oleh publik. User dari V Model berpartisipasi dalam change control board yang memproses semua change request terhadap V Model.

<u>Kekurangan V Model yaitu:</u>

- V Model adalah model yang project oriented sehingga hanya bisa digunakan sekali dalam suatu proyek.
- V Model terlalu fleksibel dalam arti ada beberapa activity dalam V Model yang digambarkan terlalu abstrak sehingga tidak bisa diketahui dengan jelas apa yang termasuk dalam activity tersebut dan apa yang tidak.

4) Prototyping Model



Prototyping adalah salah satu metode siklus hidup sistem yang didasarkan pada konsep model bekerja. *Prototyping* adalah bentuk dasar atau model awal dari suatu sistem atau bagian dari suatu sistem. *Prototyping* adalah proses pengembangan model awal tersebut

untuk digunakan terlebih dahulu dan ditingkatkan terus menerus sampai didapatkan sistem yang utuh, artinya sistem akan dikembangkan lebih cepat daripada metode tradisional dan biayanya menjadi lebih rendah.

Tujuan dari *Prototyping* ialah untuk memperkecil resiko rekayasa-ulang proses bisnis. Bila tidak mungkin dibuat prototipe-nya, maka dengan inovasi bertahap, sedemikian rupa sehingga manajemen dapat memimpin melalui serangkaian perubahan yang layak. Prototype dapat memberikan ide bagi pembuat dan pemakai potensial tentang cara sistem berfungsi dalam bentuk lengkapnya.

Tahap – Tahap Rekayasa Software Dalam Prototype Model

1. Pengumpulan kebutuhan

Developer dan klien bertemu untuk menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detail kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan.

2. Perancangan Cepat

Perancangan dilakukan cepat dan rancangan mewakili semua aspek software yang diketahui, dan rancangan ini menjadi dasar pembuatan prototype.

3. Bangun Prototype

Dalam tahap ini, membangun sebuah versi prototype yang dirancang kembali dimana masalah-masalah tersebut diselesaikan.

4. Evaluasi prototype

Pada tahap ini, klien mengevaluasi prototype yang dibuat dan digunakan untuk memperjelas kebutuhan software.

5. Perbaikan Prototype

Tahap ini Software yang sudah jadi dijalankan dilakukan perbaikan. Perbaikan termasuk dalam memperbaiki kesalahan/kerusakan yang tidak ditemukan pada langkah sebelumnya.

Kelebihan Prototype Model adalah :

End user dapat berpartisipasi aktif.

Penentuan kebutuhan lebih mudah diwujudkan.

Mempersingkat waktu pengembangan software.

Kekurangan Prototype Model adalah :

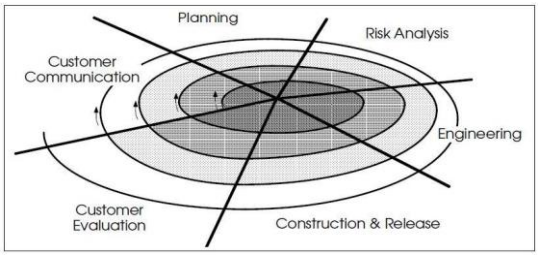
Proses analisis dan perancangan terlalu singkat.

Mengesampingkan alternatif pemecahan masalah.

Bisanya kurang fleksibel dalam menghadapi perubahan.

Prototype yang dihasilkan tidak selamanya mudah dirubah. Prototype terlalu cepat selesai.

5) Spiral Model



Model spiral (spiral model) yang pada awalnya diusulkan oleh Boehm adalah model proses perangkat lunak yang evolusioner yang merangkai sifat iteratif dari prototipe dengan cara kontrol dan aspek sistematis dari model sekuensial linier. Di dalam model spiral, perangkat lunak dikembangkan di dalam suatu deretan pertambahan.

Selama awal iterasi, rilis inkremental bisa merupakan sebuah model atau prototipe kertas. Selama iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap.

Model spiral dibagi menjadi sejumlah aktifitas kerangka kerja, disebut juga wilayah tugas, diantara tiga sampai enam wilayah tugas :

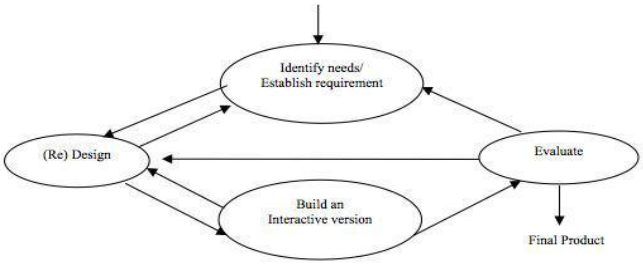
- Komunikasi pelanggan, tugas-tugas yang dibutuhkan untuk membangun komunikasi yang efektif diantara pengembang dan pelanggan.
- Perencanaan, tugas-tugas yang dibutuhkan untuk mendefinisikan sumber-sumber daya, ketepatan waktu, dan proyek informasi lain yang berhubungan.
- Analisis resiko, tugas-tugas yang dibutuhkan untuk menaksir resiko-resiko, baik manajemen maupun teknis.
- Perekayasaan, tugas-tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari aplikasi tersebut.
- Konstruksi dan peluncuran, tugas-tugas yang dibutuhkan untuk mengkonstruksi, menguji, memasang (instal) dan memberikan pelayanan kepada pemakai (contohnya pelatihan dan dokumentasi).
- Evaluasi pelanggan, tugas-tugas yang dibutuhkan untuk memperoleh umpan balik dari pelanggan dengan didasarkan pada evaluasi representasi perangkat lunak, yang dibuat selama masa perekayasaan, dan diimplementasikan selama pemasangan.

Model spiral menjadi sebuah pendekatan yang realistis bagi perkembangan system dan perangkat lunak skala besar. Karena perangkat lunak terus bekerja selama proses

bergerak, pengembang dan pemakai memahami dan bereaksi lebih baik terhadap resiko dari setiap tingkat evolusi. Model spiral menggunakan prototipe sebagai mekanisme pengurangan resiko.

Tetapi yang lebih penting lagi, model spiral memungkinkan pengembang menggunakan pendekatan prototipe pada setiap keadaan di dalam evolusi produk. Model spiral menjaga pendekatan langkah demi langkah secara sistematik seperti yang diusulkan oleh siklus kehidupan klasik, tetapi memasukkannya ke dalam kerangka kerja iterative yang secara realistis merefleksikan dunia nyata.

6) Simple Interaction Desain Model



Pada model rancangan interaksi sederhana ini input atau masukan hanya memiliki satu titik. yang mana masukan tersebut diidentifikasikan apakah sesuai dengan kebutuhan, lalu didesain sesuai dengan persyaratan yang telah ditetapkan. Setelah didesain rancangan tersebut dibangun dan harus interaktif. Setelah itu barulah rancangan tersebut dievaluasi.

Evaluasi dapat dilakukan dimana saja, rancangan yang telah di evakuasi dapat kambali didesain ulang atau apakah rancangan tersebut tidak sesuai dengan kebutuhan user, maka alur tersebut akan terus berputar hingga pada tahap evaluasi tidak lagi terjadi kesalahan, baik dalam penetapan kebutuhan user maupun pendesainannya, sehingga pada tahap evaluasi terciptalah sebuah hasil akhir yang valid.

7) Star Lifecycle Model

Dalam Siklus permodelan ini pengujian dilakukan terus menerus, tidak harus dikahir. Misalnya dimulai dari menentukan konsep desain (conceptual design) dalam proses ini akan

langsung terjadi evaluasi untuk langsung ternilai apakah sudah sesuai dengan kebutuhan user, bila belum maka akan terus berulang di evaluasi hingga benar-benar pas, selanjutnya apabila sudah pas, maka dari tahap evaluasi yang pertama akan lanjut ke proses yg selanjutnya yaitu requirements/specification yakni memverifikasikan persyaratan rancangan tersebut, dan pada tahap itu juga langsung terjadi pengevaluasian seperti tahap pertama, dan selanjutnya akan tetap sama terjadi pada tahapan-tahapan selanjutnya yakni task analysis/fungsion analysis, pengimplementasian, prototyping hingga pada akhirnya terciptalah sebuah aplikasi yang sesuai dengan kebutuhan user.

Intinya pada rancangan model ini pengevaluasian dilakukan disetiap tahapan tidak hanya pada tahapan akhir seperti model-model rancangan yang lainnya.

C. SDLC (**Systems Development Life Cycle**, **Siklus Hidup Pengembangan Sistem**)

Pengertian SDLC

SDLC 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. Terdapat 3 jenis metode siklus hidup sistem yang paling banyak digunakan, yakni: siklus hidup sistem tradisional (traditional system life cycle), siklus hidup menggunakan protoyping (life cycle using prototyping), dan siklus hidup sistem orientasi objek (object-oriented system life cycle). SDLC (*Software Development Life Cycle*) berarti sebuah siklus hidup pemngembangan perangkat lunak yang terdiri dari beberapa tahapan-tahapan yang sangat penting dalam keberadaan perangkat lunak yang dilihat dari segi pengembangannya.

Sejarah SDLC

Siklus Hidup Sistem(SLC) adalah metodologi yang digunakan untuk menggambarkan proses untuk membangun sistem informasi , dimaksudkan untuk mengembangkan sistem informasi dalam cara yang sangat disengaja, terstruktur dan teratur, mengulangi setiap tahap siklus hidup . Pengembangan sistem siklus hidup, menurut Elliott & Strachan & Radford (2004), "berasal pada tahun 1960, untuk mengembangkan skala besar fungsional sistem bisnis di zaman skala besar konglomerat bisnis . Sistem informasi kegiatan berkisar berat pengolahan data dan angka-angka rutinitas ".

Beberapa kerangka kerja pengembangan sistem telah sebagian didasarkan pada SDLC, seperti analisis sistem terstruktur dan metode desain (SSADM) diproduksi untuk pemerintah Inggris Kantor Pemerintah Commerce pada 1980-an. Sejak saat itu, menurut

Elliott (2004), “pendekatan siklus kehidupan tradisional untuk pengembangan sistem telah semakin digantikan dengan alternatif pendekatan dan kerangka kerja, yang berusaha mengatasi beberapa kekurangan yang melekat pada SDLC tradisional”.

SDLC adalah proses yang digunakan oleh analis sistem untuk mengembangkan sistem informasi , termasuk persyaratan, validasi kepemilikan (stakeholder), pelatihan, dan pengguna. Setiap SDLC harus menghasilkan sistem berkualitas tinggi yang memenuhi atau melebihi harapan pelanggan, mencapai selesai dalam waktu dan perkiraan biaya, bekerja secara efektif dan efisien di saat ini dan direncanakan Teknologi Informasi infrastruktur , dan murah untuk mempertahankan dan biaya-efektif untuk meningkatkan.

Model SDLC dapat dijelaskan sepanjang spektrum gesit untuk iteratif untuk berurut. metodologi Agile, seperti XP dan scrum, fokus pada proses ringan yang memungkinkan untuk perubahan yang cepat di sepanjang siklus pengembangan. Iteratif metodologi, seperti kesatuan proses rasional dan dinamis pengembangan sistem metode, fokus pada lingkup proyek terbatas dan memperluas atau memperbaiki produk oleh beberapa iterasi. Sequential atau besar-desain-up-depan (BDUF) model, seperti Air Terjun , fokus pada perencanaan lengkap dan benar untuk membimbing proyek-proyek besar dan risiko untuk hasil yang sukses dan dapat diprediks. Model-model lain, seperti Pembangunan Anamorphic , cenderung fokus pada bentuk pembangunan yang dipandu oleh ruang lingkup proyek dan iterasi pengembangan fitur adaptif.

Dalam manajemen proyek proyek dapat didefinisikan baik dengan siklus hidup proyek (PLC) dan SDLC, selama kegiatan yang sedikit berbeda terjadi. Menurut Taylor (2004) “siklus hidup proyek mencakup semua kegiatan proyek , sedangkan siklus hidup pengembangan sistem berfokus pada produk menyadari persyaratan ”.

Tahapan SDLC

Proses pengembangan sistem melewati beberapa tahapan dari mulai sistem itu direncanakan sampai sistem tersebut diterapkan.

Di dalam System Development Live Cycle (SDLC) terdapat 6 jenis tahapan pekerjaan yang dilakukan oleh analis sistem dan programmer dalam membangun sistem informasi. Langkah yang digunakan meliputi:

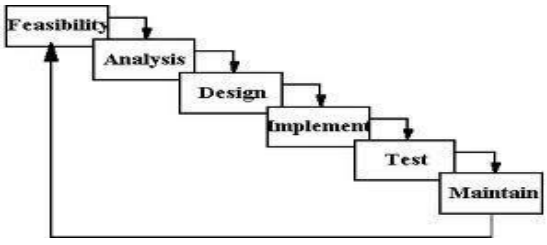
1. Perencanaan sistem, yaitu mempelajari konsep sistem dan permasalahan yang hendak diselesaikan. apakah sistem baru tersebut realistis dalam masalah pembiayaan, waktu, serta perbedaan dengan sistem yang ada sekarang.
2. Analisis system adalah sebuah proses investigasi terhadap sistem yang sedang berjalan dengan tujuan untuk mendapatkan jawaban mengenai pengguna sistem, cara

kerja sistem dan waktu penggunaan sistem. Dari proses analisa ini akan didapatkan cara untuk membangun sistem baru.

3. Desain sistem merupakan proses penentuan cara kerja sistem dalam hal architechture design, interface design, database dan spesifikasi file, dan program design. Hasil dari proses perancangan ini akan didapatkan spesifikasi sistem.
4. Seleksi sistem yaitu Tahap untuk memilih perangkat keras & perangkat lunak yang dibutuhkan.
5. Implementasi sistem adalah proses pembangunan dan pengujian sistem, instalasi sistem, dan rencana dukungan sistem.
6. Pemeliharaan sistem yaitu sistem yang telah diimplemantasikan serta dapat mengikuti perkembangan dan perubahan apapun yang terjadi guna meraih tujuan penggunaannya.

Siklus SDLC dijalankan secara berurutan, mulai dari langkah pertama hingga langkah terakhir. Setiap langkah yang telah selesai harus dikaji ulang, kadang-kadang bersama expert user, terutama dalam langkah spesifikasi kebutuhan dan perancangan sistem untuk memastikan bahwa langkah telah dikerjakan dengan benar dan sesuai harapan. Jika tidak maka langkah tersebut perlu diulangi lagi atau kembali ke langkah sebelumnya.

Gambar SDLC



Kegunaan SDLC

Adapun kegunaan utama dari SDLC adalah mengakomodasi beberapa kebutuhan. Kebutuhan-kebutuhan itu biasanya berasal dari kebutuhan pengguna akhir dan juga pengadaan perbaikan sejumlah masalah yang terkait dengan pengembangan perangkat lunak. Kesemua itu dirangkum pada proses SDLC yang dapat berupa penambahan fitur baru baik itu secara modular (instalasi parsial atau update dan upgrade perangkat lunak) maupun dengan proses instalasi baru (penggantian perangkat lunak menyeluruh atau software replacement).

Dari proses SDLC juga berapa lama umur sebuah perangkat lunak dapat diperkirakan untuk dipergunakan yang dapat diukur atau disesuaikan dengan kebijakan dukungan (software support) dari pengembang perangkat lunak terkait.

D. SWDLC (Software Development Life Cycle)

SWDLC (Software Development Life Cycle, Siklus Hidup Pengembangan Software), merupakan siklus yang harus dilakukan dalam pengembangan Perangkat Lunak (PL), agar sesuai dengan tahap-tahap yang sudah ditentukan dan untuk mendapatkan Perangkat Lunak yang sesuai dengan kebutuhan user. Salah satu hal dasar dalam rekayasa perangkat lunak adalah daur hidup perangkat lunak (software development life cycle), yang mendeksripsikan aktifitas yang terjadi mulai dari pembentukan konsep awal suatu sistem hingga tahap implementasi sistem dan pemeliharaannya.

Isu interaksi manusia dan komputer yang menyangkut daya guna sistem interaktif relevan dengan seluruh aktifitas pada SDLC. Sehingga software engineering untuk sistem interaktif bukan semata-mata menambahkan sebuah tahapan pada SDLC, namun lebih pada melibatkan teknik yang berada sepanjang SDLC itu.

Siklus Hidup Pengembangan Perangkat Lunak (Software Development Life Cycle - SWDLC).

Pengembangan perangkat lunak berjangkauan antara dua sisi ekstrim, dari sindrom “spreadsheet untuk setiap aplikasi” sampai sindrom “reinventing the wheel”. Sindrom pertama terjadi karena untuk setiap aplikasi terdapat spreadsheet yang siap pakai (ready-made) atau terdapat beberapa paket perangkat lunakkomersial yang akan menjalankan aplikasi tersebut. Disisi lain sistem mengembangkan program komputer baru dari pembuatan dari awal (scratch) untuk setiap aplikasi sistem tanpa mempedulikan apa yang telahdikembangkan secara in-house atau apa yang tersedia dari penjual (vendor) perangkat lunak. Pembangunan program mengikuti tiga tahap Siklus Hidup Pengembangan Perangkat Lunak (Software Development Life Cycle-SWDLC), yaitu :

1) Rancangan (Design)
2) Kode (Code)
3) Uji (Test)

Software Development Life Cycle-SWDLC menjadi komponen siklus hidup dari System Development Life Cylce (SDLC) untuk beberapa alasan (D.Suryadi H.S & Bunawan,1995) :

SDLC mencakup pengembangan sistem keseluruhan, yang memerlukan komponen-komponen lain disamping perangkat lunak.

Dalam sistem yang memerlukan pengembangan perangkat lunak yang didasarkan pada rancangan system yang diciptakan oleh SDLC, SWDLC akan diinisiai.

Apabila SWDLC menjadi berperan, maka SWDLC seperti halnya SDLC yang berbasis lebih luas, akan memberikan kumpulan acuan tahap-tahap yang diperlukan untuk mengembangkan perangkat lunak tersebut. SWDLC menjabarkan tugas-tugas dan prosedur-prosedur yang harus dijalankan dalam setiap tahap, hasil yang diciptakan oleh setiap tahap, dan metriks untuk menyusun jadwal, mengestimasi biaya, dan mengukur produktifitas.

SWDLC memberikan atau menyediakan kerangka dan prosedur pengoperasian standar yang mendukung cara pengembangan perangkat lunak yang terstruktur dan terancang dengan baik.

Salah satu model Perangkat Lunak yang dianggap modern saat ini adalah ***CMMI (Capability Maturity Model Integration)*** yang menganggap bahwa seluruh perangkat lunak harus terus menerus mengalami pematangan proses. ***Capability Maturity Model*** disingkat **CMM** adalah model kematangan kapabilitas) adalah suatu model kematangan kemampuan (kapabilitas) proses yang dapat membantu pendefinisian dan pemahaman proses-proses suatu organisasi. Pengembangan model ini dimulai pada tahun 1986 oleh SEI (Software Engineering Institute) Departemen Pertahanan Amerika Serikat di Universitas Carnegie Mellon di Pittsburgh, Amerika Serikat.

CMM awalnya ditujukan sebagai suatu alat untuk secara objektif menilai kemampuan kontraktor pemerintah untuk menangani proyek perangkat lunak yang diberikan. Walaupun berasal dari bidang pengembangan perangkat lunak, model ini dapat juga diterapkan sebagai suatu model umum yang membantu pemahaman kematangan kapabilitas proses organisasi di berbagai bidang. Misalnya rekayasa perangkat lunak, rekayasa sistem, manajemen proyek, manajemen risiko,teknologi informasi, serta manajemen sumber daya manusia.

Secara umum, maturity model biasanya memiliki ciri sebagai berikut:

1. Proses pengembangan dari suatu organisasi disederhanakan dan dideskripsikan dalam wujud tingkatan kematangan dalam jumlah tertentu (biasanya empat hingga enam tingkatan)
2. Tingkatan kematangan tersebut dicirikan dengan beberapa persyaratan tertentu yang harus diraih.
3. Tingkatan-tingkatan yang ada disusun secara sekuensial, mulai dari tingkat inisial sampai pada tingkat akhiran (tingkat terakhir merupakan tingkat kesempurnaan)

4. Selama pengembangan, sang entitas bergerak maju dari satu tingkatan ke tingkatan berikutnya tanpa boleh melewati salah satunya, melainkan secara bertahap berurutan.

Pada tahun 2000 CMM dileburkan ke dalam CMMI (*Capability Maturity Model Integration*). Peleburan ini disebabkan karena adanya kritik bahwa pengaplikasian CMM di pengembangan perangkat lunak khususnya bisa menimbulkan masalah karena model CMM yang belum terintegrasi di dalam dan di seantero organisasi. Ini kemudian memunculkan beban biaya dalam hal pelatihan, penaksiran kinerja, dan aktivitas perbaikan.

Namun CMM masih tetap digunakan sebagai model acuan teoritis di ranah publik untuk konteks yang berbeda. CMM sendiri telah diganti namanya menjadi SE-CMM (*Software Engineering CMM*).

A. Teknnologi Object Oriented

Teknologi object-oriented merupakan paradigma baru dalam rekayasa software yang didasarkan pada obyek dan kelas. Diakui para ahli bahwa object-oriented merupakan metodologi terbaik yang ada saat ini dalam rekayasa software. Object- oriented memandang software bagian per bagian dan menggambarkan satu bagian tersebut dalam satu obyek. Satu obyek dalam sebuah model merupakan suatu focus selama dalam proses analisis, perancangan dan implementasi dengan menekankan pada state, perilaku (behavior) dan interaksi obyek dalam model tersebut.

Teknologi obyek menganalogikan system aplikasi seperti kehidupan nyata yang didominasi oleh obyek. Manusia adalah obyek, komputer adalah obyek. Obyek memiliki atribut : manusia memiliki nama, pekerjaan, rumah, dan lain-lain. Mobil memiliki warna, merk, sejumlah roda, dan lain- lain. Komputer memiliki kecepatan, sistem operasi, dan lain-lain. Obyek dapat beraksi dan bereaksi. Manusia dapat berjalan, berbicara, makan, minum ; mobil dapat berjalan, mengerem ; komputer dapat mengolah data, menampilkan gambar, dan lain-lain.

Keunggulan teknologi obyek dengan demikian adalah bahwa model yang dibuat akan sangat mendekati dunia nyata yang masalahnya akan dipecahkan oleh system yang dibangun. Model obyek, atribut dan kelakuan bias langsung diambil dari obyek yang ada di dunia nyata. Sistem yang dibangun dengan teknologi obyek memiliki fleksibilitas yang tinggi terhadap perubahan karena menggunakan konsep komponen yang bisa digunakan kembali. Didalam dunia perangkat lunak, penggunaan berulang merupakan hal yang biasa. Contohnya ide dan formula yang hampir sama digunakan berulang oleh programmer yang berbeda untuk mengembangkan aplikasi keuangan yang khusus diadaptasikan sesuai kebutuhan dan

permintaan masing-masing klien. Oleh karena itu penggunaan berulang suatu obyek merupakan hal yang seharusnya bisa dilakukan. Suatu obyek bisa diambil untuk dimodifikasi berupa penambahan atau pengulangan untuk memecahkan suatu masalah baru. Ada empat prinsip dasar dari pemrograman berorientasi obyek, yaitu :abstraksi, enkapsulasi, modularitas dan hirarki.

1. ***Abstraksi :***

Memfokuskan perhatian pada karakteristik obyek yang paling penting dan paling dominan yang bias digunakan untuk membedakan obyek tersebut dari obyek lainnya. Dengan abstraksi ini developer bisa menerapkan konsep KIS (Keep It Simple) pada obyek didunia nyata yang memiliki tingkat kerumitan yang tinggi.

2. ***Enkapsulasi :***

Menyembunyikan banyak hal yang terdapat dalam obyek yang tidak perludiketahui oleh obyek lain. Dalam praktek pemrograman enkapsulasi diwujudkan dengan membuat suatu kelas interface yang akan dipanggil oleh obyek lain, sementara didalam obyek yang dipanggil terdapat kelas lain yang mengimplementasikan apa yang terdapat dalam kelas interface. Obyek lain hanya tahu dia perlu memanggil kelas interface tanpa perlu tahu proses apa saja yang dilakukan didalam kelas implementasinya dan untuk menuntaskan proses tersebut perlu berhubungan dengan obyek mana saja. Dengan demikian bila terjadi proses perubahan pada proses implementasi maka obyek pemanggil tidak akan terpengaruhi secara langsung.

3. ***Modularitas :***

Membagi sistem yang rumit menjadi bagian- bagian yang lebih kecil yang bisa mempermudah developer memahami dan mengelola obyek tersebut. Contohnya adalah sistem akademis yang bisa dibagi menjadi kemahasiswaan, daftar mata kuliah, pembayaran uang kuliah, dan lain-lain.

4. ***Hirarki :***

Hirarki berhubungan dengan abstraksi dan modularitas, yaitu pembagian berdasarkan urutan dan pengelompokkan tertentu. Misalnya untuk menentukan obyek mana yang berada pada kelompok yang sama, obyek mana yang merupakan komponen dari obyek yang memiliki hirarki lebih tinggi. Semakin rendah hirarki obyek berarti semakin jauh abstraksi dilakukan terhadap suatu obyek. Ke empat prinsip dasar ini merupakan hal yang mendasari teknologi obyek yang perlu ditanamkan dalam cara berpikir developer berorientasi obyek.

B. Object Oriented Analysis dan Design (OOAD)

Object-oriented mencakup bidang aplikasi yang sangat luas. Para pengguna sistem komputer dan sistem lain yang didasarkan atas teknologi computer merasakan efek object-oriented dalam bentuk meningkatnya aplikasi software yang mudah digunakan dan servis yang lebih fleksibel, yang muncul dalam berbagai bidang industri, seperti dalam perbankan, telekomunikasi, dan sebagainya. Sedangkan bagi software engineer, object-oriented berpengaruh dalam bahasa pemrograman, metodologi rekayasa, manajemen proyek, hardware dan sebagainya.

Analisis dan perancangan berorientasi obyek amat sangat perlu dilakukan dalam pengembangan system berorientasi obyek. Hanya dengan kemampuan menggunakan bahasa pemrograman berorientasi obyek yang andal, kita dapat membangun suatu system berorientasi obyek, namun sistem aplikasi yang dibangun akan menjadi lebih baik lagi bila langkah awalnya didahului dengan proses analisis dan perancangan berorientasi obyek, terutama untuk membangun system yang mudah dipelihara.

Analisis berorientasi obyek adalah metode analisis yang memeriksa requirements (syarat/keperluan yang harus dipenuhi suatu sistem) dari sudut pandang kelas-kelas dan obyek-obyek yang ditemui dalam ruang lingkup permasalahan. Sedangkan perancangan berorientasi obyek adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi obyek-obyek sistem atau subsistem.

Keunggulan Object Oriented :

Sekarang ini terdapat beberapa paradigm yang digunakan dalam rekayasa software, diantaranya : procedure-oriented, object-oriented, data structure - oriented, data flow-oriented dan constraint oriented. Tiap-tiap paradigma tersebut cocok untuk beberapa kasus dan bagian dari seluruh kemungkinan ruang lingkup aplikasi, tetapi menurut Booch berdasarkan pengalamannya, object-oriented dapat diaplikasikan dalam seluruh ruang lingkup. Untuk memahami mengapa OOAD unggul dalam banyak kasus, harus memahami masalah yang dihadapi perusahaan rekayasa software.

Pertama, software sulit untuk dimodifikasi bila memerlukan pengembangan.

Kedua, proses pembuatan software memerlukan waktu yang cukup lama sehingga kadang kala melebihi anggaran dalam pembuatannya.

Ketiga, para pogrammer selalu membuat software dari dasar karena tidak adanya kode yang bisa digunakan ulang (reuse).

Kebanyakan perusahaan tersebut sebelumnya telah menggunakan pemrogramam structural. Metode structural menggunakan pendekatan dengan pendekatan top-down, yang

mana pendekatan ini memecahkan masalah dengan membagi masalah kedalam komponen - komponen hingga didapatkan komponen yang tidak dapat dibagi-bagi lagi. Pendekatan dengan cara top-down ini telah membuat peningkatan dalam kualitas software, tetapi dalam sistem yang berskala besar, pendekatan ini sering menemui banyak masalah. Pemrograman structural seringkali tidak dapat mendesain apa yang akan terjadi dalam sistem yang telah selesai tanpa mengimplementasikan sistem terlebih dahulu. Jika ditemui kesalahan dalam sistem, maka desain harus disusun ulang dari awal sampai akhir. Hal ini akan terjadi pemborosan biaya dan waktu.

Perbedaan antara metode structural dan OOAD terletak pada bagaimana data dan fungsi disimpan. Dalam metode structural, data dan fungsi disimpan terpisah. Biasanya semua data ditempatkan sebelum fungsi ditulis. Seringkali tidak terpikirkan sebelumnya untuk mengetahui data mana yang digunakan dalam suatu fungsi tertentu. Tetapi dalam OOAD, data dan fungsi yang berhubungan dalam suatu obyek disimpan bersama- sama dalam satu kesatuan. Orang akan lebih mudah memahami sistem sebagai obyek daripada prosedur, karena biasanya orang berpikir dalam bentuk obyek. Sebagai contoh, orang melihat mobil sebagai sebuah sistem yang memiliki mesin, roda , stir, tank bahan bakar dan sebagainya. Kebanyakan orang tidak melihat mobil sebagai sebuah urutan prosedur yang membuat mobil tersebut dapat berjalan. Karena secara alamiah lebih mudah memikirkan suatu sistem dalam bentuk obyek-obyek, tidak heran kalau OOAD menjadi lebih terkenal.

Satu alasan mengapa object oriented menguntungkan bagi programmer adalah karena programmer dapat mendesain program dalam bentuk obyek-obyek dan hubungan antar obyek tersebut untuk kemudian dimodelkan dalam sistem nyata. Keuntungan yang lain adalah proses pembuatan software dapat dilakukan dengan lebih cepat karena software dibangun dari obyek-obyek standar, dapat menggunakan ulang model yang ada, dan dapat membuat model yang cepat melalui metodologi.

Kualitas yang tinggi dari software dapat dicapai karena adanya tested component. Lebih mudah dalam maintenance karena perbaikan kode hanya diperlukan pada satu tempat (bukan diurut dari awal). Mudah dalam membangun sistem yang besar karena subsistem dapat dibuat dan diuji secara terpisah. Mengubah sistem yang sudah ada tidak memerlukan membangun ulang keseluruhan sistem. alat untuk desain system.