

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/333209319>

REKAYASA PERANGKAT LUNAK [Software Engineering]

Book · May 2019

CITATIONS

2

READS

29,547

1 author:



Lila Setiyani

STMIK ROSMA

56 PUBLICATIONS 26 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Sistem Informasi Manajemen Akreditasi Balai Pelatihan Kesehatan [View project](#)

REKAYASA PERANGKAT LUNAK

[Software Engineering]

Lila Setiyani, S.T,M.Kom



Karya ini dilindungi Undang-Undang Republik Indonesia
Nomor 28 Tahun 2014 Tentang Hak Cipta

Rekayasa Perangkat Lunak
[Software Engineering]

oleh: Lila Setiyani, S.T, M.Kom
©2018

Desain cover: Lila Setiyani

diterbitkan oleh
Jatayu Catra Internusa

Email : jatayu98@gmail.com

CV. Jatayu Catra Internusa
Jl. Raya Lemahabang – Pancawati
Karawang, Jawa Barat
Telp: 0267-8621351 / 08111282222

Hak Cipta dilindungi oleh Undang-Undang.
Dilarang mengutip atau memperbanyak sebagian atau
seluruh isi buku ini tanpa izin tertulis dari Penerbit.

Buku ini di dedikasikan untuk keluarga “Catra Kastara”

Gatot Catra Kastara

Nayaka Catra Kasatra & Nararya Catra Kastara

Daftar Isi

PENDAHULUAN	VI
BAB 1 REKAYASA PERANGKAT LUNAK	1
BAB 2 PROSES REKAYASA PERANGKAT LUNAK.....	17
BAB 3 PRINSIP – PRINSIP YANG DIGUNAKAN DALAM PRAKTIK REKAYASA PERANGKAT LUNAK.....	79
BAB 4 MEMAHAMI KEBUTUHAN REKAYASA PERANGKAT LUNAK	95
BAB 5 PEMODELAN SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK	126
BAB 6 KONSEP PERANCANGAN PERANGKAT LUNAK	205
BAB 7 MANAJEMEN KUALITAS PERANGKAT LUNAK	267
BAB 8 MANAJEMEN KONFIGURASI PERANGKAT LUNAK	308
DAFTAR PUSTAKA.....	316

Pendahuluan

Perangkat lunak semakin berkembang , seiring dengan kebutuhan manusia yang semakin komplek . Hampir seluruh kegiatan proses bisnis yang terjadi didukung oleh perangkat lunak. Perkembangan ini kadang menuntut peremajaan perangkat lunak. Peremajaan ini dapat berupa keusangan perangkat lunak yang perlu di ganti , pengembangan perangkat lunak untuk menambahkan modul yang kurang atau modifikasi karena perubahan kebutuhan. Perubahan – perubahan yang terjadi pada perangkat lunak ini perlu dipertimbangkan dari berbagai perspektif , baik pengguna , fungsionalnya atau pendukungnya. Karena hal tersebut maka perekayasaan perangkat lunak perlu pendekatan yang khusus.

Ada begitu banyak pendekatan yang dapat digunakan dalam mengembangkan perangkat lunak. Dengan memahami metode dari pendekatan – pendekatan tersebut , para pengembang perangkat lunak akan memiliki arah dalam mengembangkan perangkat lunak yang sesuai kebutuhan pengguna perangkat lunak tersebut.

Buku ajar ini dirancang dan dikembangkan dari sumber – sumber teks book analisis dan desian sistem , disajikan secara khusus dan lengkap tentang pemahaman , pemodelan , perancangan , manajemen kualitas perangkat lunak. Buku ini dapat digunakan sebagai panduan bagi para mahasiswa dalam mempelajari perekayasaan perangkat lunak.

Harapan saya , buku ini dapat memberikan inspirasi dan manfaat bagi para pembaca. Melalui buku ini semoga para pembaca dapat menjadi seorang pengembang perangkat lunak yang

berkualitas, yang memahami kebutuhan pengguna sehingga dapat menyajikan perangkat lunak yang sesuai dengan ekspektasi dan fungsionalitas dari kebutuhan pengguna.

Karena perangkat lunak akan sia – sia jika dalam pengembangannya tidak dapat memenuhi kebutuhan dari penggunanya.

Akhir kata , saya ucapan selamat membaca

Karawang , Desember 2018

Penulis

BAB 1 Rekayasa Perangkat Lunak

Kecepatan proses dalam bisnis yang ada sekarang ini disebabkan karena kecanggihan teknologi perangkat lunak. Perangkat lunak semakin berkembang menyesuaikan kebutuhan manusia. Para pengembang berlomba – lomba menyajikan perangkat lunak yang memiliki efektifitas dan efisiensi dalam mendukung proses yang ada. Tentunya untuk mengembangkan perangkat lunak , kita perlu memahami terlebih dahulu karakteristik , proses dan praktik dari rekayasa perangkat lunak.

I.I. Karakteristik Perangkat Lunak

Pada saat ini , kita banyak sekali menemukan perangkat lunak yang memiliki banyak fungsi , contohnya saja kita lihat perangkat lunak pada telepon seluler , computer , laptop dan lain – lain. Perangkat lunak berperan dalam menyajikan informasi dan menjadi gerbang menuju jaringan informasi global.

Peningkatan perkembangan dari peran perangkat lunak ini didukung oleh teknologi perangkat keras serta perubahan – perubahan dari arsitektur komputer , memori yang mendukung kecepatan proses , penyimpanan yang besar , dan bermacam – macam alat input output yang dapat mendukung, sehingga semua sistem yang berbasis komputer dapat saling terhubung sehingga dapat saling berkomunikasi dan berbagi informasi.

Kecanggihan sistem tersebut , tentunya tidak lepas dari peran para pengembang perangkat lunak yang menjadi faktor dominan.

I.I.I. Definisi Perangkat Lunak

Banyak para pengguna perangkat lunak memahami fungsi dari perangkat lunak dan para ahli telah menjabarkan tentang deskripsi dari perangkat lunak sebagai berikut :

Menurut Ian Sommerville perangkat lunak tidak hanya mencakup program komputer saja, akan tetapi juga termasuk semua dokumentasi dan konfigurasi data yang berhubungan yang diperlukan untuk membuat program beroperasi dengan benar. (Sommerville, 2003)

Sedangkan menurut Pressman mengemukakan bahwa perangkat lunak adalah instruksi – instruksi dalam hal ini adalah program komputer yang ketika dijalankan menyediakan fitur – fitur , fungsi – fungsi dan kinerja – kinerja yang dikehendaki pengguna, dalam perangkat lunak terdapat struktur data yang memungkinkan program – program dapat memanipulasi informasi, dan informasi tersebut tercetak dalam bentuk maya yang menggambarkan pengoperasian dan penggunaan program – program tersebut. (Roger S. Pressman, 2012)

Dari definisi di atas dapat di simpulkan bahwa perangkat lunak adalah suatu program yang dapat beroperasi untuk memanipulasi informasi , yang di dalamnya terdapat instruksi – instruksi berupa fitur atau fungsi tertentu sehingga informasi yang telah di peroleh dapat tercetak dalam bentuk maya.

Namun untuk lebih jelasnya memahami tentang perangkat lunak , kita dapat mengidentifikasi dari karakteristiknya . Berikut adalah karakteristik – karakteristik dari perangkat lunak (Roger S. Pressman, 2012) :

1. Perangkat lunak dikembangkan atau direkayasa, tidak diproduksi dalam konteks manufaktur.
2. Perangkat lunak tidak mengalami kelelahan, namun juga kinerja dapat memburuk, untuk itu di perlukan perancangan perangkat lunak dengan cara yang lebih baik. Metode – metode rekayasa perangkat lunak pada dasarnya bertujuan untuk mengurangi simpangan angka kegagalan rekayasa perangkat lunak.
3. Meskipun industri terus beralih ke konstruksi berbasis komponen, sebagian besar perangkat lunak masih tetap dibuat berdasarkan spesifikasi yang di minta pengguna

Dengan melihat karakteristik tersebut , ini memperlihatkan bahwa karakteristik dari perangkat lunak sangat berbeda dengan karakteristik perangkat keras.

I.I.2. Ranah Aplikasi Perangkat Lunak

Secara umum aplikasi perangkat lunak dibagi kedalam 7 (tujuh) kategori yaitu (Roger S. Pressman, 2012):

1. **Perangkat lunak sistem** , sekumpulan program yang ditulis untuk melayani program – program lain , yang melakukan pemrosesan struktur –

struktur informasi yang kompleks namun umumnya bersifat terbatas. Contoh dari perangkat lunak sistem ini adalah sistem operasi , driver, perangkat lunak jaringan , dll).

2. **Perangkat lunak aplikasi** , program – program mandiri yang menjawab kebutuhan bisnis secara terperinci dengan melakukan pemrosesan data bisnis atau data teknis yang mendukung berjalannya operasi – operasi bisnis.
3. **Perangkat lunak rekayasa** , aplikasi yang telah memiliki algoritma yang penuh dengan kalkulasi data numerik. Contohnya adalah aplikasi untuk kebutuhan vulkanologi , astronomi , dan lain – lain.
4. **Perangkat lunak yang tertanam**, perangkat lunak yang berada pada suatu produk atau sistem dan digunakan untuk menjalankan dan mengendalikan fitur – fitur atau fungsi – fungsi untuk user. Contohnya tampilan indikator pada peralatan elektronik.
5. **Perangkat lunak lini produk**, perangkat lunak yang dirancang untuk menyediakan kemampuan khusus untuk digunakan oleh pelanggan yang bersifat terbatas dan berkonsentrasi pada pasar tertentu . contohnya adalah pengolah kata, *spreadsheet*, dan lain – lain.
6. **Aplikasi web**, perangkat lunak yang berpusat pada jaringan komputer , berisi sekumpulan file *hypertext* yang saling terhubung untuk menunjukkan informasi – informasi tertentu,

berada lingkungan komputasi dan terintegrasi dengan sistem basis data.

7. **Perangkat lunak kecerdasan buatan** , perangkat lunak yang menggunakan algoritma non-numerik untuk memecahkan permasalahan – permasalahan rumit yang tidak dapat di selesaikan dengan komputasi atau analisis permasalahan langsung. Contohnya adalah sistem pakar , robotik, *game*, dll.

Para rekayawan perangkat lunak bekerja keras untuk membuat sistem baru, mengoreksi aplikasi, mengadaptasi dan meningkatkan kemampuannya. Seiring dengan kerja keras tersebut karena adanya perkembangan teknologi maka muncul tantangan – tantangan baru yang harus di hadapi oleh para rekayawan diantanya adalah :

1. Pertumbuhannya jaringan nirkabel yang mengembangkan sistem dan perangkat lunak aplikasi untuk dapat berkomunikasi dengan jaringan yang sangat luas.
2. Kemudahan dalam pencarian sumber informasi melalui internet menantang para rekayawan untuk membuat sebuah aplikasi yang sederhana yang menyediakan manfaat bagi pengguna akhir di seluruh dunia.
3. Berkembangnya *open source* menantang para rekayawan untuk mengembangkan kode – kode dengan teknik – teknik yang

memungkinkan pelanggan dan pengembang tahu perubahan – perubahan yang telah dibuat.

Selain kategori perangkat lunak diatas , dikenal juga adanya perangkat lunak warisan yaitu perangkat lunak yang telah terlebih dahulu dikembangkan . Perangkat lunak warisan ini dicirikan dengan panjangnya usia dan kekritisan bisnis. Perangkat lunak warisan ini bisa saja berkualitas buruk , perancangan yang tidak memungkinkan untuk diperluas lagi, kode – kode rumit, buruknya dokumentasi , pengujian dan hasilnya tidak pernah diarsipkan, namun perangkat lunak warisan ini mendukung fungsi – fungsi bisnis inti. Jika perangkat lunak warisan ini memenuhi kebutuhan penggunanya dan beroperasi dengan baik tentunya tidak perlu diperbaiki. Namun, perangkat lunak warisan perlu dirubah atau dikembangkan karena beberapa alasan (Roger S. Pressman, 2012) :

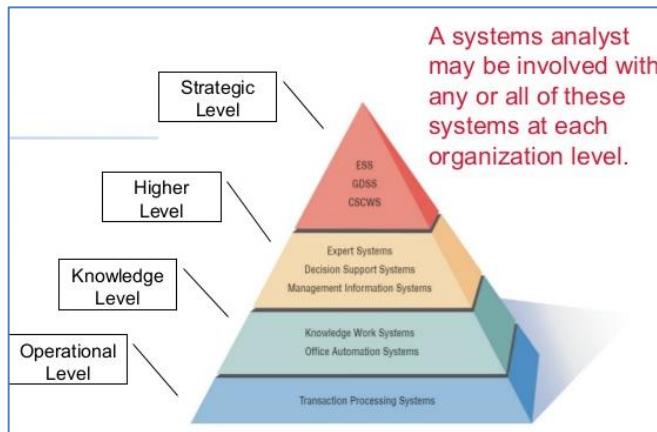
1. Perlu diadaptasikan sehingga memenuhi kebutuhan lingkungan atau teknologi komputasi yang baru.
2. Perlunya ditingkatkan kinerjanya supaya menjalankan kebutuhan – kebutuhan bisnis baru.
3. Perlu di perluas agar dapat saling berkomunikasi dengan perangkat lunak modern atau basis data modern.
4. Perlu dirancang ulang agar hidup dalam lingkungan jaringan komputer.

Dalam ranah perangkat lunak yang telah di jelaskan diatas , yang paling populer adalah aplikasi – aplikasi web. Menurut Powel aplikasi web pada dasarnya mempunyai sifat – sifat berikut (Roger S. Pressman, 2012) :

1. Kepadatan Jaringan , aplikasi web biasanya berada pada suatu jaringan komputer dan harus melayani client yang beragam.
2. Keserempakan, sejumlah besar pengguna mungkin akan mengakses aplikasi web secara serempak.
3. Jumlah pengguna akhir tidak dapat diprediksi.
4. Kinerja, jika aplikasi web yang dibuka tidak memiliki *perfomace* yang tinggi , maka pengguna akan meninggalkannya.
5. Ketersediaan, para pengguna menuntut akses 24 jam setiap harinya.
6. Digerakkan oleh data , menyajikan data baik itu teks, grafis, audio dan video.
7. Peka terhadap isi, kualitas dan karakter isi merupakan faktor penentu yang penting dari kualitas aplikasi.
8. Kesegaran, aplikasi web menunjukkan kedisiplinan waktu untuk merilis produk ke pasar dalam waktu beberapa hari saja.
9. Keamanan , karena aplikasi web tersedia melalui akses jaringan komputer , maka sulit membatasi jumlah pengguna akhir .
10. Estetika , daya tarik aplikasi web adalah tampilan dan nuansanya.

1.2. Tipe dari sistem atau perangkat lunak berdasarkan tujuan pengembangannya

Berdasarkan tujuan pengembangan sistem dengan mempertimbangkan kebutuhan manusia dan bisnis , sistem terbagi menjadi (Kenneth E. Kendall, 2010):



Gambar 1.1 Tipe dari sistem berdasarkan tujuan pengembangannya

Sumber : (Kenneth E. Kendall, 2010)

1. *Transaction processing system (TPS)*
Merupakan sistem informasi terkomputerisasi yang dikembangkan dengan tujuan untuk memproses sejumlah data yang besar untuk transaksi bisnis yang rutin. Biasanya sistem ini digunakan untuk berinteraksi dengan lingkungan eksternal.
2. *Office Automation System (OAS)*
Merupakan sistem yang mendukung pekerjaan data , yang biasanya digunakan untuk menciptakan suatu

pengetahuan baru dengan menganalisi informasi , dalam analisis tersebut dapat dilakukan manipulasi data dengan cara – cara tertentu sebelum di bagikan secara formal atau di sebarluaskan ke seluruh organisasi.

3. *Knowledge work system(KWS)*

Merupakan sistem yang mendukung para pekerja profesional seperti ilmuwan , dokter dan lain – lain dalam menciptakan pengetahuan baru dan memungkinkan mereka berkontribusi pada organisasi atau masyarakat luas.

4. *Management Information system (MIS)*

Merupakan sistem informasi terkomputerisasi yang berfungsi karena interaksi manusia dan komputer. Dengan mengharuskan manusia, perangkat lunak dan perangkat keras berkerjasama untuk mendukung organisasi dalam melakukan pemrosesan traksaksi , termasuk dalam analisis keputusan dan pengambilan keputusan.

5. *Decision support system (DSS)*

Merupakan sistem informasi terkomputerisasi level yang tinggi karena bergantung pada basis data yang merupakan sumber data , fungsi dari sistem ini adalah mendukung proses pengambilan keputusan sehingga sistem ini berfokus pada bisnis *intelligence*.

6. *Artificial Intelligence(AI) dan Expert System (ES)*

Expert system merupakan sistem berbasis pengetahuan yang secara efektif menangkap dan menggunakan pengetahuan para ahli untuk memecahkan masalah tertentu yang dialami suatu organisasi. *Expert system* menggunakan penalaran *Artificial Intelligence*.

7. *Group decision support system (GDSS) dan Computer-Supported collaborative work system (CSCWS)*

- Merupakan sistem informasi yang digunakan secara bersama – sama untuk suatu kelompok , digunakan untuk mendukung proses pengambilan keputusan dalam suatu organisasi.
8. *Executive Support System*
Merupakan sistem informasi yang mendukung eksekutif mengatur interaksi mereka dengan lingkungan eksternal. Sistem ini menyediakan dukungan grafis dan komunikasi di tempat yang dapat diakses dari kantor atau ruang pribadi. Sistem ini mengandalkan informasi dari TPS dan MIS , sistem ini memperluas dan mendukung kemampuan para eksekutif sehingga memungkinkan mereka untuk memahami lingkungan mereka.

I.3. Rekayasa Perangkat Lunak

Untuk membangun perangkat lunak , sebagai rekayawan harus memahami bahwa perangkat lunak kini telah menyalu dalam kehidupan banyak orang, maka seoarang rekayawan harus memahami masalah terlebih dahulu sebelum memberikan solusi terhadap perangkat lunak yang akan dikembangkan. Perancangan menjadi sesuatu yang sangat penting agar menghasilkan perangkat lunak yang berkualitas tinggi . Perangkat lunak berkualitas tinggi adalah perangkat lunak yang memiliki nilai bisnis bagi penggunanya , berusia panjang dan tentunya perangkat lunak tersebut harus bersifat dapat dipelihara atau dirawat. Dengan melihat kenyataan tersebut maka perangkat lunak dalam segala bentuk kategorinya membutuhkan untuk di rekayasa.

Menurut Rosa rekayasa perangkat lunak merupakan pembangunan dengan menggunakan prinsip atau konsep

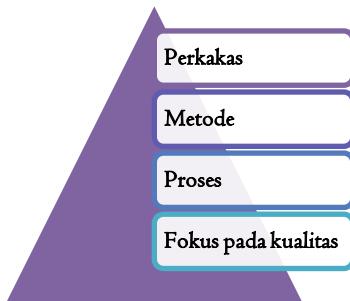
rekayasa dengan tujuan untuk menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin. (Rosa A. S, 2018).

IEEE mengembangkan definisi rekayasa perangkat lunak yaitu suatu aplikasi dari pendekatan yang sistematik, disiplin dan dapat diukur pada pengembangan , operasi dan perawatan lunak. (Roger S. Pressman, 2012).

Sedangkan menurut Ian mengemukakan bahwa rekayasa perangkat lunak adalah disiplin ilmu yang membahasa semua aspek produksi perangkat lunak , mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan. (Sommerville, 2003).

Jadi dapat disimpulkan bahwa rekayasa perangkat lunak adalah proses produksi perangkat lunak yang menggunakan pendekatan sistematik , disiplin dan dapat diukur yang pada akhirnya menghasilkan perangkat lunak yang bekerja secara efisien .

Pressman menyatakan bahwa segala pendekatan rekayasa perangkat lunak harus bersandar pada komitmen organisasi pada peningkatan kualitas. Untuk lebih jelas berikut adalah gambaran lapisan – lapisan rekayasa perangkat lunak (Roger S. Pressman, 2012) ;



Gambar 1.2 Lapisan – lapisan rekayasa perangkat lunak

Pondasi dari tindakan yang berkaitan dengan rekayasa perangkat lunak adalah lapisan proses. Proses perangkat lunak mendefinisikan kerangka kerja yang harus dibangun, serta membentuk kendali manajemen proyek dan membangun konteks. Metode rekayasa perangkat lunak berisi tentang prosedur untuk mengembangkan perangkat lunak. Dalam prosedur tersebut terdapat *task* yang menyertakan komunikasi, analisis kebutuhan, pemodelan rancangan, pembuatan program, pengujian dan dukungan – dukungan untuk *user*. Perkakas rekayasa perangkat lunak menyajikan dukungan semiotomatis maupun otomatis bagi proses dan metode – metode.

I.4. Proses Perangkat Lunak

Proses merupakan sekumpulan aktivitas, aksi dan tugas yang dijalankan ketika suatu produk harus dibuat. Aktivitas pada dasarnya berupaya mencapai tujuan umum, aksi mencakup didalamnya sederetan tugas – tugas yang menghasilkan suatu produk kerja utama, tugas pada dasarnya berkonsentrasi pada sebuah tujuan yang kecil namun terdefinisi dengan baik.

Kerangka kerja proses membangun dasar bagi proses rekayasa perangkat lunak yang lengkap dengan cara mengidentifikasi sejumlah kecil aktivitas kerangka kerja yang cocok bagi semua proyek rekayasa perangkat lunak. Kerangka kerja proses pada rekayasa perangkat lunak terdiri atas lima aktivitas berikut (Roger S. Pressman, 2012) :

1. **Komunikasi**, bertujuan untuk memahami tujuan – tujuan stakeholder atas proyek perangkat lunak yang sedang dikembangkan dan mengumpulkan kebutuhan – kebutuhan yang akan membantu mendefinisikan fitur – fitur perangkat lunak berikut dengan fungsi – fungsinya.
2. **Perencanaan**, kegiatan perencanaan menciptakan suatu peta yang dapat membantu membimbing tim perangkat lunak. Rencana proyek perangkat lunak menggambarkan risiko – risiko yang mungkin muncul, sumber daya yang akan dibutuhkan, produk – produk kerja yang harus dihasilkan dan *schedule* kerja.
3. **Pemodelan**, dilakukan bertujuan untuk membuat sketsa sehingga tim perangkat lunak dapat memahami gambaran besar produk yang akan dibuat.
4. **Konstruksi**, kegiatan yang menggabungkan pengkodean dan pengujian .
5. **Penyerahan perangkat lunak kepada user** , penyajian perangkat lunak kepada *user* untuk di evaluasi.

Lima aktivitas diatas dapat juga berulang , pengulangan tersebut menghasilkan sebuah peningkatan perangkat lunak yang memberikan *user* suatu fitur dan fungsionalitas perangkat lunak. Aktivitas – aktivitas kerangka kerja perangkat lunak

diatas disempurnakan dengan aktivitas – aktivitas yang bertindak sebagai penyangga, yaitu :

1. **Penelusuran dan kendali proyek perangkat lunak**, dilakukannya penilaian terhadap kemajuan proyek perangkat lunak yang berjalan sehingga dapat diambil tindakan – tindakan untuk membuat tepat waktu.
2. **Manajemen risiko**, menilai risiko – risiko yang dapat berpengaruh terhadap hasil akhir dan kualitas perangkat lunak
3. **Penjaminan kualitas perangkat lunak**, melakukan kegiatan – kegiatan yang memastikan kualitas perangkat lunak.
4. **Ulasan** , mengevaluasi produk untuk menemukan dan menyinkirkan *defect*.
5. **Pengukuran**, menjelaskan dan mengukur proses, proyek dan produk perangkat lunak yang akan membantu tim perangkat lunak dalam memenuhi kebutuhan stakeholder.
6. **Manajemen konfigurasi perangkat lunak**
7. **Manajemen penggunaan ulang**
8. **Persiapan produk kerja dan produksi**

I.5. Praktik Rekayasa Perangkat Lunak

Kegiatan kerangka kerja dari perangkat lunak membentuk kerangka arsitektur bagi pekerjaan perangkat lunak, namun bagaimana praktik implementasi dari rekayasa perangkat lunak yang baik. Untuk itu perlu di pahami konsep dan prinsip yang merujuk pada kegiatan kerangka kerja tersebut.

I.5.I. Esensi Praktik

Menurut George Poyla dalam (Roger S. Pressman, 2012) menjelasakan esensi dari praktik rekayasa perangkat lunak yaitu :

1. Pahami masalahnya
2. Rancang solusinya
3. Laksanakan rancangan
4. Periksa ketepatan hasilnya

I.5.2. Prinsip – prinsip Umum dari Praktek Rekayasa Perangkat Lunak.

Davis Hooker dalam (Roger S. Pressman, 2012) menyatakan ada tujuh prinsip pada praktik rekayasa perangkat lunak :

1. Alasan keberadaan perangkat lunak – memberikan nilai tertentu bagi para penggunanya.
2. Tetap sederhana , sehingga mendukung terbentuknya sistem yang lebih mudah dipahami dan di rawat.
3. Pertahankan visi, tanpa visi yang jelas, proyek perangkat lunak hampir pasti akan berakhir menjadi dua atau lebih pemikiran.
4. Apa yang anda buat, akan digunakan oleh konsumen, dengan mempermudah pekerjaan konsumen , maka akan menambahkan nilai tertentu pada perangkat lunak yang dibangun.
5. Membuka diri terhadap masa depan, perangkat lunak harus dapat beradaptasi dengan perubahan – perubahan sehingga dapat berumur panjang dan nilai lebih.

6. Rancanglah selangkah ke depan sehingga dapat digunakan kembali, perencanaan selangkah kedepan untuk penggunaan uang akan dapat mengurangi biaya dan akan meningkatkan nilai dari perangkat lunak yang dibangun.
7. Pikirlah, efek samping dari berfikir adalah belajar untuk mengenal ketika anda tidak mengetahui sesuatu yang disaat yang sama anda dapat meneliti jawabannya.

I.6. Penggunaan CASE TOOLS

Dalam proses pengembangan perangkat lunak , agar mendapatkan produktivitas yang tinggi , analis dapat menggunakan alat produktivitas yang disebut dengan Computer Aided Software Engineering (CASE), yang dibuat secara eksplisit untuk meningkatkan produktivitas, komunikasi yang lebih efektif dengan pengguna dan mengintegrasikan pekerjaan mereka dari awal sistem hingga akhir siklus sistem (Kenneth E. Kendall, 2010).

Salah satu contoh dari CASE tools adalah *Visible Analyst* yang digunakan untuk perancangan grafis , analisis dan desain untuk membangun aplikasi , database dan lain – lain. Contoh CASE tools adalah Microsoft office visio , star UML , Blasamiq Mockups , dan lain – lain

BAB 2 Proses Rekayasa Perangkat Lunak

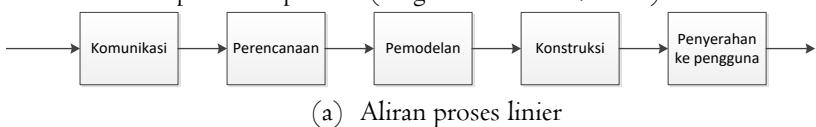
Proses perangkat lunak mendefinisikan pendekatan yang harus diambil saat perangkat lunak di rekayasa. Proses perangkat lunak juga mencakup teknologi – teknologi yang dibutuhkan oleh proses perangkat lunak. pada proses perangkat lunak terdapat beberapa pendekatan yang terangkum dalam aktivitas kerangka kerja perangkat lunak . Pada bab 2 ini akan dibahas model – model pendekatan yang dapat digunakan pada proses perangkat lunak.

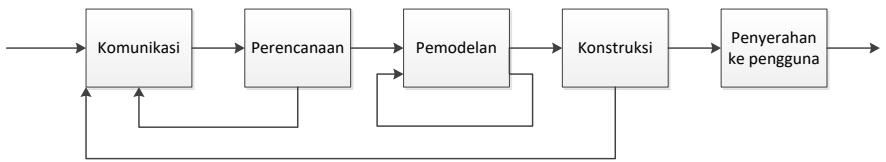
2.I. Model – model yang digunakan dalam proses rekayasa perangkat lunak

Pada kenyataannya ada beberapa model proses diantaranya :

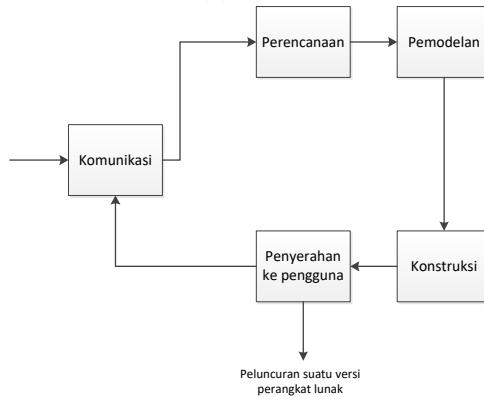
2.I.I. Model Proses generik

Secara garis besar model proses yang telah dibahas pada bab I adalah model proses generik yang memiliki 5 aktivitas kerangka kerja dan ditunjang dengan sejumlah aktivitas – akivitas pendukung. Pada dasarnya proses dari rekayasa perangkat lunak dapat menjadi sebuah aliran proses yang masing – masing memiliki urutan akivitas kerja yang berbeda – beda disesuaikan berdasarkan lingkup proyeknya , , Pressman menggambarkan pada model proses generik terdapat empat aliran proses . (Roger S. Pressman, 2012)

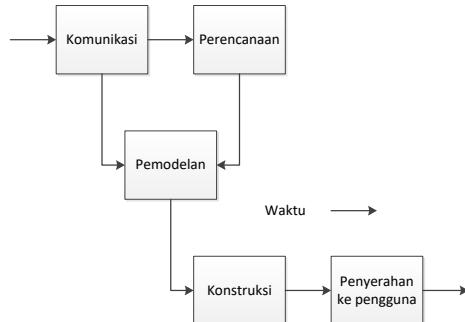




(b) Aliran proses iteratif



(c) Aliran proses evolusioner



(d) Aliran proses paralel

Gambar 2.1 Aliran proses

Pada proses perangkat lunak maka ada beberapa hal yang perlu diperhatikan agar proses perangkat lunak dapat berjalan dengan baik (Roger S. Pressman, 2012):

1. Mendefinisikan aktivitas kerangka kerja, pendefinisan ini dilakukan untuk dapat menentukan tindakan – tindakan yang sesuai dengan aktivitas kerangka kerja yang sesuai dengan permasalahan yang akan diselesaikan dan menentukan karakteristik orang – orang yang akan mengerjalan serta para stakeholder yang akan membiayai proyek.
2. Mengidentifikasi himpunan pekerjaan, mendefinisikan pekerjaan – pekerjaan yang harus di selesaikan untuk memenuhi sasaran tertentu dari suatu aksi rekayasa perangkat lunak.
3. Pendiskripsian pola – pola proses, mendiskripsikan permasalahan – permasalahan yang terkait dengan proses, yang dijumpai selama pekerjaan rekayasa perangkat lunak berlangsung. Pola – pola proses ini menyediakan template untuk menyelesaikan suatu masalah sehingga tim perangkat lunak dapat menyelesaikan permasalahan – permasalahan dan membentuk proses yang paling sesuai dengan kebutuhan dari suatu proyek yang sedang dikerjakan.
4. Penilaian dan perbaikan proses, proses perangkat lunak dapat dinilai untuk memastikan bahwa perangkat lunak tersebut telah memenuhi sejumlah kriteria proses dasar yang esensial. Ada beberapa pendekatan penilaian proses perangkat lunak diantarnya :
 - a. SCAMPI (Standard CMMI Assesment Method for Process Improvement)

- menyediakan model penilaian dengan lima tahapan yaitu melakukan pemberian nilai awal (*initiating*), melakukan diagnosa (*diagnosing*), penetapan(*establishing*), bertindak(*acting*), dan belajar(*learning*).
- b. **CBA IPI (CMM-Based Appraisal for Internal Process Improvement).** Pendekatan yang menggunakan teknik diagnosa untuk melakukan penilaian kematangan perangkat lunak.
 - c. **ISO 9001:2000 for software.** Merupakan standar generik penilaian untuk melakukan perbaikan kualitas produk perangkat lunak.

2.1.2. Model Proses Perspektif

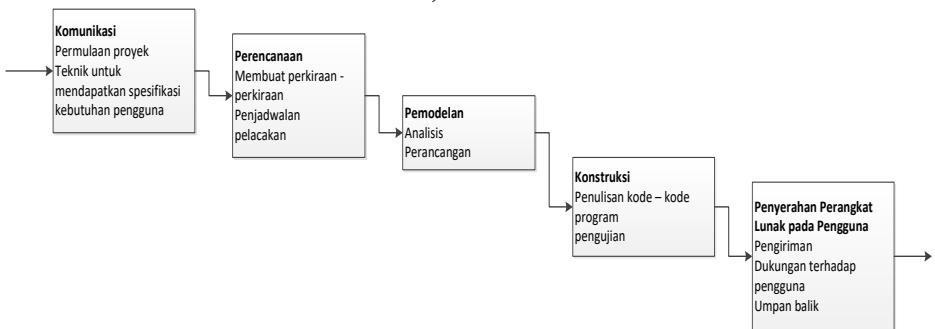
Disebut model proses perspektif karena pendekatan ini memberikan sejumlah elemen proses yaitu berupa aktivitas – aktivitas kerangka kerja, tindakan – tindakan rekayasa perangkat lunak, produk – produk kerja , penilaian jaminan kualitas dan mekanisme kendali perubahan untuk setiap proyek perangkat lunak. berikut adalah beberapa pendekatan dari model proses perspektif :

I. Model Air Terjun (*waterfall*) atau siklus hidup klasik (*classic life cycle*)

Menurut Rosa model air terjun ini menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis,

desain, pengkodean, pengujian dan tahap pendukung. (Rosa A. S, 2018)

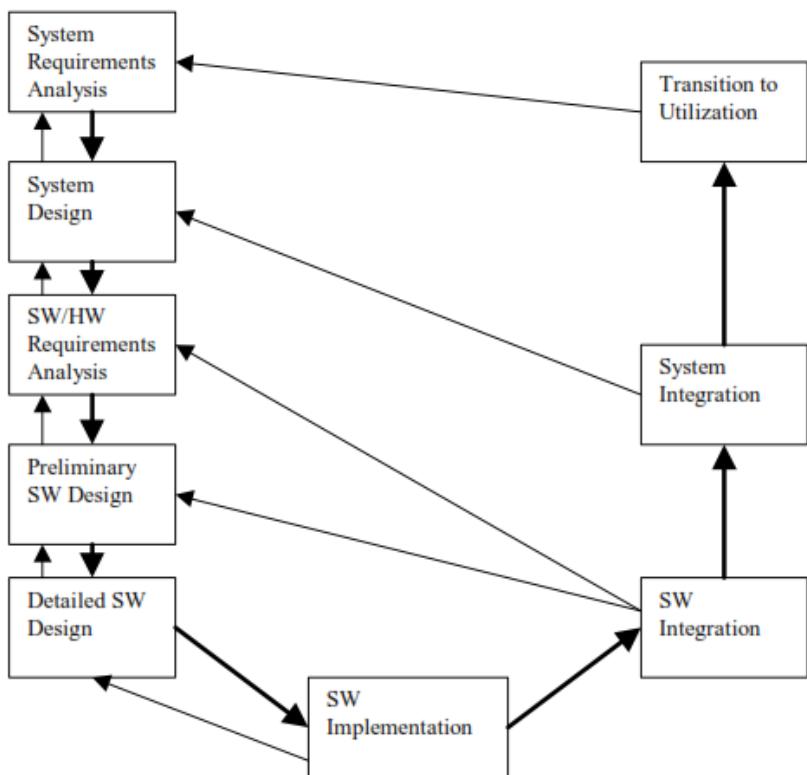
Sedangkan Pressman mengemukakan bahwa model air terjun ini menyiratkan pendekatan yang sistematis dan berurutan pada perangkat lunak yang dimulai dari perencanaan (*planning*), pemodelan(*modeling*), konstruksi(*construction*), serta penyerahan perangkat lunak kepada pelanggan (*deployment*), yang diakhiri dengan dukungan berkelanjutan pada perangkat lunak lengkap yang dihasilkan . Model air terjun ini dapat digambarkan sebagai berikut (Roger S. Pressman, 2012):



Gambar 2.2 Model air terjun

Pada perkembangannya Bucanac mengemukakan model air terjun ini di variasi menjadi model-V, yang pada dasarnya model-V ini menyediakan secara visual bagaimana tindakan – tindakan verifikasi dan validasi yang seharusnya diterapkan pada bagian – bagian rekayasa perangkat lunak yang lebih awal

(Bucanac, 2018). Model-V digambarkan sebagai berikut :



Gambar 2.3 Model-V

Pada implementasinya pendekatan menggunakan model air terjun ini banyak sekali ditemukan kekurangan . Hanna mengemukakan beberapa kekurangan dari model air terjun ini

antara lain pada kenyataannya proyek perangkat lunak jarang mengikuti aliran sekuensial seperti yang diusulkan model air terjun , sering kali pelanggan sulit menetapkan semua spesifikasi kebutuhan secara eksplisit sedangkan model air terjun sulit untuk mengakomodasi ketidakpastian yang selalu harus pada tahapan proyek, dan kebanyakan pelanggan tidak memiliki kesabaran untuk melihat program sampai proyek berakhir (Roger S. Pressman, 2012).

Bradac juga mengemukakan kekurangan dari model air terjun ini, menurutnya tahapan pada model air terjun cenderung terkunci sehingga banyak anggota tim proyek perangkat lunak harus menghabiskan waktu untuk menunggu anggota tim lainnya bekerja hingga dapat menyelesaikan pekerjaannya sendiri (Roger S. Pressman, 2012).

Berikut kelebihan dan kekurangan pendekatan *waterfall*(tatvasoft, 2015) :

Keuntungan dari Model Waterfall:

- a. Model Waterfall sangat sederhana dan mudah dimengerti dan menggunakan metode yang sangat bermanfaat untuk pemula atau pengembang pemula
- b. Sangat mudah untuk dikelola, karena kekakuan model. Selain itu, setiap fase memiliki kiriman spesifik dan proses peninjauan individu

- c. Dalam fase model ini diproses dan diselesaikan sekaligus dalam satu waktu sehingga menghemat banyak waktu.
- d. Jenis model pengembangan ini bekerja lebih efektif dalam proyek-proyek yang lebih kecil di mana persyaratan sangat dipahami dengan baik.
- e. Pengujian lebih mudah karena dapat dilakukan dengan mengacu pada skenario yang didefinisikan dalam spesifikasi fungsional sebelumnya.

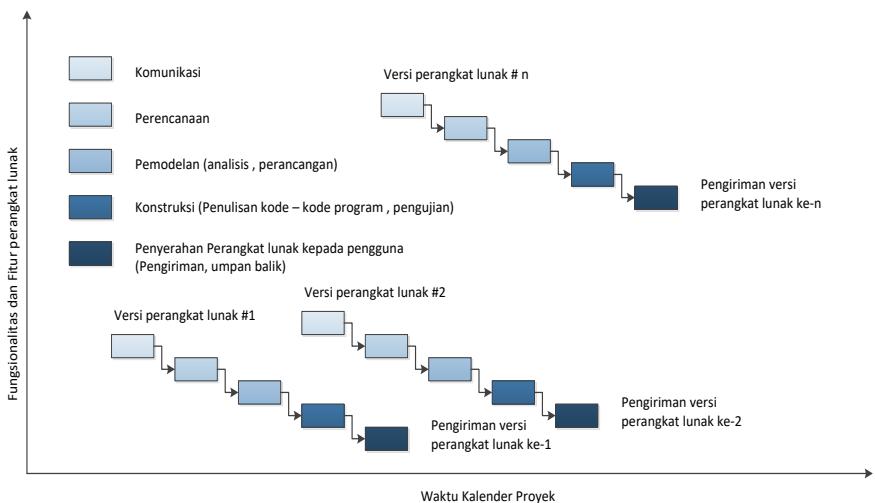
Kekurangan Model Air Terjun:

- a. Model ini hanya dapat digunakan ketika persyaratan depan yang sangat tepat tersedia.
- b. Model ini tidak berlaku untuk jenis proyek pemeliharaan.
- c. Kelemahan utama dari metode ini adalah bahwa sekali aplikasi berada dalam tahap pengujian, tidak mungkin untuk kembali dan mengedit sesuatu.
- d. Tidak ada kemungkinan untuk menghasilkan perangkat lunak yang bekerja sampai mencapai tahap terakhir dari siklus.
- e. Dalam model ini, tidak ada pilihan untuk mengetahui hasil akhir dari keseluruhan proyek.
- f. Model ini bagus untuk proyek kecil tetapi tidak cocok untuk proyek yang panjang dan berkelanjutan.

g. Tidak ideal untuk proyek yang persyaratannya sangat moderat, dan ada ruang lingkup yang bagus untuk modifikasi.

2. Model Proses Inkremental

Model inkremental atau disebut dengan model penambahan sedikit demi sedikit ini menggabungkan elemen – elemen aliran proses dari model generik yaitu aliran proses linier dan aliran proses paralel (Roger S. Pressman, 2012). Masing – masing urutan linier menghasilkan bagian penambahan dari perangkat lunak dengan cara serupa dengan penambahan sedikit demi sedikit yang dihasilkan oleh suatu aliran proses yang bersifat evolusioner. Contohnya adalah perangkat lunak pengolah kata yang dikembangkan dengan cara penambahan sedikit demi sedikit dengan menambahkan fungsi – fungsi pengelolaan berkas, penyuntingan dan fungsi – fungsi dokumen yang dibuat pada tahap pertama, kemudian fungsi penyuntingan dan fungsi produksi yang lebih canggih pada tahap kedua. Pada setiap aliran proses setiap penambahan dapat digabungkan dengan pembentukan prototipe. Produk yang dihasilkan pada tahap pertama merupakan produk inti , produk yang dihasilkan pada tahap kedua dan selanjutnya merupakan fitur – fitur tambahan yang merupakan pengembangan dari produk tersebut. Model inkremental digambarkan sebagai berikut :



Gambar 2.4 Model Inkremental

Pada pengembangan perangkat lunak dengan model proses inkremental ini sangat bermanfaat pada strategi pemnafaatan sumber daya , dan juga dapat direncanakan dengan mempertimbangkan risiko – risiko teknik tertentu.

3. Model Proses Evolusioner

Model proses evolusioner adalah model proses perangkat lunak yang dirancang untuk mengakomodasi suatu produk perangkat lunak yang akan berubah secara perlahan sepanjang waktu. Model proses evolusioner ini digunakan untuk menjawab tantangan perubahan kebutuhan perangkat lunak yang semakin kompleks. Model proses evolusioner ini bersifat

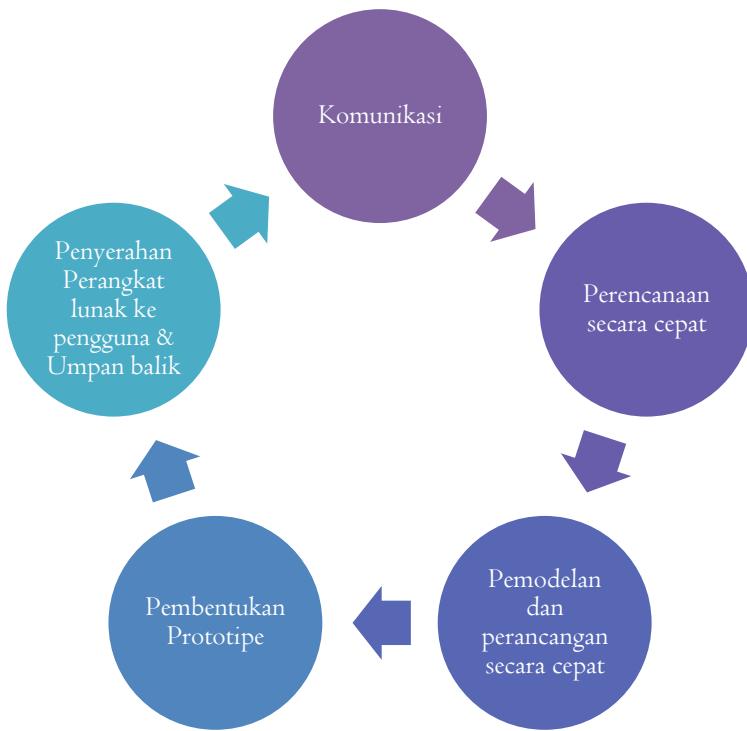
iteratif yang dicirikan dalam bentuk yang memungkinkan pengembang perangkat lunak untuk mengembangkan produknya menjadi semakin kompleks pada versi – versi berikutnya. Model proses evolusioner ini terbagi menjadi beberapa paradigma pendekatan yaitu :

Pembuatan prototipe (*prototyping*).

Pengguna perangkat lunak kadang mendefinisikan kebutuhannya secara umum, dan tidak dapat mendefinisikan kebutuhan secara rinci terkait fitur – fitur atau fungsi – fungsi yang nantinya akan dikembangkan. Dengan melihat kasus ini maka pengembang tidak memiliki kepastian terkait efisiensi algoritma yang digunakan untuk mengembangkan perangkat lunak yang dibutuhkan oleh pengguna. Pada kasus tersebut , pendekatan yang paling baik adalah menggunakan paradigma pembuatan prototipe. Dalam hal ini paradigma pembuatan prototipe seringkali membantu tim pengembang perangkat lunak dan para stakeholder untuk memahami kebutuhan perangkat lunak yang akan dikembangkan.

Pembuatan prototipe dimulai dengan dilakukannya komunikasi antara tim pengembang perangkat lunak dengan pelanggan, kemudian menetapkan sasaran pengembangan secara keseluruhan dan mengidentifikasi spesifikasi kebutuhan. Pembuatan prototipe direncanakan dan dirancang dengan cepat.

Rancangan kemudian di kontruksi untuk membuat prototipe. Prototipe kemudian diserahkan kepada stakeholder dan mereka melakukan evaluasi terhadap prototipe tersebut, yang pada akhirnya memberikan umpan balik yang akan digunakan untuk memperbaiki atau mengembangkan spesifikasi kebutuhan perangkat lunak. Pengulangan yang terjadi pada saat prototipe diperbaiki secara tidak langsung memenuhi kebutuhan stakeholder, dan pada saat yang sama memungkinkan pengembang memahami lebih dalam kebutuhan dari perangkat lunak yang dikerjakan. Semakin banyak pengembangan maka perangkat lunak berevolusi memenuhi kebutuhan pengguna. Paradigma pembuatan prototipe digambarkan sebagai berikut :



Gambar 2.5 Paradigma pembuatan prototipe
Berikut adalah kelebihan dan kekurangan pendekatan prototipe (tatvasoft, 2015).

Keuntungan Model Prototype:

- Ketika prototipe ditunjukkan kepada klien, mereka mendapatkan pemahaman yang jelas dan menyelesaikan 'rasa' fungsionalitas perangkat lunak.
- Metode ini secara signifikan mengurangi risiko kegagalan, karena potensi risiko

- dapat diidentifikasi pada tahap awal dan langkah-langkah moderasi dapat diambil dengan cepat.
- c. Komunikasi antara tim pengembangan perangkat lunak dan klien membuat lingkungan yang sangat baik dan kondusif selama proyek.
 - d. Ini membantu dalam pengumpulan kebutuhan dan analisis kebutuhan ketika ada kekurangan dokumen persyaratan.

Kekurangan Model Prototype:

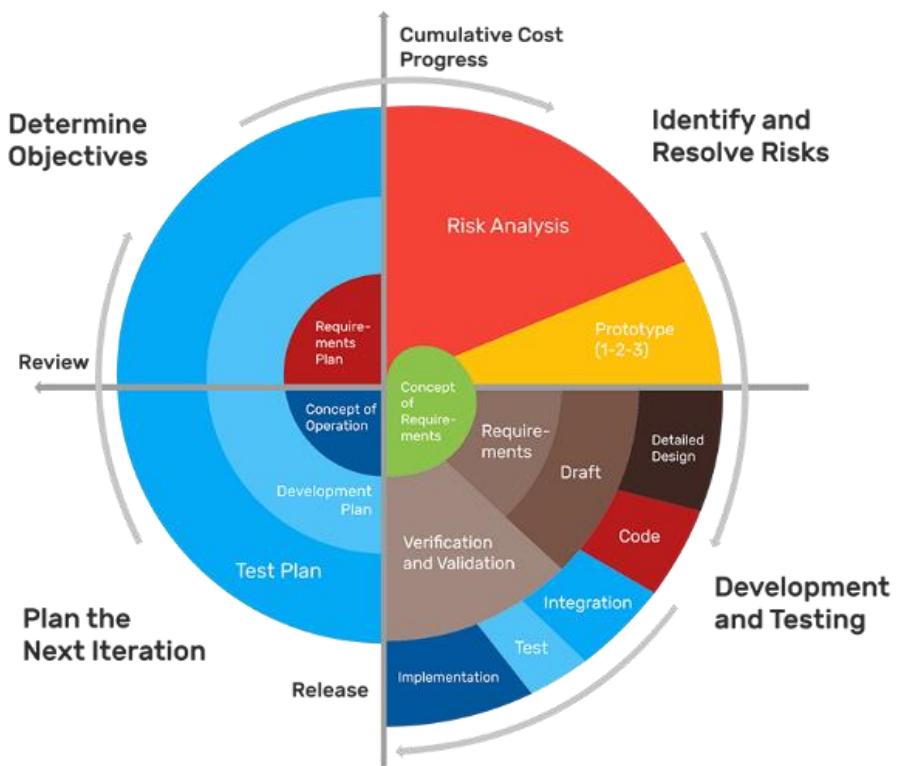
- a. Pembuatan prototipe biasanya dilakukan dengan biaya pengembang, jadi harus dilakukan dengan menggunakan sumber daya minimal jika biaya pengembangan organisasi terlalu banyak.
- b. Terlalu banyak keterlibatan klien tidak selalu disukai oleh pengembang perangkat lunak.
- c. Terlalu banyak modifikasi mungkin tidak baik untuk proyek, karena mudah mengganggu alur kerja seluruh tim pengembangan perangkat lunak

Model Spiral

Model spiral merupakan suatu model proses perangkat lunak evolusioner yang menggabungkan pendekatan *prototyping* yang

bersifat iteratif dengan aspek – aspek sistematis dan terkendali yang di jumpai pada model air terjun. Menggunakan model spiral , perangkat lunak dikembangkan mengikuti peluncuran produk yang bersifat evolusioner. Selama tahap awal, produk perangkat lunak yang di luncurkan mungkin berupa sebuah model atau suatu prototipe. Pada langkah – langkah iterasi berikutnya versi – versi perangkat lunak yang semakin lengkap akan dihasilkan (Roger S. Pressman, 2012).

Model pengembangan spiral dimulai dengan fase perencanaan, yang mengevaluasi elemen kerja utama dari proyek seperti tujuan, kondisi pasar dan banyak lagi. Fase berikutnya, analisis risiko mencoba memecahkan masalah dalam lebih dari satu cara. Fase ketiga adalah menyelesaikan pekerjaan, diikuti oleh ulasan pelanggan yang bergantung pada umpan balik dari pelanggan. Iterasi berhenti ketika penyampaian akhir dikembangkan. Dengan cara ini baik klien dan tim pengembangan berinteraksi secara berkala, meningkatkan produk yang dapat dikirimkan satu per satu (Receptives, 2018). Berikut gambar model spiral :



Gambar 2.6 Spiral Methodology

Sumber : (Receptives, 2018)

Model spiral dibagi menjadi sejumlah aktivitas kerangka kerja yang didefinisikan oleh pengembang perangkat lunak. Lintasan pertama di sekitar spiral mungkin menghasilkan sesuatu yang penting untuk spesifikasi produk, lintasan – lintasan selanjutnya di sekitar spiral mungkin digunakan untuk mengembangkan suatu

prototipe dan kemudian pada lintasan – lintasan berikutnya tim pengembang bisa secara progresif bergerak ke versi – versi perangkat lunak selanjutnya yang semakin canggih.

Walaupun model spiral seolah – olah dapat memberikan kemudahan dalam pengembangan perangkat lunak namun Pressman menyampaikan bahwa model spiral membutuhkan pakar – pakar yang bisa melakukan penilaian risiko dengan cara yang sempurna , dan model ini mengandalkan dirinya pada pakar – pakar terkait agar berhasil, jika risiko utama tidak tersingkap dan tidak bisa dikelola dengan baik, permasalahan – permasalahan yang buruk sangat mungkin akan terjadi (Roger S. Pressman, 2012).

Berikut adalah kelebihan dan kekurangan pendekatan spiral (tatvasoft, 2015).

Keuntungan dari Model Spiral:

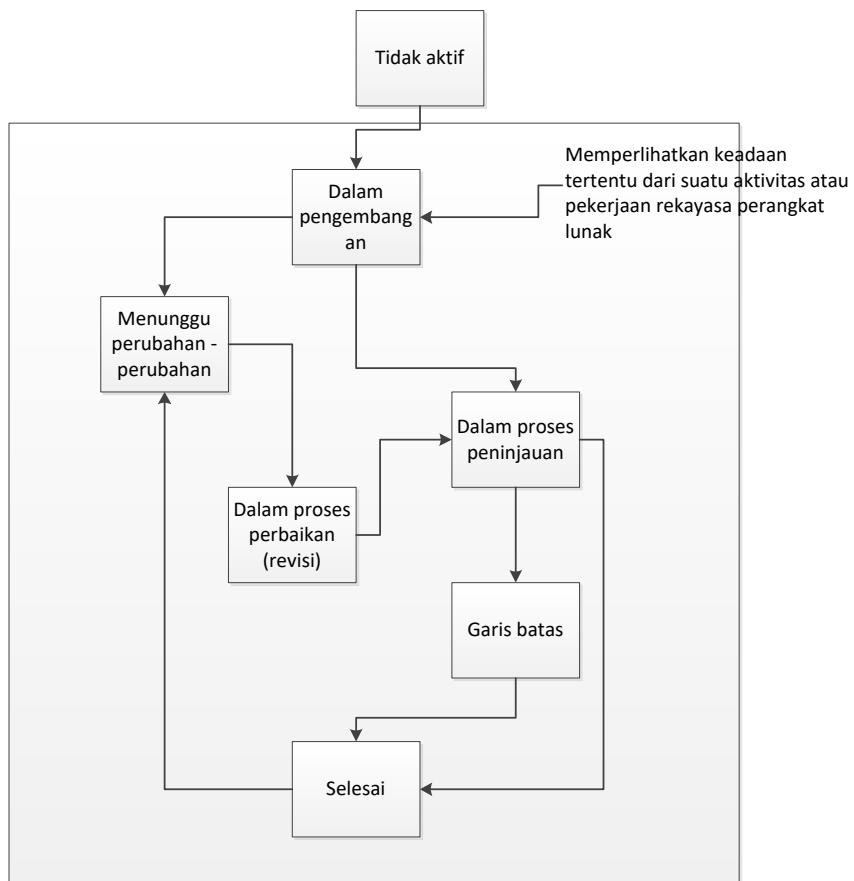
- a. Tingginya jumlah analisis risiko karenanya, menghindari kemungkinan risiko tentu berkurang.
- b. Model ini bagus untuk ukuran besar dan proyek penting.
- c. Dalam model spiral, fungsi tambahan dapat ditambahkan di kemudian hari.
- d. Ini lebih cocok untuk proyek-proyek berisiko tinggi, di mana kebutuhan bisnis dapat berbeda dari waktu ke waktu.

Kekurangan Model Spiral:

- a. Ini tentu model mahal untuk digunakan dalam hal pengembangan.
- b. Keberhasilan keseluruhan proyek tergantung pada fase analisis risiko sehingga kegagalan dalam fase ini dapat merusak keseluruhan proyek.
- c. Ini tidak sesuai untuk proyek berisiko rendah
- d. Risiko besar dari metodologi ini adalah bahwa ia dapat terus berlanjut tanpa batas dan tidak pernah selesai.

Model Konkuren

Model pengembangan perangkat lunak konkuren, memungkinkan tim perangkat lunak untuk menggunakan unsur – unsur yang bersifat berulang atau iteratif dan konkuren atau berjalan bersamaan dalam setiap model proses. Berikut gambaran aktivitas model konkuren (Roger S. Pressman, 2012).



Gambar 2.7 Aktivitas model proses konkuren

Model konkuren sering lebih tepat untuk proyek rekayasa sistem di mana tim teknik yang berbeda terlibat.

Model konkuren direpresentasikan dengan skema sebagai series dari kerangka aktivitas , aksi

software engineering dan juga tugas dari jadwal. Aktivitas dikerjakan secara bersamaan, setiap proses memiliki beberapa pemicu kerja dari aktivitas. Pemicu dapat berasal dari awal proses kerja maupun dari pemicu yang lain karena setiap pemicu akan saling berhubungan. Model ini digambarkan secara skematik dengan diagram modeling activity sebagai rangkaian dari teknis utama, tugas dan hubungan antar bagian. Model konkuren sering digunakan sebagai paradigma untuk pengembangan aplikasi *client/server*. Model konkuren memiliki kelebihan hasil sistem yang sangat baik karena terdapat perancangan yang besar , terencana dan matang. Sedangkan kekurangan dari model konkuren memungkinkan terjadinya perubahan besar – besaran yang akan membuat biaya dan waktu yang diperlukan membengkak. Jadi model konkuren ini suatu cara atau langkah kerja untuk membuat suatu sistem yang dikerjakan secara besar – besaran namun tetap mempertahankan kualitas sesuai dengan permintaan customer, bila ada permintaan lain dari customer maka langkah – langkah kerja dihentikan sementara untuk memaksimalkan hasil akhir dari model ini (IT, 2014).

Simpulan pada proses – proses evolusioner

Perhatian model – model evolusioner adalah mengembangkan perangkat lunak berkualitas

tinggi dalam arti interatif dan bersifat penambahan sedikit demi sedikit(inkremental). Meski demikian , merupakan hal yang mungkin untuk menggunakan proses yang evolusioner yang menenkankan pada fleksibilitas , perluasan serta kecepatan pengembangan. Tantangan bagi tim perangkat lunak dan para manajernya adalah menetapkan keseimbangan antara proyek yang bersifat kritis dan parameter – parameter produk dan kepuasan pelanggan (Roger S. Pressman, 2012).

4. Model Proses yang Terkhususkan

Model proses terkhususkan mengambil karakteristik – karakteristik dari satu atau lebih model – model yang tradisional yang telah dibahas sebelumnya.

Pengembangan Berbasis Komponen

Model ini biasa disebut CBD atau *component-based development*. Model ini kebanyakan menggunakan karakteristik model spiral. Model CBD mengkonstruksi perangkat lunak menggunakan komponen – komponen yang telah ada sebelumnya dengan cara merakitnya ke perangkat lunak yang sedang di kembangkan. Langkah – langkah pengembangan perangkat lunak dengan menggunakan pendekatan CBD adalah pertama komponen – komponen yang akan diintegrasikan diteliti dan diintegrasikan kemudian isu integrasi komponen dipertimbangkan, setelah dipertimbangkan

kemudian dirancang arsitektur perangkat lunak yang dapat mengakomodasi komponen, kemudian komponen tersebut diintegrasikan ke arsitektur dan dilakukan pengujian – pengujian yang bersifat komprehensif untuk melihat apakah komponen yang diintegrasikan memberikan fungsionalitas yang baik.

Model CBD ini memberikan peluang untuk penggunaan ulang komponen yang dapat menguntungkan rekayasa perangkat lunak (Roger S. Pressman, 2012).

5. Proses Terpadu (*Unified Proces*)

Ivar Jacobson , Graddy Booch dan James Rumbaugh dalam (Roger S. Pressman, 2012) membahas tentang kebutuhan untuk suatu proses perangkat lunak yang bersifat dikendalikan oleh *use case* , berpusat pada arsitektur , bersifat interatif menyatakan :

“saat ini perangkat lunak cenderung semakin besar dan kompleks, ini di sebabkan karena perangkat keras yang semakin *powerfull* sehingga para pengguna menjadi berharap banyak darinya. Kecendrungan ini juga dipengaruhi dengan internet yang digunakan sebagai budaya bertukar informasi. Dan para pengguna menginginkan perangkat lunak yang mampu beradaptasi lebih baik dengan kebutuhannya.”

Proses terpadu (*unified process*) berusaha menggunakan karakteristik – karakteristik terbaik model proses perangkat lunak

tradisional, tetapi menggabungkannya dengan prinsip – prinsip terbaik yang dimiliki pengembang perangkat lunak yang cepat. Proses terpadu mengenali pentingnya komunikasi dengan para pelanggan dan menekankan deskripsi sistem dari sudut pandang pelanggan (melalui diagram – diagram usecase).

Sejarah singkat

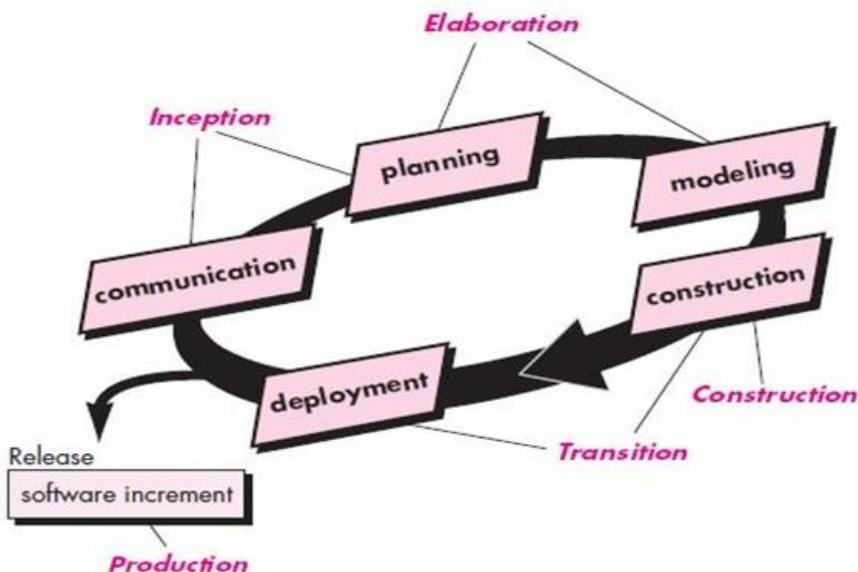
Pressman mengemukakan pada tahun 1990-an James Rumbaugh , Graddy Booch dan Ivar Jacobson mulai mengembangkan metode terpadu yang menggabungkan fitur terbaik dari masing – masing metode analisis dan perancangan berorientasi objek dan juga mengadopsi fitur – fitur tambahan yang diusulkan oleh para pakar dalam pemodelan berorientasi objek. Hasilnya adalah diagram – diagram UML (*Unified Modeling Language*), yang memuat notasi yang cukup tangguh untuk digunakan sebagai sarana pemodelan dan pengembangan perangkat lunak berorientasi objek (Roger S. Pressman, 2012).

UML menyediakan teknologi yang diperlukan untuk mendukung rekayasa perangkat lunak berorientasi objek , tetapi UML tidak menyediakan kerangka kerja proses untuk memandu tim – tim proyek dalam mengaplikasikan teknologi. Saat ini proses terpadu dan UML digunakan secara meluas pada proyek – proyek berorientasi objek yang beragam. Model yang bersifat interatif dan

inkremental diusulkan oleh UP dan sebaliknya diadaptasi untuk memperoleh kebutuhan – kebutuhan proyek perangkat lunak.

Tahapan – tahapan dalam proses terpadu

Tahapan dari proses terpadu adalah sebagai berikut (Roger S. Pressman, 2012) :



Gambar 2.8 Proses terpadu

Tahapan pengenalan (*inception*) dari proses terpadu membahas tentang komunikasi dengan para pelanggan dan juga membahas aktivitas – aktivitas perencanaan. Dengan cara berkolaborasi dengan para stakeholder, spesifikasi – spesifikasi bisnis untuk perangkat lunak dapat diidentifikasi, arsitektur garis besar

untuk sistem dapat diusulkan dan suatu rencana untuk tahapan – tahapan yang bersifat iteratif dan inkremental yang berkaitan dengan proyek mulai dikembangkan. Spesifikasi – spesifikasi bisnis yang mendasar dijelaskan melalui sejumlah use case awal yang deskripsikan fitur – fitur serta fungsi – fungsi yang mana untuk masing – masing kelas utama yang diinginkan oleh para pengguna. Arsitektur pada titik ini tidak lebih dari gambaran garis besar tentatif tentang subsitem – subsitem utama dan fungsi – fungsi serta fitur – fitur yang diperlukan untuk membentuk arsitektur sistem. Perencanaan sumber – sumber daya , melakukan penilaian terhadap risiko – risiko utama, mendefinisikan jadwal, serta menetapkan suatu dasar bagi tahapan – tahapan yang akan diaplikasikan saat pengembangan sedikit demi sedikit perangkat lunak (inkremental) dikembangkan.

Tahapan elaboration menggunakan aktivitas – aktivitas komuniasi dan pemodelan milik model proses generik. Tahap elaboration digunakan untuk menghaluskan dan mengembangkan use case awal yang dikembangkan pada tahap pengenalan dan mengembangkan representasi arsitektural dengan melibatkan 5 sudut pandang yang berbeda dari suatu perancangan perangkat lunak yaitu model use case, model spesifikasi kebutuhan, model perancangan , model

implementasi dan model penyebaran komponen (*deployment model*).

Tahapan construction pada metode proses terpadu identik dengan aktivitas yang sama yang didefinisikan untuk proses perangkat lunak generik. Pada tahapan konstruksi ini , tim pengembang perangkat lunak akan mengembangkan komponen – komponen perangkat lunak yang akan membuat masing – masing use case bersifat operasional untuk masing – masing pengguna akhir. Untuk dapat melakukan dengan baik, model – model spesifikasi kebutuhan dan perancangan yang diperoleh pada tahap elaboration dilengkapi untuk mencerminkan versi terakhir dari perangkat lunak yang pada dasarnya dikembangkan secara inkremental. Semua fitur – fitur dan fungsi – fungsi yang penting dan diperlukan untuk produk diimplementasi dalam bentuk kode – kode dalam bahasa pemrograman berorientasi objek tertentu yang dipilih. Setelah komponen – komponen diimplementasikan , unit – unit pengujian dirancang dan dieksekusi. Juga aktivitas – aktivitas yang berkaitan dengan pengintegrasian sistem dilaksanakan. Use case – use case akan digunakan untuk membentuk sejumlah paket pengujian yang akan dieksekusi sebelum semua masuk pada tahapan proses terpadu yang berikutnya.

Tahapan transition pada proses terpadu adalah penyerahan komponen dan umpan balik. Perangkat lunak diserahkan kepada pengguna akhir untuk pengujian beta dan untuk mendapatkan umpan balik dari pengguna tentang hal – hal yang berkaitan dengan cacat – cacat program dan perubahan – perubahan yang diperlukan.

Tahapan production pada tahapan ini perangkat lunak dipantau , dukungan untuk lengkungan operasional atau infrastruktur yang disediakan , dan laporan tentang cacat program dan permintaan untuk perubahan – perubahan dikirimkan dan dievaluasi.

Perlu dicatat bahwa tidak semua pekerjaan yang diidentifikasi untuk suatu aliran kerja proses terpadu harus dikerjakan pada setiap proyek perangkat lunak . Tim perangkat lunak akan mengadaptasi proses – proses sesuai kebutuhan proyek – proyek perangkat lunak secara spesifik.

2.2. Pengembangan Cepat

Rekayasa perangkat lunak cepat menggabungkan suatu falsafah dan serangkaian tuntunan pengembangan. Falsafah tersebut mengupayakan kepuasan pelanggan dan penghantaran perangkat lunak yang meningkat, tim – tim proyek yang kecil dan termotivasi, metode – metode informal, produk kerja rekayasa perangkat lunak yang minimal, dan keseluruhan kesederhanaan pengembangan. Panduan pengembangan

menekankan penghantaran daripada analisa dan perancangan , komunikasi yang berkelanjutan dan aktif antara pengembang dan pelanggan. Pengembangan cepat dapat memberikan manfaat , namun tidak selalu cocok untuk semua situasi (Roger S. Pressman, 2012).

Konteks kecepatan dalam rekayasa perangkat lunak , suatu tim cepat adalah suatu tim yang cekatan yang mampu merespon perubahan – perubahan dengan cepat dan tepat. Perubahan berarti perubahan dalam perangkat lunak yang dibuat, perubahan pada anggota tim, perubahan karena teknologi baru , perubahan semua hal yang mungkin berdampak pada produk perangkat lunak yang dikembangkan. Tim cepat tahu bahwa perangkat lunak yang dikembangkan oleh individu – individu yang bekerja dalam tim, keahlian, kolaborasi merupakan inti menuju keberhasilan proyek perangkat lunak. Kecepatan mendorong terbentuknya struktur tim dan mendorong tercapainya sikap yang menjadikan komunikasi menjadi lebih mudah dan kondusif. Kecepatan menekankan penghantaran yang cepat dari perangkat lunak, mengadopsi pelanggan sebagai salah satu tim pengembang. Proses cepat yang mencakup penghantaran perangkat lunak ke pengguna yang meningkat , serta dipasangkan dengan pengujian unit yang berkelajutan akan dapat menekan biaya pengembangan perangkat lunak. Proses cepat merupakan proses yang harus dapat beadaptasi, namun adaptasi berkelanjutan tanpa adanya kemajuan tidak dapat diharapkan. Untuk itu proses perangkat lunak yang cepat harus beradaptasi secara meningkat. Untuk dapat beradaptasi secara meningkat , suatu tim cepat sangat membutuhkan umpan balik yang datang dari pelanggan (Roger S. Pressman, 2012).

Ada 12 prinsip kecepatan dalam pengembangan perangkat lunak (Roger S. Pressman, 2012) :

1. Prioritas tertinggi adalah kepuasan pelanggan melalui penghantaran perangkat lunak dalam waktu yang lebih awal dan berkesinambungan.
2. Menyambut kebutuhan – kebutuhan yang berubah – ubah. Proses cepat memanfaatkan perubahan – perubahan untuk mencapai keunggulan kompetitif.
3. Sering mengahantarkan perangkat lunak
4. Pelaku bisnis dan pengembang perangkat lunak harus bekerja sama setiap hari selama proyek perangkat lunak dilaksanakan.
5. Bangun proyek perangkat lunak di lingkungan individu yang termotivasi.
6. Metode yang paling efisien dan efektif untuk menyampaikan informasi ke dan dalam tim pengembangan adalah percakapan tatap muka.
7. Perangkat lunak yang berjalan dengan baik adalah tolok ukur utama dari kemajuan suatu proyek perangkat lunak.
8. Proses pengembangan perangkat lunak cepat mendukung pengembangan berkelanjutan.
9. Peningkatan terus menerus mengenai keunggulan teknis dan perencanaan yang baik dapat meningkatkan kecepatan pengembangan perangkat lunak.
10. Kesederhanaan
11. Arsitektur, persyaratan dan rancangan perangkat lunak terbaik muncul dari tim perangkat lunak yang mengatur dirinya secara mandiri

12. Pada interval waktu yang teratur , tim perangkat lunak mencerminkan keadaan bagaimana caranya mereka menjadi lebih efektif , lalu menyesuaikan perilakunya.

Ada perdebatan besar tentang manfaat dan kecocokan pengembangan perangkat lunak cepat yang berlawanan dengan proses rekayasa perangkat lunak yang lebih konvensional. Ketika kecepatan dianggap penting bagaimana caranya membangun perangkat lunak yang memenuhi kebutuhan pelanggan hari ini dan menunjukkan karakteristik kualitas yang memungkinkan perangkat lunak untuk diperluas dan diukur untuk memenuhi kebutuhan pelanggan dalam jangka panjang ? tentunya tidak ada jawaban yang mutlak dari pertanyaan tersebut, namun banyak konsep cepat sebenarnya hanya diadaptasi dari konsep – konsep rekayasa perangkat lunak yang baik. Pada dasarnya ada banyak hal yang dapat diperoleh dengan mempertimbangkan hal – hal yang terbaik.

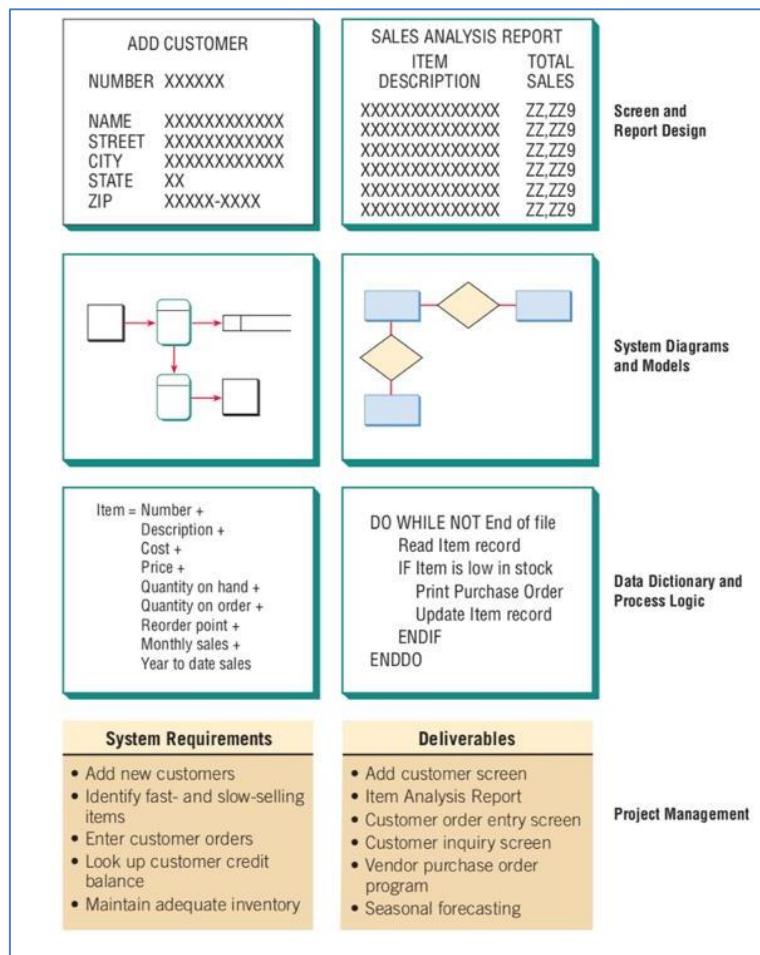
Pengembangan perangkat lunak secara cepat berfokus pada kemampuan individu, membentuk proses bagi tim. Untuk itu sejumlah ciri – ciri kunci harus ada diantara tim cepat yaitu (Roger S. Pressman, 2012):

1. Kompetensi , konteks kompetensi mencakup bakat , ketrampilan khusus yang berhubungan dengan perangkat lunak dan mencakup seluruh pengetahuan proses yang dipilih tim perangkat lunak untuk diterapkan.
2. Fokus bersama, meskipun anggota tim memiliki kecerdasan yang dapat melakukan tugas yang berbeda – beda , semuanya pada dasarnya harus dipusatkan pada satu tujuan yaitu untuk meningkatkan kualitas

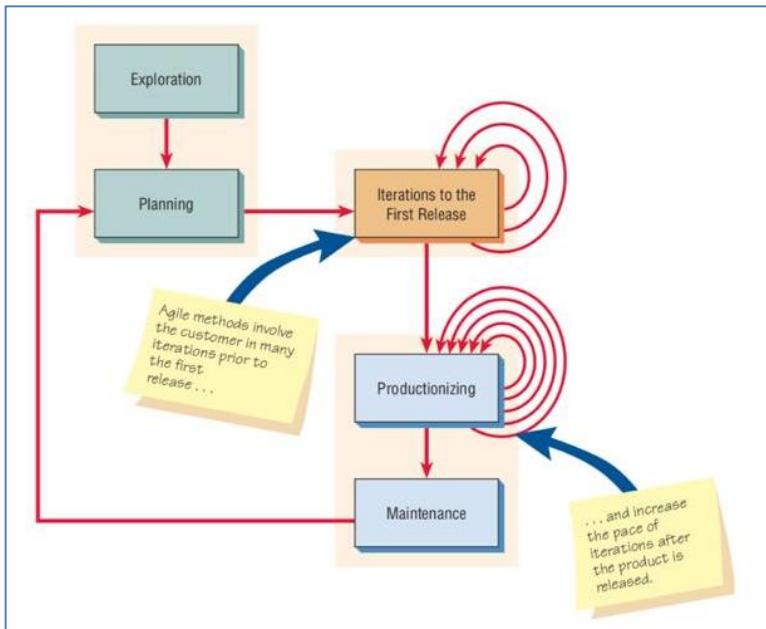
- perangkat lunak yang akan dihantarkan kepada pelanggan dalam jangka waktu yang dijanjikan. Untuk mencapai tujuan ini, tim cepat akan berfokus pada adaptasi – adaptasi yang berkelanjutan yang akan menyesuaikan proses perangkat lunak dengan kebutuhan tim perangkat lunak.
3. Kolaborasi, rekayasa perangkat lunak adalah tentang penilaian, analisis, dan penggunaan informasi yang dikomunikasikan kepada tim dan dia antara anggota – anggotan tim perangkat luna, tentang pembuatan informasi yang membantu semua stakeholder untuk memahami kerja tim, dan tentang bagaimana cara membangun atau mengembangkan informasi – informasi yang memberikan nilai bisnis tertentu bagi para pelanggan . Untuk mencapai tugas – tugas ini , anggota tim cepat harus saling bekerjasama dengan stakeholder lainnya.
 4. Kemampuan pengambilan keputusan. Setiap tim perangkat lunak yang baik harus memungkinkan memiliki kebebasan mengendalikan nasibnya sendiri. Hal ini berarti bahwa tim perangkat lunak harus diberikan otonomi secukupnya maksudnya otoritas pengambilan keputusan untuk isu – isu proyek maupun isu – isu yang terkait dengan masalah teknis.
 5. Kemampuan pemecahan masalah yang kabur. Manajer perangkat lunak menyadari bahwa tim cepat akan tetap harus berhadapan dengan ambiguitas dan akan diterpa perubahan – perubahan yang harus dihadapinya, namun demikian, pelajaran yang di petik dari aktivitas pemecahan masalah mungkin bermanfaat bagi tim

- perangkat lunak saat nantinya mereka melakukan tugas – tugas penting dalam proyek.
6. Saling percaya dan menghormati . Tim cepat harus menjadi tim kental yaitu menunjukkan kepercayaan dan penghargaan yang diperlukan untuk menjadikan mereka begitu kuat dan rapat sehingga keseluruhan merupakan bagian yang lebih besar daripada kumpulan bagian – bagian kecil.
 7. Pengorganisasian diri. Menyiratkan tiga hal yaitu tim cepat mampu mengorganisasikan dirinya sendiri untuk pekerjaan yang akan diselesaikan, tim cepat mampu mengatur proses perangkat lunak untuk dapat mengakomodasikan dengan sangat baik lingkungan lokalnya, tim cepat mampu mengatur jadwal kerja untuk mencapai penghantaran perangkat lunak ke para pelanggan dengan cara yang sangat baik.

Menurut (Kenneth E. Kendall, 2010) berikut adalah *repository concept* dalam pengembangan cepat :



Gambar 2.9 *Repository concept pengembangan cepat*



Gambar 2.10 Tahapan dalam model proses pengembangan cepat

2.2.1. Pemograman Extreme (XP)

Pemograman Extrem(*Extreme Programming*) yaitu suatu pendekatan yang paling banyak digunakan untuk pengembangan perangkat lunak cepat. Nilai – nilai yang membentuk dasar dari XP adalah komunikasi, kesederhanaan, umpan balik , keberanian dan rasa hormat.

Untuk mencapai *komuniiasi* yang efektif yang seharusnya terjadi antara pengembang dan para stakeholder , XP menekankan kolaborasi informal antara pelanggan dan pengembang perangkat lunak , menekankan pentingnya pembentukan metafora – metafora yang efektif untuk mengkomunikasikan konsep – konsep yang penting, menekankan pentingnya adaptasi terhadap umpan balik yang berkesinambungan , dan menekankan pentingnya dokumentasi yang produktif sebagai se suatu media komunikasi. Untuk mencapai kesederhanaan, XP membatasi pengembang perangkat lunak melakukan perancangan hanya untuk kebutuhan – kebutuhan yang sifatnya mendesak, tujuannya adalah untuk menciptakan rancangan yang sederhana yang dapat dengan mudah diimplementasikan dalam bentuk kode – kode program yang sangat cepat, jika rancangan tersebut harus ditingkatkan , maka rancangan tersebut dapat di *refactorisasi* di waktu lain.

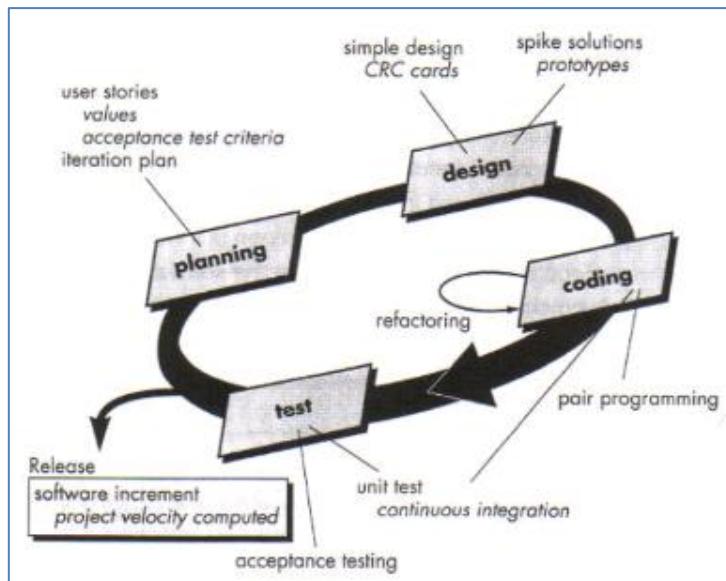
Umpan balik pada umumnya berasal dari tiga sumber yaitu dari perangkat lunak yang diimplementasikan sendiri , dari pelanggan dan dari anggota tim perangkat lunak.

Kepatuhan ketat dengan praktik – praktik XP tertentu keberanian. Sebuah tim XP yang cepat harus memiliki disiplin untuk melakukan perancangan hanya untuk saat ini dan m negakui kebutuhan di masa depan berubah secara drastis, sehingga menuntut pengrajan ulang substansial atas rancangan – rancangan yang telah

dibuat serta menuntut pengeringan ulang atas kode – kode program komputer yang telah diimplementasikan.

Dengan mengikuti masing – masing nilai penting tersebut, tim cepat menanamkan rasa hormat diantara para anggota tim lainnya dan anggota tim perangkat lunak secara tidak langsung menanamkan rasa hormat untuk perangkat lunak itu sendiri.

Proses XP



Gambar 2.11 Proses Pemrograman Ekstrem

XP menggunakan suatu pendekatan berorientasi objek sebagai paradigma pengembangan yang diinginkan dan mencakup di dalamnya seperangkat aturan praktik –

praktik yang terjadi didalam konteks empat kerangka kerja yaitu (Roger S. Pressman, 2012) :

1. **Perencanaan** , kegiatan perencanaan dimulai dengan mendengarkan yaitu suatu kegiatan yang bertujuan untuk mengumpulkan kebutuhan – kebutuhan yang memungkinkan anggota tim teknis XP memahami konteks bisnis untuk perangkat lunak yang akan dikembangkan sehingga mendapatkan inspirasi terkait output , fitur dan fungsionalitas dari perangkat yang akan dikembangkan. Pelanggan memberikan suatu nilai yaitu seluruh nilai bisnis dari fitur atau fungsi. Anggota tim menilai dari apa yang telah di uraikan pelanggan, dari uraian tersebut anggota tim menetapkan biaya yang diukur dari kesulitan serta lamanya proyek pengembangan perangkat lunak. Ketika terbentuk komitmen dasar antara pelanggan dengan pengembang perangkat lunak, maka anggota tim pengembang perangkat lunak akan membuat jadwal pengembangan perangkat lunak , sehingga pelanggan dapat mengetahui perkiraan perangkat lunak tersebut siap untuk dipakai.
2. **Perancangan** , perancangan XP mengikuti prinsip “tetap sederhana”. Pada pengembangan menggunakan pendekatan XP , digunakan kartu CRC (*Class-responsibility-collaborator*), kartu ini digunakan untuk mengidentifikasi dan mengatur kelas – kelas dalam konteks pemrograman berorientasi objek. Jika pada

identifikasi kebutuhan, kesulitan dalam menentukan masalah perancangan , maka disarankan untuk membuat prototipe operasional dari perancangan tersebut. Prototipe ini selanjutnya diimplementasikan dan dievaluasi oleh pelanggan sehingga pelanggan dapat menvalidasi dari fungsionalitas serta fitur – fitur perangkat lunak yang di kembangkan. Metode XP ini mendorong refaktorisasi yaitu perancangan perangkat lunak yang terjadi terus menerus ketika sistem di konstruksi, kegiatan ini memberikan tim pedoman bagaimana cara untuk meningkatkan kualitas rancangan.

3. **Pengkodean**, setelah cerita pelanggan di kembangkan menjadi sebuah perancangan , langkah selanjutnya adalah mengembangkan rangkaian pengujian , kemudian beralih ke kode –kode program. Kode – kode program tersebut selanjutnya diimplementasikan. Dari implementasi tersebut akan memberikan umpan balik kepada pengembang. Pada pendekatan ini biasanya diterapkan konsep pemogram berpasangan yaitu satu komputer workstation digunakan untuk menuliskan kode - kode oleh dua orang , sehingga mereka dapat bekerjasama dalam menyelesaikan suatu masalah.
4. **Pengujian**, uji kelayakan XP sering disebut uji pelanggan, dirinci oleh pelanggan dan pada dasarnya berfokus pada fitur – fitur dan fungsionalitas – fungsionalitas sistem perangkat

lunak secara keseluruhan yang dapat terlihat dan ditinjau kembali oleh pelanggan.

Berikut adalah kelebihan dan kekurangan pendekatan XP (tatvasoft, 2015).

Keuntungan Metodologi Pemrograman Ekstrim :

1. Metodologi pemrograman ekstrem menekankan pada keterlibatan pelanggan.
2. Model ini membantu untuk menetapkan rencana dan jadwal yang rasional dan untuk membuat para pengembang secara pribadi berkomitmen untuk jadwal mereka yang tentunya merupakan keuntungan besar dalam model XP.
3. Model ini konsisten dengan kebanyakan metode pengembangan modern sehingga, pengembang dapat menghasilkan perangkat lunak berkualitas

Kekurangan Metodologi Pemrograman Ekstrim :

1. Model pengembangan perangkat lunak semacam ini membutuhkan pertemuan pada interval yang sering dengan biaya besar kepada pelanggan
2. Ini membutuhkan terlalu banyak perubahan pengembangan yang sangat sulit untuk diadopsi setiap waktu bagi pengembang perangkat lunak
3. Dalam metodologi ini, cenderung mustahil untuk diketahui perkiraan pasti dari upaya kerja yang diperlukan untuk memberikan penawaran, karena pada awal proyek tidak ada yang tahu

tentang seluruh ruang lingkup dan persyaratan proyek

XP Industrial (IXP)

IXP merupakan evolusi organik dari XP. IXP diilhami dengan semangat minimaslis, berpusat pada pelanggan dan yang diarahkan oleh pengujian. IXP didalamnya mencakup enam praktik baru yang dirancang untuk memastikan proyek XP berkerja dengan sukses untuk proyek – proyek perangkat lunak yang signifikan yang ada dalam organisasi – organisasi yang besar . berikut enam praktik dalam IXP (Roger S. Pressman, 2012):

1. Penilaian kesiapan, sebelum memulai proyek IXP, organisasi pada awalnya melakukan penilaian kesiapan yang bertujuan untuk memastikan lingkungan pengembangan perangkat lunak tepat sudah mendukung IXP, tim perangkat lunak diisi dengan stakeholder yang tepat, organisasi perangkat lunak memiliki program yang berkualitas dan mendukung peningkatannya, budaya organisasi perangkat lunak akan mendukung nilai – nilai baru yang cepat, proyek masyarakat yang lebih luas akan diisi dengan tepat.
2. Komunitas proyek, ketika XP harus diterapkan konsep tim menjadi sebuah komunitas, dalam IXP anggota komunitas perangkat lunak akan ditetapkan perannya dan mekanisme untuk komunikasi dan koordinasi antar anggota komunitas perangkat lunak yang akan dibuat.

3. Pelabelan proyek, pelabelan menguji konteks proyek untuk menentukan bagaimana konteks tersebut melengkapi , memperluas atau menggantikan sistem atau proses yang telah ada sebelumnya.
4. Pengaturan yang diarahkan oleh pengujian, pengaturan yang diarahkan melalui mekanisme pengujian akan menetapkan serangkaian tujuan yang terukur dan kemudian mendefinisikan mekanisme untuk menentukan apakah tujuan – tujuan tersebut telah tercapai atau belum.
5. Retrospektif , yaitu proses penelaahan teknis khusus setelah perangkat lunak dihantarkan ke pelanggan. Penelaahan ini merupakan proses peninjauan isu – isu dari sebuah peningkatan perangkat lunak yang bertujuan untuk meningkatkan proses IXP.
6. Pembelajaran yang berkelanjutan, anggota tim IXP di dorong untuk mempelajari metode – metode dan teknik – teknik yang baru yang dapat menghasilkan produk perangkat lunak yang berkualitas tinggi.

Selain enam praktik diatas , IXP juga melakukan modifikasi beberapa praktik yaitu (Roger S. Pressman, 2012):

- I. Pengembangan yang didorong cerita (*story-driven development – SDD*)
2. Perancangan yang didorong oleh ranah (*domain –driven development- DDD*)

3. Pemasangan (*pairing*)

Meskipun XP merupakan sebuah pendekatan yang menghasilkan suatu produk yang cepat dihantarkan ke pelanggan , namun beberapa ahli memperdebatkan beberapa isu yaitu diantaranya (Roger S. Pressman, 2012) :

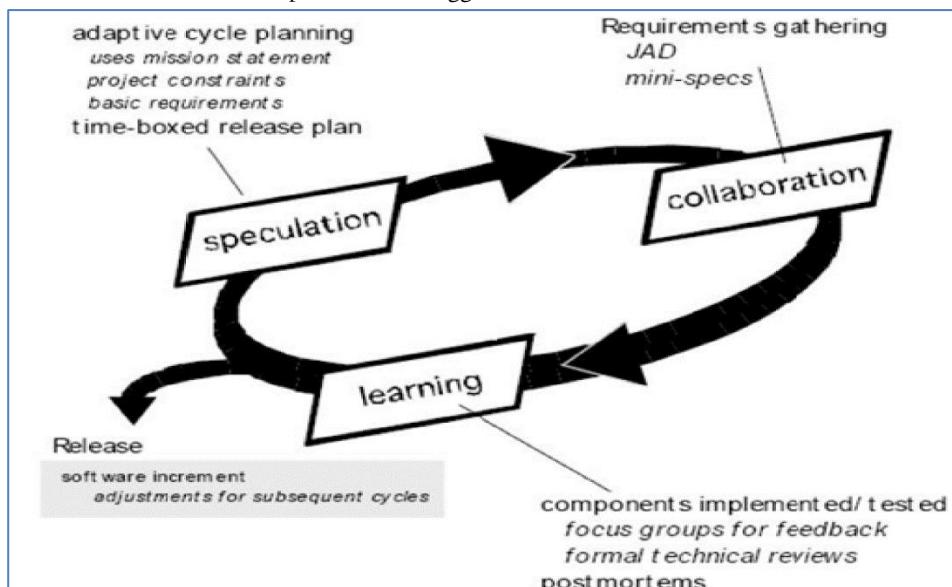
1. Volatilitas kebutuhan , dalam XP pelanggan masuk ke dalam tim sehingga , perubahan kebutuhan diminta secara informal.
2. Kebutuhan pelanggan yang bertentangan , dalam proyek perangkat lunak sering ditemukan banyak pelanggan yang memiliki berbeda – beda kebutuhan .
3. Kebutuhan dinyatakan secara informal.
4. Kurangnya perancangan formal.

2.2.2. Model Proses Cepat Lainnya

Berikut adalah beberapa model proses cepat lainnya :

1. Pengembangan perangkat lunak adaptif (*Adaptive Software Development – ASD*)
Fokus dari ASD adalah kolaborasi manusia. Siklus hidup ASD menggabungkan tiga fase yaitu spekulasi, kolaborasi dan pembelajaran. Spekulasi proyek dimulai dan perencanaan siklus adaptif dilakukan . siklus perencanaan adaptif menggunakan informasi inisiasi proyek yaitu pernyataan misi pelanggan, kendala proyek dan kebutuhan dasar yang akan dibutuhkan untuk proyek tersebut. Orang – orang yang

termotivasi akan melakukan kolaborasi sehingga dapat meningkatkan kinerja. Kolaborasi ini meliputi komunikasi, kerjasama , menekan individualisme sehingga akan menciptakan kepercayaan antar anggota tim. Penekanan dari pengembangan komponen siklus adaptif adalah pembelajaran , pembelajaran akan membantu pemahaman anggota tim.

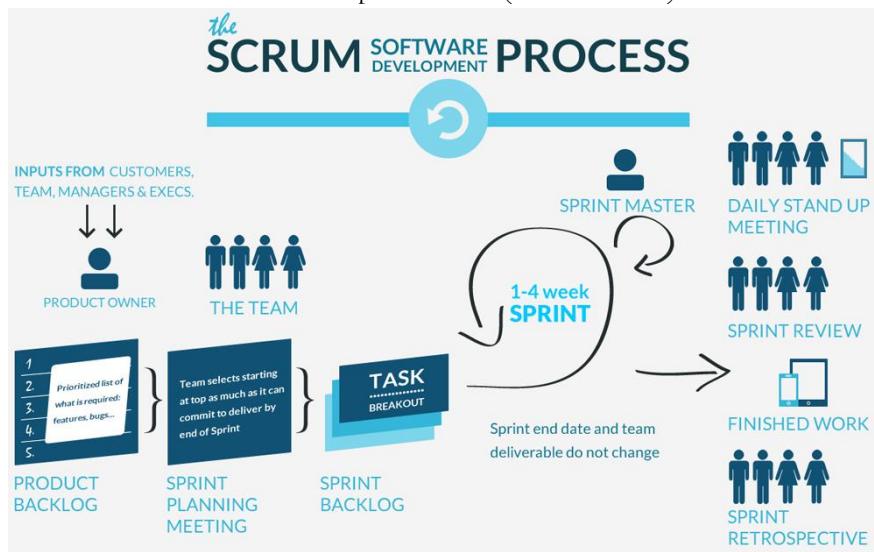


Gambar 2.12 Pengembangan Perangkat Lunak Adaptif

2. Scrum

Scrum merupakan suatu metode pengembangan perangkat lunak yang proses pengembangannya

mencakup kegiatan kerangka kerja berikut : kebutuhan , analisis , perancangan , evolusi dan penghantaran. Dalam masing – masing kegiatan kerangka kerja , tugas kerja terjadi dalam suatu pola proses yang disebut sebagai sprint. Kerja yang dilakukan dalam suatu sprint disesuaikan dengan masalah yang dihadapi dan didefinisikan dan seiring dirubah secara *real time* oleh tim scrum (Roger S. Pressman, 2012). Berikut adalah aliran proses scrum (Maxxor, 2018):



Gambar 2.13 Aliran Proses Scrum

Sumber : (Maxxor, 2018)

Backlog merupakan daftar prioritas dari kebutuhan proyek atau fitur yang menyediakan nilai bisnis bagi pelanggan. Item – item dapat

ditambahkan pada backlog setiap saat. Para manajer produk menilai backlog dan prioritas update sesuai kebutuhan. Sprint terdiri atas unit kerja yang diperlukan untuk mencapai kebutuhan yang didefinisikan dalam proses backlog yang harus sesuai dengan *time-box* yang sudah didefinisikan sebelumnya, sprint memungkinkan anggota tim untuk bekerja dalam lingkungan jangka pendek , namun stabil. Scrum meetings biasanya pendek waktunya, ada tiga pertanyaan yang diajukan dan dijawab anggota tim dalam meeting tersebut yaitu apa yang anda lakukan sejak pertemuan terakhir tim ? kendala apa saja yang anda hadapi ? apa yang anda rencanakan untuk mencapai sesuatu sebelum rapat tim berikutnya ?. Seorang pemimpin tim disebut scrum master , akan memimpin rapat dan menilai tanggapan yang muncul. Scrum meeting membantu tim perangkat lunak untuk mengungkap potensi terjadinya masalah – masalah sedini mungkin. Demo memberikan peningkatan perangkat lunak untuk pelanggan sehingga fungsionalitas yang telah dilaksanakan dapat didemonstrasikan dan dievaluasi oleh pelanggan (Roger S. Pressman, 2012).

Berikut adalah kelebihan dan kekurangan pendekatan scrum (tatvasoft, 2015).

Keuntungan Pengembangan Scrum:

- a. Dalam metodologi ini, pengambilan keputusan sepenuhnya berada di tangan tim
- b. Metodologi ini memungkinkan proyek di mana dokumentasi persyaratan bisnis tidak dianggap sangat signifikan untuk pengembangan yang sukses
- c. Ini adalah metode yang dikontrol ringan yang benar-benar berempati pada pembaruan sering dari kemajuan, oleh karena itu, langkah-langkah pengembangan proyek terlihat dalam metode ini
- d. Pertemuan harian dengan mudah membantu pengembang untuk memungkinkan mengukur produktivitas individu. Ini mengarah pada peningkatan produktivitas masing-masing anggota tim

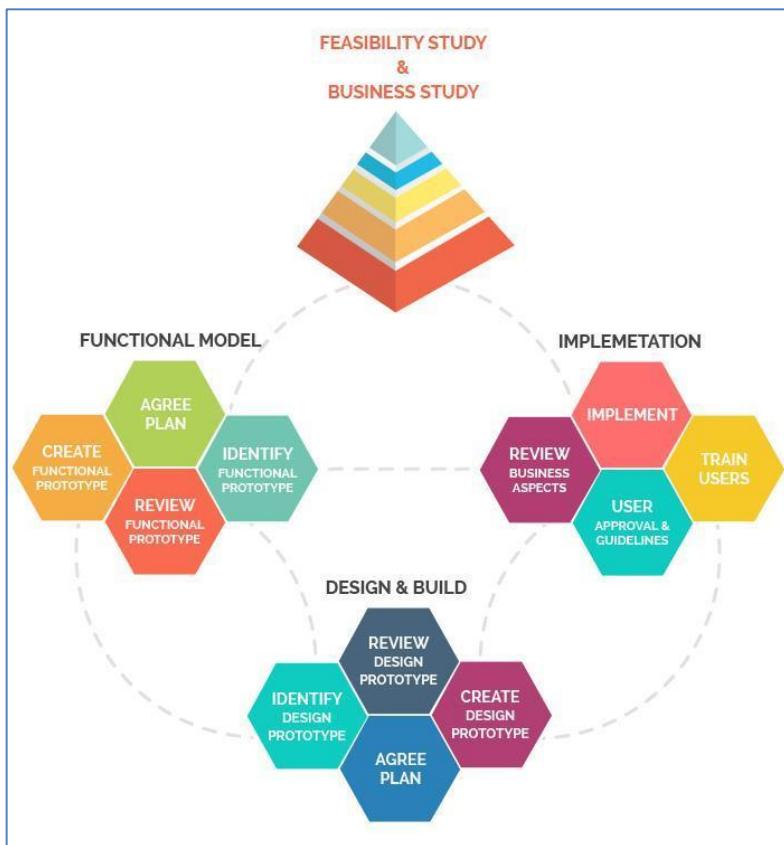
Kekurangan Pengembangan Scrum:

- a. Model pengembangan semacam ini menyulitkan jika perkiraan biaya dan waktu proyek tidak akan akurat.
- b. Ini bagus untuk proyek kecil yang bergerak cepat tetapi tidak cocok untuk proyek ukuran besar.
- c. Metodologi ini membutuhkan anggota tim yang berpengalaman saja. Jika tim terdiri dari orang-orang yang pemula,

proyek tidak dapat diselesaikan dalam jangka waktu yang tepat.

3. Metode Pengembangan Sistem Dinamik(*Dynamic System Development Methode - DSDM*)

Merupakan sebuah pendekatan pengembangan perangkat lunak cepat yang menyediakan kerangka kerja untuk membangun dan memelihara sistem yang memenuhi batas waktu yang ketat melalui penggunaan protipe yang secara perlahan ditambahi dan diperluas dalam lingkungan proyek terkendali. Siklus hidup dari DSDM ini meliputi :



Gambar 2.14 *Dynamic System Development Methode Process*

Sumber : (AZM Ehtesham Chowdhury, 2017)

- Studi Kelayakan
Menetapkan persyaratan bisnis dasar dan kendala yang terkait dengan aplikasi yang akan dibangun atau dikembangkan dan

- kemudian menilai apakah aplikasi adalah calon yang layak untuk proses DSDM.
- b. Studi Bisnis
Menentapkan persyaratan fungsionalitas dan informasi yang akan memungkinkan aplikasi untuk memberikan nilai bisnis dan mendefinisikan pula arsitektur aplikasi dasar dan mengidentifikasi kebutuhan pemeliharaan untuk aplikasi.
 - c. Iterasi Model Fungsional
Menghasilkan satu set prototipe yang secara perlahan ditambahkan dan diperluas yang menujukkan fungsionalitas untuk pelanggan, tujuan dari siklus iterasi yaitu mengumpulkan kebutuhan tambahan dengan melancarkan umpan balik dari pengguna saat mereka Menjalankan Prototipe.
 - d. Perancangan Dan Membangun Iterasi
Mengunjungi kembali prototipe yang dibangun selama iterasi model fungsional untuk memastikan masing – masing telah direkayasa sedemikian rupa sehingga memungkinkannya memberikan nilai bisnis operasional bagi pengguna akhir.
 - e. Implementasi
Menempatkan peningkatan petangkat lunak baru ke lingkungan operasional.

Berikut adalah kelebihan dan kekurangan dari pendekatan DSDM (tatvasoft, 2015).

Keuntungan dari Model Pengembangan Sistem Dinamis:

- a. Pengguna sangat terlibat dalam pengembangan sistem sehingga, mereka lebih mungkin untuk mendapatkan pegangan pada proyek pengembangan perangkat lunak.
- b. Dalam model ini, fungsi dasar disampaikan dengan cepat, dengan lebih banyak fungsi yang dikirimkan pada interval yang sering.
- c. Metode ini memberikan akses yang mudah oleh pengembang kepada pengguna akhir.
- d. Dalam pengembangan semacam ini, pendekatan proyek disampaikan tepat waktu dan sesuai anggaran tertentu.

Kekurangan Model Pengembangan Sistem Dinamis:

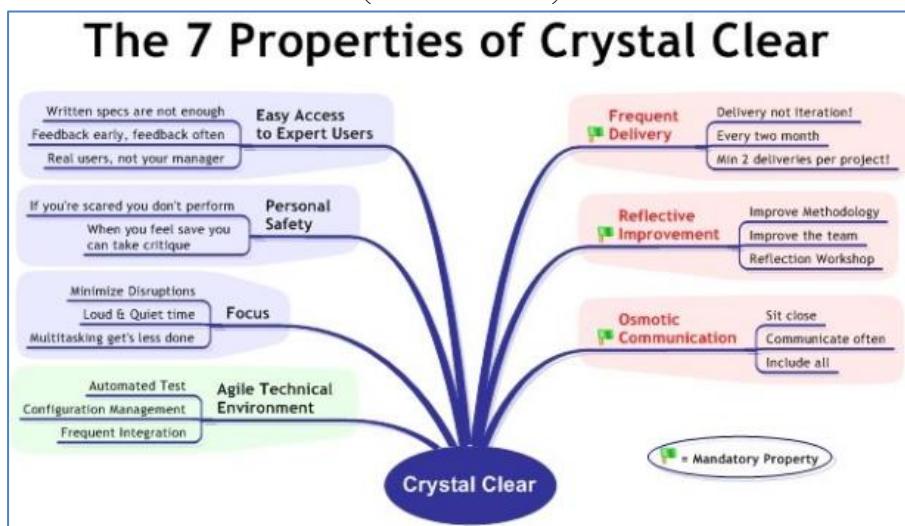
- a. Hal pertama adalah DSDM sangat mahal untuk diterapkan, karena membutuhkan pengguna dan pengembang untuk dilatih untuk menggunakannya secara efektif. Ini mungkin tidak cocok untuk organisasi kecil atau proyek satu kali
- b. Ini adalah model yang relatif baru, oleh karena itu, tidak terlalu umum dan mudah dimengerti



Crystal

Metode Crystal adalah keluarga metodologi pengembangan perangkat lunak yang

dikembangkan oleh Alistair Cockburn dari studinya dan wawancara tim. Metode ini diberi kode warna untuk menandakan risiko terhadap kehidupan manusia. Misalnya, proyek yang mungkin melibatkan risiko terhadap kehidupan manusia akan menggunakan Crystal Sapphire sementara proyek yang tidak memiliki risiko seperti itu akan menggunakan Crystal Clear. Crystal berfokus pada enam aspek utama: orang, interaksi, komunitas, komunikasi, keterampilan, dan bakat (SANTOS, 2017).



Gambar 2.15 The 7 Properties of Crystal Clear
Sumber : (SANTOS, 2017)

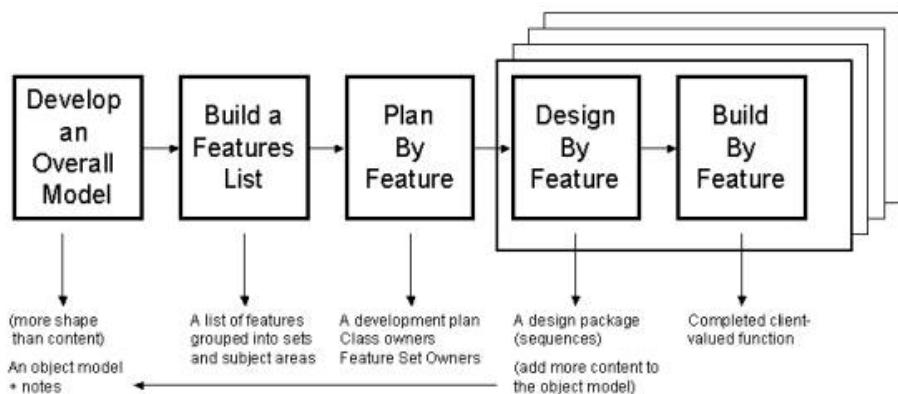
Metode Crystal menganggap orang sebagai yang paling penting, jadi proses harus dimodelkan untuk memenuhi persyaratan tim. Ini adaptif,

tanpa seperangkat alat dan teknik yang ditentukan. Ini juga ringan, tanpa terlalu banyak dokumentasi, manajemen atau pelaporan. Bobot metodologi ditentukan oleh lingkungan proyek dan ukuran tim.

4. Pengembangan Dikendalikan Fitur (*Feature Driven Development – FDD*)

Feature-Driven Development (FDD) diperkenalkan pada tahun 1997 oleh Jeff De Luca ketika ia bekerja dalam proyek pengembangan perangkat lunak untuk sebuah bank Singapura yang besar. FDD juga dibangun di sekitar praktik terbaik rekayasa perangkat lunak seperti pemodelan objek domain, yang dikembangkan oleh fitur dan kepemilikan kode. Perpaduan dari praktik-praktik ini yang menghasilkan keseluruhan kohesif adalah karakteristik terbaik dari FDD. Ini terdiri dari lima kegiatan dasar, yaitu pengembangan model keseluruhan, pembangunan daftar fitur, perencanaan berdasarkan fitur, perancangan berdasarkan fitur, dan pembangunan berdasarkan fitur. Setiap proyek akan memiliki model uniknya sendiri, yang akan menghasilkan daftar fitur. Tiga aktivitas terakhir adalah proses berulang yang singkat, dengan fitur yang tidak membutuhkan waktu lebih lama dari dua minggu untuk dibangun. Jika akan memakan waktu lebih dari dua minggu, maka harus

dipecah menjadi fitur yang lebih kecil (SANTOS, 2017).

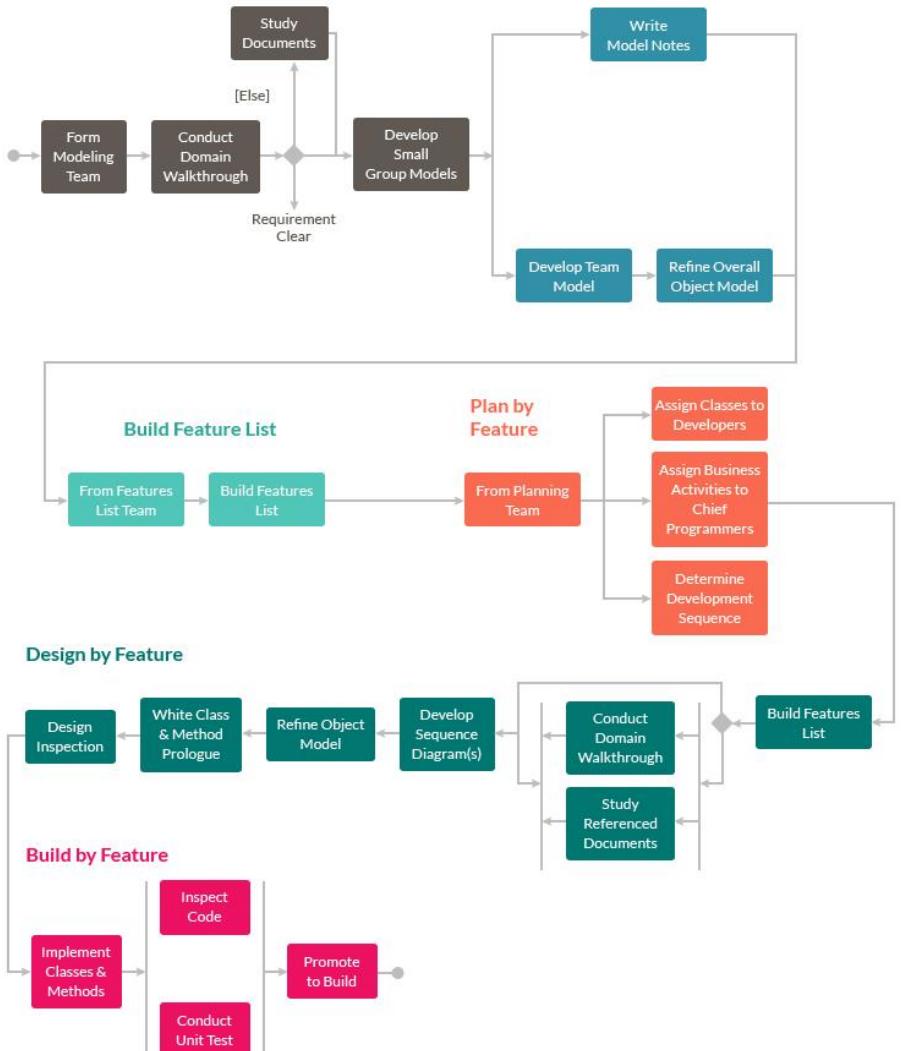


Gambar 2.16 *Feature Driven Development Process*

Sumber : (SANTOS, 2017)

Tujuan utama FDD adalah menyampaikan perangkat lunak yang nyata dan bekerja secara berulang pada waktu yang tepat. Keuntungan menggunakan FDD adalah *scalable* bahkan untuk tim besar karena konsep ‘just enough design first’ (JEDI). Ini adalah solusi yang bagus untuk mempertahankan kendali atas proyek yang lincah, inkremental dan inheren kompleks karena prosesnya yang berpusat pada fitur. Ini digunakan dalam proyek-proyek perusahaan serta proyek web seperti situs game online Mousebraker.com (SANTOS, 2017).

Develop Overall Model



Gambar 2.17 Detail process FDD

Sumber : (tatvasoft, 2015)

Berikut keuntungan dan kelebihan dari pendekatan FDD (tatvasoft, 2015) :

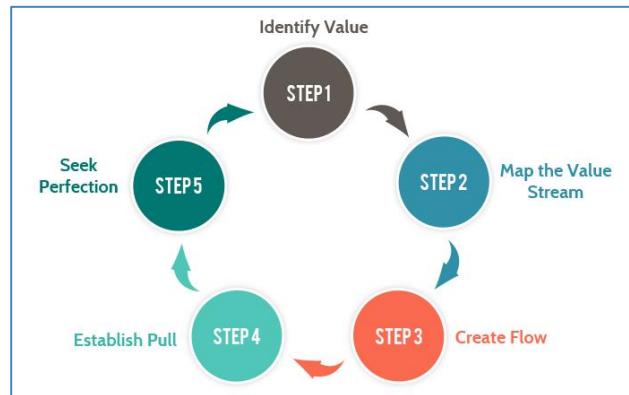
Keuntungan Metodologi FDD:

- a. FDD Membantu memindahkan proyek ukuran yang lebih besar dan memperoleh keberhasilan yang berulang.
- b. Lima proses sederhana membantu menyelesaikan pekerjaan dalam waktu singkat dan cara termudah.
- c. Jenis model ini dibangun di atas standar yang ditetapkan untuk industri pengembangan perangkat lunak, sehingga membantu pengembangan yang mudah dan praktik terbaik yang diakui industri

Kekurangan Metodologi FDD:

- a. Bukan metodologi yang ideal untuk proyek yang lebih kecil, jadi tidak baik untuk pengembang perorangan.
- b. Ketergantungan yang tinggi pada pengembang utama berarti orang tersebut harus dilengkapi sepenuhnya untuk bertindak sebagai koordinator, desainer utama, dan mentor.

- c. Tidak ada dokumentasi tertulis yang diberikan kepada klien dalam metodologi ini, sehingga mereka tidak dapat memperoleh bukti untuk perangkat lunak mereka sendiri
5. Pengembangan Perangkat Lunak yang Ringkas (*Lean Software Development – LSD*) berfokus pada penciptaan perangkat lunak yang mudah berubah. Model Pengembangan Perangkat Lunak ini lebih strategis daripada berbagai jenis metodologi tangkas lainnya. Tujuan dari metodologi ini adalah untuk mengembangkan perangkat lunak dalam sepertiga waktu, dengan anggaran yang sangat terbatas, dan jumlah alur kerja yang sangat sedikit (tatvasoft, 2015).



Gambar 2.18 Proses *lean software development*.

Sumber : (tatvasoft, 2015)

Prinsip – prinsip ringkas yang diilhami LSD dapat diringkas menjadi menghilangkan pemborosan, membangun kualitas, menciptakan pengetahuan, menunda komitmen, cepat menghantarkan , menghargai orang dan mengoptimalkan keseluruhan (Roger S. Pressman, 2012).

Berikut keuntungan dan kelebihan dari pendekatan *Lean* (tatvasoft, 2015).

Keuntungan Metodologi Pengembangan Lean :

- a. Penghapusan awal efisiensi keseluruhan proses pengembangan tentu membantu mempercepat proses pengembangan perangkat lunak keseluruhan yang pasti mengurangi biaya proyek
- b. Menyampaikan produk lebih awal adalah keuntungan yang pasti. Ini berarti bahwa tim pengembangan dapat memberikan lebih banyak fungsi dalam waktu yang lebih singkat, sehingga memungkinkan lebih banyak proyek yang akan dikirimkan
- c. Pemberdayaan tim pengembangan membantu dalam mengembangkan kemampuan pengambilan keputusan dari anggota tim yang menciptakan motivasi lebih di antara anggota tim

Kekurangan Metodologi Pengembangan Lean :

- a. Keberhasilan dalam pengembangan perangkat lunak tergantung pada bagaimana disiplin para anggota tim dan bagaimana memajukan keterampilan teknis mereka.
 - b. Peran seorang analis bisnis sangat penting untuk memastikan dokumentasi persyaratan bisnis dipahami dengan benar. Jika ada organisasi yang tidak memiliki orang dengan analis bisnis yang tepat maka metode ini mungkin tidak berguna bagi mereka.
 - c. Dalam model pengembangan ini, fleksibilitas yang besar diberikan kepada pengembang yang pasti hebat, tetapi terlalu banyak dari itu akan dengan cepat mengarah ke tim pengembangan yang kehilangan fokus pada tujuan awalnya sehingga, hati aliran seluruh pekerjaan pengembangan proyek.
6. Pemodelan Cepat (*Agile Methode*)
Scott Ambler dalam (Roger S. Pressman, 2012) mengemukakan bahwa pemodelan cepat adalah metodologi berbasis praktik untuk pemodelan dan dokumentasi berbasis perangkat lunak yang efektif, yang di dalamnya terdapat kumpulan dari nilai – nilai , prinsip dan praktik untuk model perangkat lunak yang dapat diterapkan

pada proyek pengembangan perangkat lunak secara efektif dan secara ringan.

Berikut adalah kelabihan dan kekurangan dari pendekatan *agile* (tatvasoft, 2015)

Keuntungan Metodologi Pengembangan Agile:

- a. Metodologi tangkas memiliki pendekatan adaptif yang mampu menanggapi perubahan persyaratan dari klien
- b. Komunikasi langsung dan umpan balik konstan dari perwakilan pelanggan tidak menyisakan ruang untuk dugaan apa pun dalam sistem

Kekurangan Metodologi Pengembangan Agile:

- a. Metodologi ini berfokus pada perangkat lunak yang bekerja daripada dokumentasi, sehingga dapat mengakibatkan kurangnya dokumentasi
 - b. Proyek pengembangan perangkat lunak dapat keluar jalur jika pelanggan tidak begitu jelas tentang hasil akhir proyeknya
- Berikut adalah lima *critical factor* yang membedakan antara *Agile* dengan pendekatan tradisional dalam pengembangan sistem :

Factor	Agile Methods	Traditional Methods
Size	Well matched to small products and teams. Reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams. Hard to tailor down to small projects.
Criticality	Untested on safety-critical products. Potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products. Hard to tailor down to products that are not critical.
Dynamism	Simple design and continuous refactoring are excellent for highly dynamic environments but a source of potentially expensive rework for highly stable environments.	Detailed plans and Big Design Up Front, excellent for highly stable environment but a source of expensive rework for highly dynamic environments.
Personnel	Requires continuous presence of a critical mass of scarce experts. Risky to use non-agile people.	Needs a critical mass of scarce experts during project definition but can work with fewer later in the project, unless the environment is highly dynamic.
Culture	Thrives in a culture where people feel comfortable and empowered by having many degrees of freedom (thriving on chaos).	Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear practices and procedures (thriving on order).

Gambar 2.19 *Critical factor* yang membedakan antara pendekatan Agile dengan pendekatan tradisional

Sumber : (Joseph S. Valacich, 2015)

7. Pemodelan Terpadu Secara Cepat (*Agile Unified Process – AUP*)

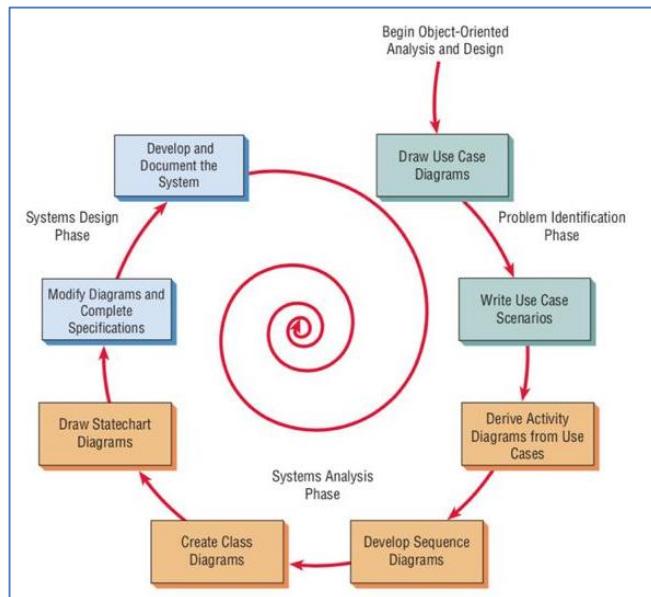
AUP merupakan pendekatan yang menyediakan suatu susunan yang bersifat serial yaitu urutan linier dari kegiatan rekayasa perangkat lunak yang memungkinkan sebuah tim perangkat lunak memvisualkan keseluruhan aliran proses untuk sebuah model perangkat lunak. , namun dalam setiap kegiatan , tim berulang untuk mencapai kecepatan dan untuk memberikan

peningkatan perangkat lunak yang berarti kepada pengguna akhir secepat mungkin. Pada setiap pengulangan AUP mengacup pada kegiatan berikut (Roger S. Pressman, 2012):

- a. Pemodelan , representasi UML untuk bisnis dan ranah masalah diciptakan.
- b. Pelaksanaan, model tersebut di terjemahkan ke dalam kode sumber.
- c. Pengujian, seperti XP, tim perangkat lunak akan merancang dan menjalankan serangkaian pengujian untuk menemukan kesalahan – kesalahan dalam kode sumber yang telah dihasilkan melalui tahapan sebelumnya dan memastikan bahwa kode sumber memenuhi kebutuhan dan harapan para penggunanya.
- d. Pemasangan, seperti kegiatan proses generik , pemasangan dalam konteks ini berfokus pada penghantaran peningkatan perangkat lunak dan perolehan umpan balik dari pengguna akhir.
- e. Konfigurasi dan manajemen proyek, dalam konteks AUP , manajemen konfigurasi menunjukkan perubahan , manajemen risiko dan pengendalian produk kerja yang terus-menerus yang dihasilkan oleh tim. Manajemen proyek menelusuri dan mengendalikan kemajuan tim dan mengoordinasikan kegiatan tim.

f. Lingkungan manajemen, mengkoordinasikan infrastruktur proses yang meliputi standar , alat dna teknologi pendukung lainnya yang tersedia bagi tim.

Dalam pengembangan rekayasa perangkat lunak yang menggunakan UML dapat digambarkan proses sebagai berikut (Kenneth E. Kendall, 2010):



Gambar 2.20 Langkah dalam proses pengembangan perangkat lunak menggunakan UML

BAB 3 Prinsip – Prinsip yang digunakan dalam Praktik Rekayasa Perangkat Lunak

Dalam penerapan proses perangkat lunak yang bermakna , pada dasarnya dipandu oleh prinsip – prinsip inti yang menjadi dasar untuk mengeksekusi metode – metode perangkat lunak yang efektif. Secara rinci prinsip – prinsip yang harus diikuti secara seksama untuk memandu proses – proses dan praktik – praktik rekayasa perangkat lunak modern saat ini adalah sebagai berikut :

3.I. Prinsip – prinsip Inti

Prinisp – prinsip inti yang harus diikuti untuk memandu proses dan praktik perangkat lunak adalah sebagai berikut (Roger S. Pressman, 2012) :

3.I.I. Prinsip – prinsip yang Memandu Proses Rekayasa Perangkat Lunak

Berikut adalah prinsip – prinsip inti yang dapat diterapkan pada kerangka kerja dan dengan perluasannya dapat diterapkan pada setiap proses yang ada di dalam rekayasa perangkat lunak :

- I. Cepat , apapun model proses yang dipilih bersifat preskritif atau cepat , kecepatan seharusnya merupakan pendekatan penting yang akan memandu. Berusaha agar pendekatan teknis kita sesederhana mungkin.

2. Pada setiap langkah berfokuslah pada kualitas, setiap aktivitas proses , tindakan dan pekerjaan seharusnya berfokus pada kualitas produk – produk kerja yang akan dihasilkan.
3. Siap untuk beradaptasi, adaptasilah pendekatan pada batasan yang muncul pada permasalahan ,orang dan pada proyek rekayasa perangkat lunak itu sendiri.
4. Bentuklah tim yang efektif, tim yang efektif adalah tim yang dapat mengorganisasi dirinya sendiri , yang saling mempercayai dan saling menghargai.
5. Menetapkan mekanisme untuk komunikasi dan koordinasi.
6. Mengelola perubahan , mekanisme tertentu harus ditetapkan untuk mengelola bagaimana perubahan – perubahan diminta, dinilai , diterima dan diimplementasikan.
7. Lakukan penilaian terhadap risiko, sesuatu yang buruk mungkin saja terjadi saat perangkat lunak dikembangkan. Dengan demikian merupakan hal yang esensial untuk menetapkan langkah – langkah untuk mengatasinya.
8. Buat produk – produk kerja yang bermanfaat untuk lainnya.

3.1.2. Prinsip – prinsip yang Memandu Praktik Rekayasa Perangkat

Praktik rekayasa perangkat lunak memiliki sasaran yaitu mengirimkan tepat waktu , memiliki kualitas yang tinggi , berupa perangkat lunak operasional yang

memuat fungsi – fungsi dan fitur – fitur yang sesuai dengan kebutuhan semua penyandang dana. Prinsip inti yang menyadari suatu praktik rekayasa perangkat lunak adalah sebagai berikut :

1. Bagi dan pecahkan, sebuah permasalahan yang berukuran besar akan lebih mudah diselesaikan jika dibagi – bagi ke dalam sejumlah elemen – elemen yang merupakan bagianya.
2. Memahami penggunaan abstraksi, abstraksi sesungguhnya adalah suatu penyederhanaan dari beberapa elemen kompleks suatu perangkat lunak yang digunakan untuk mengkomunikasikan suatu makna tertentu dalam bentuk yang relatif lebih sederhana. Abstraksi pada dasarnya adalah untuk menghilangkan kebutuhan untuk komunikasi yang bersifat sangat rinci. Tanpa pemahaman yang baik tentang rincian, penyebab dari permasalahan tidak akan bisa dengan mudah didiagnosa.
3. Berusaha untuk konsisten , prinsip konsistensi pada dasarnya menyarankan sesuatu yang mudah dipahami akan membuat perangkat lunak mudah untuk digunakan.
4. Berfokus pada pengalihan informasi, dalam setiap kasus, informasi – informasi akan mengalir melintasi antarmuka dan sebagai akibatnya muncul kemungkinan – kemungkinan untuk terjadinya kesalahan – kesalahan atau pengabaian – pengabaian atau kemenduan arti, Implikasi dari prinsip ini adalah bahwa kita

harus memberikan perhatian khusus pada analisis , perancangan , konstruksi dan pengujian antarmuka – antarmuka.

5. Kembangkan perangkat lunak yang menekankan pada modularitas yang efektif. Setiap perangkat lunak yang kompleks pada dasarnya dapat dibagi menjadi modul – modul yang merupakan bagian –bagiannya.
6. Mencari pola. Formalisasi penyelesaian permasalahan – permasalahan dan hubungan – hubungannya akan sangat membantu untuk keberhasilan dalam mendapatkan isi pengetahuan yang mendefinisikan pemahaman tentang arsitektur perangkat lunak yang baik, yang akan memenuhi kebutuhan para pengguna.
7. Jika mungkin , lihatlah permasalahan dan penyelesaiannya dari berbagai sudut pandang yang berbeda.
8. Ingatlah bahwa seseorang akan melakukan pemeliharaan terhadap perangkat lunak yang dikembangkan. Aktivitas – aktivitas pemeliharaan perangkat lunak ini dapat difasilitasi dengan baik jika praktik rekayasa perangkat lunak yang baik diterakan di sepanjang proses pengembangan perangkat lunak.

3.2. Prinsip – prinsip yang Memandu setiap Aktivitas Kerangka Kerja dalam rekayasa perangkat lunak

Prinsip – prinsip yang memandu aktivitas kerja adalah sebagai berikut (Roger S. Pressman, 2012) :

3.2.I. Prinsip – prinsip Komunikasi

Berikut adalah prinsip yang secara umum dapat diterapkan pada semua bentuk komunikasi yang terjadi di dalam proyek perangkat lunak .

1. Dengarkan. Berfokuslah pada kata – kata , alih – alih mencoba merumuskan tanggapan atas kata – kata tersbeut. Lakukanlah pertanyaan – pertanyaan klarifikasi jika sesuatu tidak jelas tetapi berusahalah untuk tidak melakukan interupsi.
2. Lakukan persiapan sebelum melakukan komunikasi. Luangkanlah waktu untuk memahami permasalahan tersebut sebelum bertemu dengan orang terkait. Jika mungkin lakukanlah penelitian sederhana untuk memahami jargon – jargon yang sesuai ranah bisnis lawan bicara.
3. Seseorang sebaiknya menjadi fasilitator aktivitas komunikasi. Setiap komunikasi sebaiknya memiliki pemimpin agar percakapan berada pada ranah yang benar , ada yang dapat menjadi penengah konflik dan untuk memastikan prinsip – prinsip komunikasi yang lain diikuti.
4. Komunikasi yang dilakukan secara tatap muka adalah yang terbaik.
5. Lakukan pencatatan dan dokumentasi.
6. Usahakan untuk bekerjasama
7. Tetaplah terfokus , lakukan modularitas pembahasan.

8. Jika sesuatu tidak jelas, gambarlah. Sketsa atau gambar seringkali memperjelas saat kata – kata gagal menjelaskannya.
9. Pada prinsip ke sembilan ini ada tiga catatan yaitu :
 - a. Sekali kita telah membuat kesepakatan tentang suatu pembahasan, bergeraklah ke pembahasan lainnya.
 - b. Jika kita tidak bisa mencapai kesepakatan tentang suatu pokok pembahasan, bergeraklah dengan segera ke pembahasan lainnya.
 - c. Jika suatu fitur atau fungsi tidak jelas dan tidak bisa diklarifikasi saat ini, bergeraklah pada pembahasan berikutnya.
10. Negoisasi bukanlah suatu kontes atau perlombaan . Merupakan hal yang terbaik saat kedua pihak menang.

3.2.2. Prinsip – Prinsip Perencanaan

Aktivitas perencanaan sesungguhnya merupakan sejumlah praktik manajemen dan teknis yang memungkinkan tim perangkat lunak untuk mendefinisikan suatu peta jalan yang pada gilirannya memungkinkan tim perangkat lunak mencapai tujuan – tujuan yang bersifat strategik dan taktis. Berikut adalah prinsip – prinsip perencanaan yang sebaiknya diterapkan :

1. Memahami lingkup proyek. Lingkup proyek menyediakan bagi tim perangkat lunak suatu tujuan.
2. Melibatkan stakeholder pada aktivitas perencanaan. Para stakeholder harus mendefinisikan prioritas – prioritas dan menetapkan batasan proyek perangkat lunak.
3. Memahami bahwa perancangan bersifat iteratif. Suatu perencanaan proyek tidak pernah bersifat kaku. Saat pekerjaan dimulai, sangat mungkin bahwa segala sesuatunya akan mengalami perubahan – perubahan . Sebagai akibatnya , perencanaan harus diatur sedemikian rupa untuk bisa mengakomodasi perubahan – perubahan itu.
4. Melakukan perkiraan berdasarkan apa yang kita mengerti. Perhatian yang perlu dimasukkan dalam suatu perencanaan adalah indikasi usaha, biaya, lamanya penggerjaan pekerjaan – pekerjaan ,berdasarkan pemahaman tim saat ini tentang bagaimana mengerjakan proyek perangkat lunak.
5. Pertimbangkan risiko saat kita mendefinisikan perencanaan. Perencanaan proyek perangkat lunak semestinya disesuaikan dengan risiko – risiko yang mungkin akan terjadi.
6. Cobalah bersikap realistik. Hampir tidak pernah ada rekayasawan perangkat lunak yang tidak pernah membuat kesalahan. Kenyataan ini perlu dipertimbangkan saat perencanaan proyek perangkat lunak.

7. Sesuaikan granulitas saat mendefinisikan perencanaan. Granulitas merujuk pada peringkat rincian yang dilakukan saat perencanaan proyek perangkat lunak dilakukan.
8. Definisikan bagaimana memikirkan hal – hal yang berkaitan dengan kualitas . Perencanaan proyek perangkat lunak seharusnya mengidentifikasi bagaimana tim perangkat lunak berfikir tentang bagaimana cara memastikan kualitas.
9. Deskripsikan bagaimana kita akan melakukan sesuatu yang berkaitan dengan cara kita mengakomodasi perubahan. Bagaimanapun bagusnya perencanaan kita, perencanaan itu dapat saja digagalkan oleh perubahan yang tidak terkendali. Kita harus mengidentifikasi bagaimana perubahan – perubahan akan diakomodasi saat pekerjaan rekayasa perangkat lunak berlangsung.
10. Lakukan pelacakan terhadap perencanaan dan buatlah penyesuaian – penyesuaian jika dibutuhkan.

3.2.3. Prinsip – prinsip Pemodelan

Dalam pekerjaan rekayasa perangkat lunak, ada dua jenis model yang dapat dibuat yaitu model – model yang berkaitan dengan spesifikasi kebutuhan dan model – model yang berkaitan dengan perancangan yang mengarah pada implementasi selanjutnya. Model – model spesifikasi kebutuhan memperlihatkan spesifikasi – spesifikasi kebutuhan pengguna dengan

menggambarkan perangkat lunak dalam tiga ranah yang berbeda yaitu ranah informasi, ranah fungsional dan ranah prilaku. Sedangkan ranah perancangan menggambarkan karakteristik – karakteristik perangkat lunak yang akan sangat membantu para praktisi untuk mengembangkan perangkat lunak secara efektif yaitu arsitektur perangkat lunak, antarmuka pengguna dna rincian berperingkat komponen. Scott Ambler dan Rob Jeffries dalam (Roger S. Pressman, 2012) mendefinisikan sejumlah prinsip pemodelan proses cepat sebagai berikut :

1. Tujuan utama tim perangkat lunak adalah untuk mengembangkan perangkat lunak , bukan model – model.
2. Jangan membuat model lebih banyak dari yang kita butuhkan.
3. Berusalah membuat model sederhana yang menjelaskan permasalahan atau perangkat lunak.
4. Kembangkan model – model sedemikian rupa sehingga perubahan dimungkinkan.
5. Berusalah untuk menetapkan suatu kegunaan eksplisit untuk masing – masing model yang dibuat.
6. Adaptasi model – model yang dikembangkan dengan perubahan yang terjadi pada sistem.
7. Jangan kaku dengan sintaks model. Jika model saat ini dapat mengkomunikasikan isi dengan baik, penampilan adalah nomor dua.
8. Jika secara intuisi kita merasa bahwa model tidak benar meski kelihatan diatas kertas tidak ada

masalah apa – apa , kita mungkin memiliki alasan untuk mempertimbangkan ulang.

9. Dapatkan umpan balik sesegera mungkin.

Prinsip – prinsip Pemodelan Spesifikasi Kebutuhan

- I. Ranah informasi dari suatu permasalahan harus ditampilkan dan dipahami. Ranah informasi menggambarkan data yang mengalir ke dalam sistem , menggambarkan data yang akan keluar dari sistem , menggambarkan penyimpanan data yang mengumpulkan dan mengorganisasi objek – objek data yang bersifat persisten.
2. Fungsi – fungsi yang akan dilakukan perangkat lunak harus didefiniskan. Fungsi – fungsi perangkat lunak tentunya menyediakan manfaat langsung pada para pengguna akhir dan juga menyediakan dukungan internal untuk fitur – fitur yang tampak oleh pengguna.
3. Perilaku perangkat lunak akibat kejadian – kejadian yang bersifat eksternal harus direpresentasikan. Input diberikan oleh pengguna akhir, kendali data dilakukan sistem eksternal atau pemantauan data yang dikumpulkan melintas jaringan, semuanya menyebabkan perangkat lunak berperilaku dengan cara tertentu.
4. Model yang menjelaskan informasi, fungsi dan perilaku harus dipisahkan dalam bentuk yang tidak menyingkapkan rinciannya dan harus digambarkan dalam bentuk perlapisan atau

hierarki. Spesifikasi – spesifikasi kebutuhan memungkinkan untuk memahami permasalahan dan menetapkan penyelesaiannya .

5. Pekerjaan analisis seharusnya bergerak dari informasi yang bersifat esensial menuju rincian implementasi.

Prinsip – prinsip Pemodelan Rancangan :

1. Rancangan harus bisa dilacak balik ke model spesifikasi kebutuhan
2. Selalu pertimbangkan arsitektur perangkat lunak yang akan dikembangkan
3. Perancangan data sama pentingnya dengan perancangan fungsi – fungsi pemrosesan.
4. Antarmuka – antarmuka harus dirancang secara hati – hati.
5. Perancangan antarmuka pengguna seharusnya disesuaikan dengan kebutuhan pengguna. Meski demikian , dalam setiap kasus, antarmuka harus dibuat dengan menekankan kemudahan penggunaannya.
6. Perancangan peringkat komponen sebaiknya mandiri secara fungsional.
7. Komponen – komponen seharusnya bersifat saling tidak bergantung satu sama lain dan juga tidak bergantung pada lingkungan eksternal.
8. Representasi rancangan atau model seharusnya dapat dipaham dengan mudah.
9. Perancangan seharusnya dikembangkan secara iteratif , dimana masing – masing itersi ,

perancangan seharusnya menekankan kesederhanaan .

Saat prinsip – prinsip perancangan diatas diterapkan , maka akan menghasilkan suatu rancangan yang menampilkan baik kualitas internal maupun eksternal . Faktor kualitas eksternal adalah properti perangkat lunak yang dapat dilihat oleh para pengguna. Sedangkan faktor kualitas internal adalah kepentingannya untuk memandu pekerjaan rekayasaan perangkat lunak.

3.2.4. Prinsip – prinsip Konstruksi

Aktivitas konstruksi mencakup di dalamnya sejumlah pekerjaan penulisan kode dan pengujian hingga perangkat lunak yang siap dikirimkan ke pelanggan dan ke para pengguna akhir. Berikut adalah prinsip – prinsip dan konsep – konsep yang mendasari penulisan kode program dan pengujinya .

- I. Prinsip – prinsip persiapan . Sebelum kita menulis baris pertama kode program komputer kita, pastikan bahwa :
 - a. Memahami permasalahan yang akan diselesaikan
 - b. Memahami prinsip – prinsip dan konsep – konsep dasar perancangan
 - c. Telah memilih bahasa pemrograman yang sesuai dengan kebutuhan perangkat lunak yang akan dikembangkan dan lingkungan dimana ia akan bekerja.

- d. Telah memilih lingkungan pemograman yang memiliki perkakas – perkakas yang akan mempermudah pekerjaan.
 - e. Membuat sejumlah pengujian unit yang akan diterapkan setelah kode program selesai.
2. Prinsip – prinsip pemograman. Saat mulai menulis kode program komputer, pastikan bahwa :
- a. Mengimpelemtasikan algoritma yang akan digunakan dalam program komputer yang ditulis dengan mengikuti praktik pemograman terstruktur.
 - b. Mempertimbangkan penggunaan pemograman parsial
 - c. Memilih struktur data yang sesuai dengan apa yang telah ditetapkan oleh rancangan yang telah dibuat sebelumnya.
 - d. Memahami arsitektur program dan membuat antarmuka – antarmuka yang konsisten terhadap arsitektur program
 - e. Membuat logika kondisional sesederhana mungkin
 - f. Membuat peulangan – perulangan bersarang dengan cara sedemikian rupa sehingga perulangan bersarang tersebut mudah diuji.
 - g. Memilih nama – nama perubah yang bermakna dan penamaan itu seharusnya mengikuti standar pengkodean yang

- dimiliki oleh bahasa pemograman yang dipilih.
- h. Menulis kode yang dapat mendokumentasikan dirinya sendiri
 - i. Membuat kode – kode dengan cara yang memudahkan pemahaman.
3. Prinsip – prinsip validasi. Setelah menyelesaikan penulisan kode pastikan bahwa :
- a. Jika mungkin, melakukan penelusuran pada kode programan yang telah ditulis untuk melakukan pemeriksaan ulang kebenaran sintaks dan logikanya.
 - b. Melakukan pengujian – pengujian unit dan melakukan pembetulan kesalahan – kesalahan yang sebelumnya tidak tesingkap
 - c. Melakukan refaktor terhadap kode yang dihasilkan.
4. Prinsip – prinsip pengujian menurut Glen Myers dalam (Roger S. Pressman, 2012)
- a. Pengujian merupakan proses eksekusi suatu program dengan tujuan untuk menemukan kesalahan – kesalahan yang ada didalamnya.
 - b. Kasus pengujian yang baik adalah pengujian yang memiliki kemungkinan yang tinggi untuk menemukan kesalahan – kesalahan yang tidak ditemukan sebelumnya

- c. Pengujian yang berhasil adalah pengujian yang mampu menyingkapkan kesalahan yang tidak ditemukan sebelumnya.

Prinsip pengujian menurut Davis dalam (Roger S. Pressman, 2012) dapat diadaptasi sebagai berikut :

- a. Semua pengujian seharusnya dapat dilacak balik ke spesifikasi kebutuhan pelanggan.
- b. Pengujian – pengujian seharusnya direncanakan lama sebelum pengujian – pengujian itu dimulai
- c. Prinsip pareto dapat diterapkan pada pengujian perangkat lunak. Prinsip Pareto mengatakan bahwa 80 persen kesalahan yang tidak tersingkap selama pengujian dilaksanakan pada umumnya berada di 20 persen komponen – komponen program secara keseluruhan.
- d. Pengujian seharusnya bergerak dari bagian yang lebih kecil dan menerus ke bagian yang semakin besar.
- e. Pengujian yang lengkap merupakan hal yang tidak mungkin.

3.2.5. Prinsip – prinsip Penyerahan

Aktivitas penyerahan perangkat lunak kepada pelanggan (deployment) memiliki tiga aksi penting yaitu pengiriman , dukungan dan umpan balik. Berikut adalah prinsip – prinsip yang seharusnya dipatuhi saat

tim perangkat lunak mengirimkan suatu versi perangkat lunak.

1. Harapan – harapan pelanggan untuk perangkat lunak harus dikelola dengan baik.
2. Paket – paket pengiriman lengkap seharusnya dirakit dan sudah diuji.
3. Sistem – sistem pendukung harus ditetapkan sebelum perangkat lunak dikirimkan.
4. Material – material instruksional yang sesuai harus disediakan bagi para pengguna.
5. Perangkat lunak yang penuh dengan kesalahan seharusnya diperbaiki lebih dahulu , pengiriman bisa dilakukan di waktu – waktu selanjutnya.

BAB 4 Memahami Kebutuhan Rekayasa Perangkat Lunak

Bagi rekayasawan perangkat lunak memahami kebutuhan – kebutuhan untuk se suatu permasalahan mungkin merupakan pekerjaan yang sulit yang biasa dihadapi. Apakah pelanggan tahu apa yang mereka butuh ? Apakah peran pengguna akhir memiliki pemahaman yang cukup baik tentang fitur – fitur dan fungsi – fungsi yang akan memberikan pada mereka keuntungan – keuntungan berkaitan dengan penggunaan perangkat lunak yang dikembangkan ? jawaban dari pertanyaan tersebut adalah tidak, walaupun para pengguna akhir dapat merumuskan kebutuhan – kebutuhan secara eksplisit namun seringnya kebutuhan – kebutuhan mereka mengalami perubahan di sepanjang waktu berjalannya proyek.

4.I. Rekayasa Kebutuhan

Sejumlah besar pekerjaan dan teknik yang dapat dilakukan untuk lebih memahami kebutuhan – kebutuhan biasanya dinamakan sebagai rekayasa kebutuhan. Dari sudut pandang proses perangkat lunak, rekayasa kebutuhan adalah tindakan – tindakan utama dalam kebanyakan rekayasa perangkat lunak yang bermula pada aktivitas – aktivitas komunikasi dan kemudian berlanjut ke aktivitas – aktivitas pemodelan . Rekayasa kebutuhan harus diadaptasi ke kebutuhan – kebutuhan proses , proyek , produk dan orang – orang yang akan mengerjakannya. Rekayasa kebutuhan membangun jembatan dari tahap analisis ke tahapan – tahapan perencanaan dan konstruksi.

Rekayasa kebutuhan pada dasarnya menyediakan mekanisme – mekanisme yang digunakan untuk memahami kebutuhan pelanggan, menyediakan mekanisme yang dapat digunakan untuk melakukan analisis kebutuhan , mekanisme – mekanisme untuk menegoisiasikan solusi atas permasalahan yang memiliki alasan yang kuat, mekanisme – mekanisme untuk menspesifikasikan solusi permasalahan sehingga tidak menimbulkan makna ganda, mekanisme – mekanisme yang digunakan untuk menvalidasi spesifikasi – spesifikasi kebutuhan dan mekanisme – mekanisme yang cocok untuk mengelola kebutuhan – kebutuhan saat ditransformasikan ke dalam perangkat lunak yang bersifat operasional. Pada dasarnya rekayasa kebutuhan mencakup 7 (tujuh) pekerjaan yaitu (Roger S. Pressman, 2012) :

1. Pengenalan masalah (*inception*)

Proyek perangkat lunak dimulai saat kebutuhan bisnis telah teridentifikasi dengan baik , atau saat pasar baru yang bersifat potensial dikenali atau layanan – layanan ditemukan. Pada pengenalan proyek , ditetapkan pemahaman dasar tentang permasalahan , tentang siapa yang menginginkan solusi bagi permasalahan itu, tentang sifat – sifat alamiah solusi yang dikehendaki dan tentang efektivitas komunikasi awal dan kolaborasi yang akan terjadi di antara para stakeholder dan tim perangkat lunak.

2. Pengenalan lanjutan (*elicitation*)

Pada pekerjaan pengenalan lanjutan dilakukan identifikasi tentang apa sesungguhnya sasaran – sasaran dari sistem atau produk yang akan dikembangkan, bagaimana sistem atau produk yang akan

dikembangkan , bagaimana sistem atau produk yang akan dikembangkan sesuai dengan kebutuhan bisnis dan yang terakhir bagaimana penggunaan sistem atau produk yang akan dikembangkan. Christel dan kang dalam (Roger S. Pressman, 2012) mengidentifikasi sejumlah permasalahan yang dijumpai saat tahap pengenalan lanjut ini dilaksanakan yaitu :

- a. Lingkup permasalahan. Batasan – batasan dari sistem tidak jelas atau para pelanggan menentukan rincian teknis yang membingungkan alih – alih menjelaskan sasaran sistem secara keseluruhan.
 - b. Permasalahan yang berkaitan dengan pemahaman. Para pelanggan tidak secara lengkap merasa pasti apa yang mereka butuhkan.
 - c. Permasalahan – permasalahan yang berkaitan dengan kestabilan. Kebutuhan – kebutuhan akan mengalami perubahan – perubahan di sepanjang waktu berjalannya proyek perangkat lunak.
3. Elaborasi
- Elaborasi dilakukan dengan cara membuat atau menghaluskan skenario – skenario pengguna yang pada dasarnya bermanfaat untuk mendeskripsikan bagaimana para pengguna akhir akan berinteraksi dengan sistem. Masing – masing skenario pengguna diuraikan dan ditafsirkan lebih lanjut untuk mendapatkan kelas – kelas analisis yaitu entitas – entitas ranah bisnis yang tampak dari arah para pengguna akhir. Atribut – atribut masing – masing kelas analisis didefinisikan dna layanan – layanan yang diperlukan oleh masing – masing kelas diidentifikasi dan kolaborasi – kolaborasi

antar kelas diidentifikasi dan berbagai diagram pelengkap dihasilkan

4. Negoisasi

Sebagai rekayawan sering harus mendamaikan konflik – konflik melalui negoisasi. Para pengguna, para stakeholder lainnya harus ditanyai tentang prioritas masing – masing kebutuhan dan kemudian mereka bisa diajak untuk membahas masing – masing konflik yang terjadi menurut prioritas masing – masing kebutuhan itu. Gunakan pendekatan yang sifatnya iteratif untuk melakukan penentuan skala priorits kebutuhan – kebutuhan , menilai biaya – biaya, risiko – risikonya masing – masing, dan menyelesaikan konflik internal, dan dengan demikian rekayawan sistem bisa menghapuskan kebutuhan – kebutuhan yang tidak diperlukan , mengabung – gabungkan dan memodifikasi kebutuhan – kebutuhan sedemikian rupa sehingga masing – masing pihak menjadi puas.
5. Spesifikasi

Terminologi spesifikasi memiliki arti yang berbeda – beda bagi setiap orang . sebuah spesifikasi dapat berupa dokumen – dokumen tertulis, sejumlah model – model grafis, suatu model matematika formal, sejumlah skenario penggunaan perangkat lunak, sebuah protipe, atau gabungan semuanya. Beberapa menyarankan agar suatu pola yang bersifat kaku dikembangkan dan digunakan untuk melakukan spesifikasi kebutuhan – kebutuhan dengan pendapat bahwa hal itu memungkinkan kebutuhan – kebutuhan bisa dipresentasikan dengan cara yang konsisten sehingga

pada gilirannya akan menjadi lebih mudah untuk dipahami.

6. Validasi

Produk – produk kerja yang dihasilkan sebagai akibat dilakukannya proses rekayasa kebutuhan dinilai untuk menilai kualitasnya pada setiap tahap validasi . validasi kebutuhan – kebutuhan dilakukan dengan cara melakukan pemeriksaan pada spesifikasi – spesifikasi untuk memastikan bahwa semua kebutuhan perangkat lunak telah didapatkan tanpa adan kemungkinan terjadinya pemahaman ganda yang membingungkan , untuk meastikan bahwa semua tidak ketidakkonsistenan, pengabaian – pengabain dan kesalahan – kesalahan yang terdeteksi telah di perbaiki, dan untuk memastikan bahwa semua produk telah sesuai dengan standar yang telah ditetapkan sebelumnya untuk proses – proses proyek dan produk perangkat lunak sendiri. Mekanisme – mekanisme validasi atas kebutuhan – kebutuhan yang paling penting adalah tinjauan – tinjauan teknis.

7. Pengelolaan kebutuhan

Pengelolaan kebutuhan merupakan sejumlah aktivitas yang akan membantu tim proyek perangkat lunak untuk mengidentifikasi , mengendalikan dan melacak kebutuhan – kebutuhan dan melacak perubahan – perubahan pada kebutuhan – kebutuhan setiap saat di dalam siklus pengembangan sistem perangkat lunak.

Pola untuk spesifikasi kebutuhan perangkat lunak

SRS (*Software Requirement Specification*) merupakan dokumen yang harus dibuat saat deskripsi rinci dari semua aspek perangkat lunak yang akan dikembangkan harus dispesifikasikan sebelum proyek perangkat lunak itu sendiri dimulai. Adapun garis besar topiknya yang diungkapkan oleh Karl Weigers dalam (Roger S. Pressman, 2012) adalah sebagai berikut :

Daftar Isi

Sejarah Revisi

1. Pendahuluan
 - a. Kegunaan
 - b. Konvensi – konvensi(aturan) dalam dokumen
 - c. Pembaca yang dituju
 - d. Lingkup proyek
 - e. Rujukan - rujukan
2. Deskripsi Keseluruhan
 - a. Sudut pandang produk
 - b. Fitur –fitur produk
 - c. Kelas – kelas pengguna dan karakteristik – karakterisknya
 - d. Lingkungan Operasional
 - e. Perancangan dan Implementasi Batasan – batasan
 - f. Dokumentasi pengguna
 - g. Asumsi – asumsi dan ketergantungan – ketergantungan
3. Fitur – fitur sistem
 - a. Fitur sistem - I

- b. Fitur sistem 2 (dst)
- 4. Kebutuhan – kebutuhan antarmuka eksternal
 - a. Antarmuka pengguna
 - b. Antarmuka perangkat keras
 - c. Antarmuka perangkat lunak
 - d. Antarmuka komunikasi
- 5. Kebutuhan – kebutuhan non fungsional lainnya
 - a. Kebutuhan – kebutuhan kinerja
 - b. Kebutuhan – kebutuhan keamanan
 - c. Kebutuhan – kebutuhan kualitas perangkat lunak
- 6. Kebutuhan – kebutuhan lainnya

Lampiran A : Daftar Istilah

Lampiran B : Model – model Analisis

Lampiran C : Daftar Permasalahan

4.2. Menetapkan Pekerjaan Dasar

Langkah – langkah yang diperlukan untuk meletakkan kerangka kerja yang dapat digunakan untuk memahami kebutuhan – kebutuhan perangkat lunak yaitu untuk mendapatkan langkah awal proyek perangkat lunak yang kemudian akan bergerak maju untuk mendapatkan solusi – solusi atas permasalahan yang berhasil. Berikut langkah – langkah dasar dalam menetapkan pekerjaan dasar (Roger S. Pressman, 2012):

4.2.I. Mengidentifikasi orang – orang yang berkepentingan

Sommerville dan Sawyer mendefinisikan orang – orang yang berkepentingan (stakeholder) sebagai semua orang yang mendapatkan keuntungan langsung maupun tidak langsung dari perangkat lunak yang akan dikembangkan. Masing – masing orang – orang yang berkepentingan sering memiliki pandangan yang berbeda – beda tentang sistem, mendapatkan keuntungan yang berbeda saat sistem secara berhasil dikembangkan, dan mungkin saja mendapatkan berbagai risiko yang berbeda saat upaya pengembangan sistem perangkat lunak gagal. Pada tahap pengenalan , seharusnya mendaftarkan orang – orang yang akan memiliki kontribusi atas kebutuhan – kebutuhan yang akan ditangkap (Roger S. Pressman, 2012).

4.2.2. Mengenali Berbagai Sudut Pandang yang Berbeda

Karena ada banyak orang – orang yang berkepentingan , kebutuhan perangkat lunak harus digali secara mendalam dari berbagai perspektif yang berbeda. Masing – masing stakeholder akan menyumbangkan informasi – informasi tertentu pada proses rekayasa kebutuhan . saat informasi – informasi dari berbagai perspektif dikumpulkan, kebutuhan – kebutuhan itu mungkin tidak konsisten satu dengan lainnya saat digabungkan atau memiliki konflik antar kebutuhan. Dalam hal ini seharusnya melakukan pengelompokan terhadap masing – masing informasi itu sedemikian rupa sehingga para pengambil keputusan dapat memilih suatu kumpulan kebutuhan perangkat lain secara internal konsisten (Roger S. Pressman, 2012).

4.2.3. Bekerja melalui Kolaborasi

Para pelanggan dan para stakeholder dan rekayawan perangkat lunak harus bekerjasama atau berkolaborasi jika mereka menginginkan perangkat lunak yang akan dikembangkan hasilnya bagus. Kolaborasi tidak berarti bahwa kebutuhan – kebutuhan didefinisikan oleh kelompok. Dalam banyak kasus, para stakeholder berkolaborasi dengan cara menyediakan pandangan mereka masing – masing tentang kebutuhan – kebutuhan , tetapi jagoan proyek yang kuat mungkin akan melakukan pengambilan keputusan yang sifatnya final (Roger S. Pressman, 2012).

4.2.4. Mananyakan Pertanyaan – pertanyaan pertama

Pertanyaan – pertanyaan yang ditanyakan pada tahap pengenalan suatu proyek perangkat lunak seharusnya bersifat bebas dari konteks, dan pada dasarnya berfokus pada pelanggan serta penyandang dana dan orang – orang yang berkepentingan , serta fokus juga pada sasaran dan keuntungan proyek pengembangan perangkat lunak secara keseluruhan. Pertanyaan – pertanyaan tersebut akan membantu mengidentifikasi semua stakeholder pada perangkat lunak yang akan dikembangkan. Selain itu digunakan untuk mengidentifikasi keuntungan – keuntungan yang dapat diukur dari implementasi yang berhasil dan dapat digunakan untuk mengidentifikasi adanya alternatif – alternatif lain yang mungkin untuk melakukan kustomisasi pengembangan perangkat lunak. Pertanyaan – pertanyaan tersebut juga memungkinkan rekayasaawan mendapatkan pemahaman yang lebih tentang permasalahan dan memungkinkan pelanggan menyuarakan perspektif mereka tentang solusi yang ditawarkan oleh rekayasaawan kebutuhan (Roger S. Pressman, 2012).

4.3. Pengenalan Permasalahan lebih lanjut untuk Mendapatkan Kebutuhan – kebutuhan

Pengenalan lebih lanjut untuk mendapatkan kebutuhan – kebutuhan menggabungkan unsur – unsur penyelesaian masalah, elaborasi, negoisasi dan spesifikasi. Tujuannya agar menjadi bersifat kolaboratif , pendekatan berorientasi pada tim untuk memperoleh kebutuhan – kebutuhan , orang – orang

yang berkepentingan harus saling bekerja sama untuk mengidentifikasi permasalahan , mengusulkan solusi – solusi atas permasalahan – permasalahan , menegoisasiakan pendekatan – pendekatan yang berbeda dan menetapkan sejumlah solusi kebutuhan yang bersifat pendahuluan. Berikut langkah – langkah untuk pengenalan masalah lebih lanjut untuk mendapatkan kebutuhan – kebutuhan (Roger S. Pressman, 2012):

4.3.I. Penentuan Kebutuhan – kebutuhan yang Bersifat Kolaboratif

Panduan untuk menentukan kebutuhan – kebutuhan yang bersifat kolaboratif adalah (Roger S. Pressman, 2012):

1. Pertemuan – pertemuan dipimpin dan dikuti oleh baik rekayawan maupun stakeholder
2. Aturan – aturan untuk persiapan dan partisipasi ditetapkan
3. Agenda dibuat dengan bentuk yang cukup formal untuk bisa memperoleh semua poin yang penting.
4. Seorang fasilitator bisa ditunjuk untuk mengendalikan pertemuan
5. Mekanisme definisi dapat digunakan

Sasaran dari kegiatan ini adalah untuk mengidentifikasi permasalahan – permasalahan , mengusulkan solusi – solusi atas permasalahan – permasalahan , menegoisasiakan pendekatan – pendektan yang berbeda dan mengspesifikasikan sejumlah kebutuhan dalam

atmosfir yang kondusif untuk mencapai sasaran dari solusi perangkat lunak.

4.3.2. QFD (*Quality Function Deployment*)

QFD merupakan teknik manajemen kualitas yang bekerja dengan cara menerjemahkan kebutuhan – kebutuhan pelanggan menjadi kebutuhan – kebutuhan teknis untuk perangkat lunak. QFD mengidentifikasi 3 jenis kebutuhan yaitu (Roger S. Pressman, 2012):

- I. Kebutuhan – kebutuhan normal
Tujuan – tujuan dan sasaran – sasaran produk ditentukan saat terjadinya pertemuan – pertemuan dengan para pelanggan. Jika kebutuhan – kebutuhan itu ditemukan pada produk atau sistem , para pelanggan akan terpuaskan.
2. Kebutuhan – kebutuhan yang diharapkan
Kebutuhan – kebutuhan bersifat implisit pada produk dan mungkin saja bersifat mendasar secara teknis sehingga para pelanggan tidak dapat secara eksplisit menyatakan ketidakhadiran kebutuhan – kebutuhan yang diharapkan ini akan mengakibatkan ketidakpuasan yang mendalam.
3. Kebutuhan – kebutuhan yang menyenangkan
Kebutuhan – kebutuhan yang menyenangkan ini berada di luar apa yang diharapkan para pelanggan dan terbukti akan sangat memuaskan para pelanggan itu jika kebutuhan – kebutuhan

yang menyenangkan memang dapat dipersembahkan pada mereka.

QFD biasanya menggunakan wawancara dengan pengguna dan observasi – observasi ,survei – survei dan penelitian terhadap data yang bersifat historis sebagai data mentah untuk aktivitas penangkapan kebutuhan – kebutuhan. Data itu kemudian diterjemahkan kedalam tabel – tabel kebutuhan yang dinamakan sebagai tabel suara para pelanggan yang kemudian ditinjau oleh para pelanggan dan orang – orang yang berkepentingan lainnya. Sejumlah diagram , matriks dan metode evaluasi bisa digunakan untuk mendapatkan kebutuhan – kebutuhan yang diharapkan dan bisa digunakan untuk mengupayakan kebutuhan – kebutuhan yang menyenangkan (Roger S. Pressman, 2012).

4.3.3. Skenario Penggunaan

Setelah kebutuhan – kebutuhan perangkat lunak berhasil diperoleh, visi tentang keseluruhan fungsi dan fitur sistem yang akan dikembangkan mulai jelas. Namun tetap saja masih untuk bergerak langsung pada aktivitas rekayasa perangkat lunak sampai memahami fungsi – fungsi dan fitur – fitur yang akan digunakan. Untuk menyelesaikan permasalahan ini pengembang dan para stakeholder dapat membuat sejumlah skenario yang dapat digunakan untuk mengidentifikasi sejumlah penggunaan sstem yang mungkin. Skenario – skenario ini disebut use case yang menyediakan deskripsi yang rinci tentang bagaimana sistem akan digunakan (Roger S. Pressman, 2012).

Use case name:	Register for Conference	UniqueID: Conf RG 003
Area:	Conference Planning	
Actor(s):	Participant	
Stakeholder:	Conference Sponsor, Conference Speakers	
Level:	Blue	
Description:	Allow conference participant to register online for the conference using a secure Web site.	
Triggering Event:	Participant uses Conference Registration Web site, enters userID and password, and clicks the logon button.	
Trigger type:	<input checked="" type="checkbox"/> External <input type="checkbox"/> Temporal	
Steps Performed (Main Path)		
1. Participant logs in using the secure Web server.	Information for Steps userID, Password	
2. Participant record is read and password is verified.	Participant Record, userID, Password	
3. Participant and session information is displayed on the Registration Web page.	Participant Record, Session Record	
4. Participant enters information on the Registration Web form and clicks Submit button.	Registration Web Form	
5. Registration information is validated on the Web server.	Registration Web Form	
6. Registration Confirmation page is displayed to confirm registration information.	Confirmation Web Page	
7. Credit card is charged for registration fees.	Secure Credit Card Web Page	
8. Add Registration Journal record is written.	Confirmation Web Page	
9. Registration record is updated on the Registration Master.	Confirmation Web Page, Registration Record	
10. Session record is updated for each selected session on the Session Master.	Confirmation Web Page, Session Record	
11. Participant record is updated for the participant on the Participant Master.	Confirmation Web Page, Participant Record	
12. Successful Registration Confirmation Web page is sent to the participant.	Registration Record Confirmation Number	
Preconditions:	Participant has already registered and has created a user account.	
Postconditions:	Participant has successfully registered for the conference.	
Assumptions:	Participant has a browser and a valid userID and password.	
Success Guarantee:	Participant has registered for the conference and is enrolled in all selected sessions.	
Minimum Guarantee:	Participant was able to logon.	
Requirements Met:	Allow conference participants to be able to register for the conference using a secure Web site.	
Outstanding Issues:	How should a rejected credit card be handled?	
Priority:	High	
Risk:	Medium	

Gambar 4.1 Contoh skenario usecase

Sumber : (Kenneth E. Kendall, 2010)

4.3.4. Produk Kerja Pengenalan Lanjutan

Produk kerja dari pengenalan lanjutan adalah (Roger S. Pressman, 2012) :

1. Pernyataan tentang kebutuhan – kebutuhan sistem dan kelayakannya.
2. Pernyataan lengkap tentang lingkup sistem dan produk.
3. Daftar para pelanggan , pengguna , orang – orang lain yang berkepentingan , yang berpartisipasi dalam pengenalan lanjut untuk mendapatkan kebutuhan – kebutuhan sistem.
4. Deskripsi yang berkaitan dengan lingkungan teknis yang dimiliki sistem
5. Daftar kebutuhan – kebutuhan dan batasan – batasan yang dapat diterapkan pada masing – masing kebutuhan.
6. Skenario – skenario pengguna yang menyediakan wawasan – wawasan yang berkaitan dengan penggunaan sistem atau produk saat produk di eksekusi di bawah berbagai kondisi operasi yang berbeda.
7. Semua prototipe yang dikembangkan untuk dapat mendefinisikan kebutuhan – kebutuhan sistem dengan cara yang lebih baik.

4.4. Mengembangkan Use Case

Use case pada dasarnya memperlihatkan sistem atau perangkat lunak dari sudut pandang para pengguna akhir. Use case selalui di definisikan dari sudut pandang aktor – aktor yang

berinteraksi dengan perangkat lunak yang sedang dikembangkan. Sebuah aktor pada dasarnya merupakan peran yang orang – orang saat mereka berinteraksi dengan perangkat lunak. langkah pertama yang harus dilakukan saat hendak menuliskan sebuah use case adalah mendefinisikan sejumlah aktor yang terlibat dalam cerita. Aktor merupakan sejumlah orang yang berbeda yang menggunakan produk dalam konteks fungsi – fungsi dan perilaku – perilaku yang harus dideskripsikan. Perlu dicatat , aktor dan pengguna adalah sesuatu yang berbeda. Seorang pengguna mungkin saja memainkan sejumlah peran yang berbeda saat pengguna yang bersangkutan menggunakan perangkat lunak , sementara sebuah aktor merepresentasikan sebuah kelas eksternal yang memainkan suatu peran dalam konteks suatu kasus pengguna. Saat aktor – aktor teridentifikasi dengan baik , use case dapat dikembangkan. Pada kebanyakan kasus use case dielaborasikan lebih lanjut sehingga pengembang mendapatkan lebih banyak rincian tentang interaksi – interaksi yang mungkin terjadi antara aktor dengan perangkat lunak yang akan dikembangkan

Deskripsi rincian tentang suatu use case mencakup :

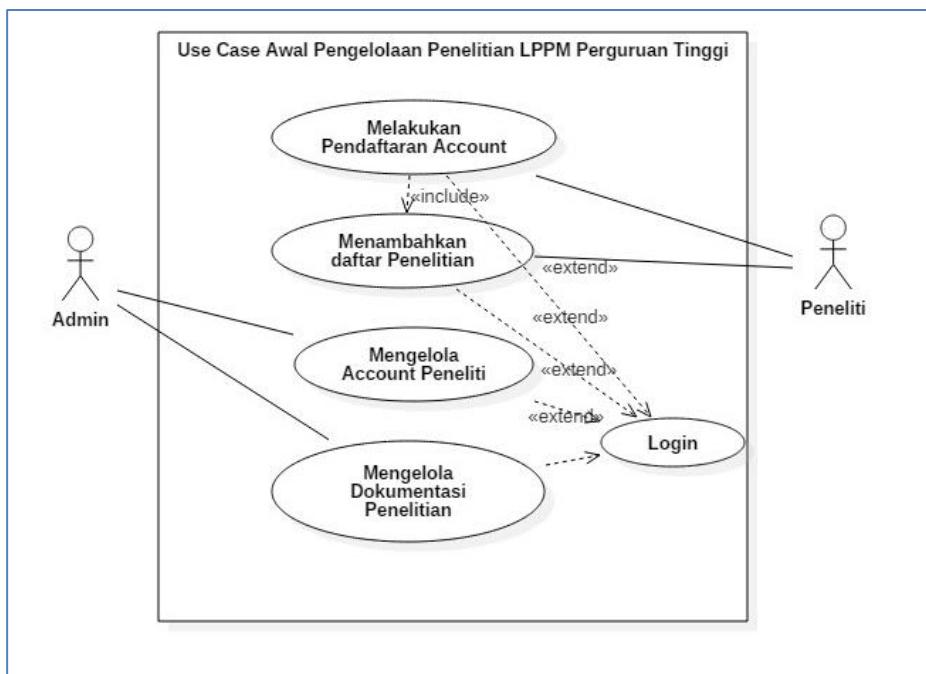
Tabel 4.I Contoh deskripsi use case

I. Use case name	Pengelolaan Penelitian di LPPM
2. Aktor Primer	Peneliti
3. Sasaran dalam kontek	Untuk melakukan penyimpanan dokumentasi penelitian,

	sehingga dokumentasi penelitian dapat di publish pada portal <i>research community</i>
4. Prakondisi	Aktor telah diberikan akses berupa account untuk login memasuki portal
5. Pemicu	Peneliti memutuskan untuk menambahkan dokumentasi penelitian
6. Skenario	<ul style="list-style-type: none"> a. Peneliti membuka link dashboard b. Peneliti memasukkan user dan password c. Peneliti memilih tombol ok d. Peneliti memilih menu add research e. Peneliti mengisi data research f. Peneliti mengupload dokumentasi penelitian g. Peneliti memilih tombol save

7. Pengecualian	<p>a. Link dashboard tidak dapat membuka dashboard : peneliti memeriksa koneksi internet , kemudian peneliti harus mengkoneksikan computer ke internet</p> <p>b. Kata sandi tidak benar : peneliti memasukkan ulang password</p> <p>c. Kata sandi tidak dikenali : sistem memberi tanggapan dengan memberikan peringatan</p>
8. Permasalahan – permasalahan yang terbuka	<p>a. Adakah cara peneliti menambahkan dokumentasi research dalam bentuk demo video atau bentuk lain?</p> <p>b. Haruskah sistem menampilkan pesan – pesan teks tambahan untuk</p>

	<p>validasi penyimpanan data atau validasi akses masuk sistem?</p> <p>c. Apakah sistem dapat mendeteksi bahwa data research sudah ada , sehingga tidak ada duplikasi data ?</p> <p>d. Apakah sistem dapat memberikan wadah media untuk komunikasi antar peneliti ?</p> <p>e. Apakah sistem dapat memberikan fasilitas pemberian komentar pada dokumentasi research yang telah di publish ?</p>
--	--



Gambar 4.2 Diagram Use Case Awal Pengelolaan Penelitian LPPM Perguruan Tinggi

4.5. Mengembangkan Model Kebutuhan

Sasaran model analisis sesungguhnya adalah untuk memberikan deskripsi dari ranah informasional , fungsional dan prilaku yang dibutuhkan untuk sistem – sistem yang berbasis komputer. Model analisis merupakan gambaran kebutuhan – kebutuhan pada suatu waktu tertentu. Saat model kebutuhan berkembang, beberapa elemen akan menjadi relatif stabil , menyediakan dasar yang kokoh untuk pekerjaan perancangan yang akan mengikuti tahap analisis ini. Namun

demikian, elemen – elemen lainnya dari model analisis mungkin akan terus mengalami perubahan dimana hal ini menunjukkan bahwa orang - orang yang berkepentingan tidak secara peripurna memahami kebutuhan – kebutuhan sistem yang akan dikembangkan.

4.5.I. Elemen – elemen Model Kebutuhan

Ada beberapa elemen cara yang berbeda untuk mencari spesifikasi kebutuhan – kebutuhan untuk sistem berbasis komputer yang akan dikembangkan. Praktisi – praktisi perangkat lunak percaya bahwa merupakan hal yang sangat baik untuk menggunakan berbagai bentuk representasi yang berbeda untuk mendapatkan model kebutuhan . Bentuk – bentuk representasi yang berbeda pada dasarnya akan memaksa pengembang untuk mempertimbangkan kebutuhan – kebutuhan perangkat lunak dari berbagai sudut pansang yang berbeda yaitu suatu pendekatan yang pada dasarnya memiliki peluang tinggi untuk dapat menyingskapkan pengabaian – pengabaian , ketidakkonsistenan dan ambiguitas. Ada sejumlah elemen yang bersifat generik yang bersifat umum untuk kebanyak model kebutuhan .

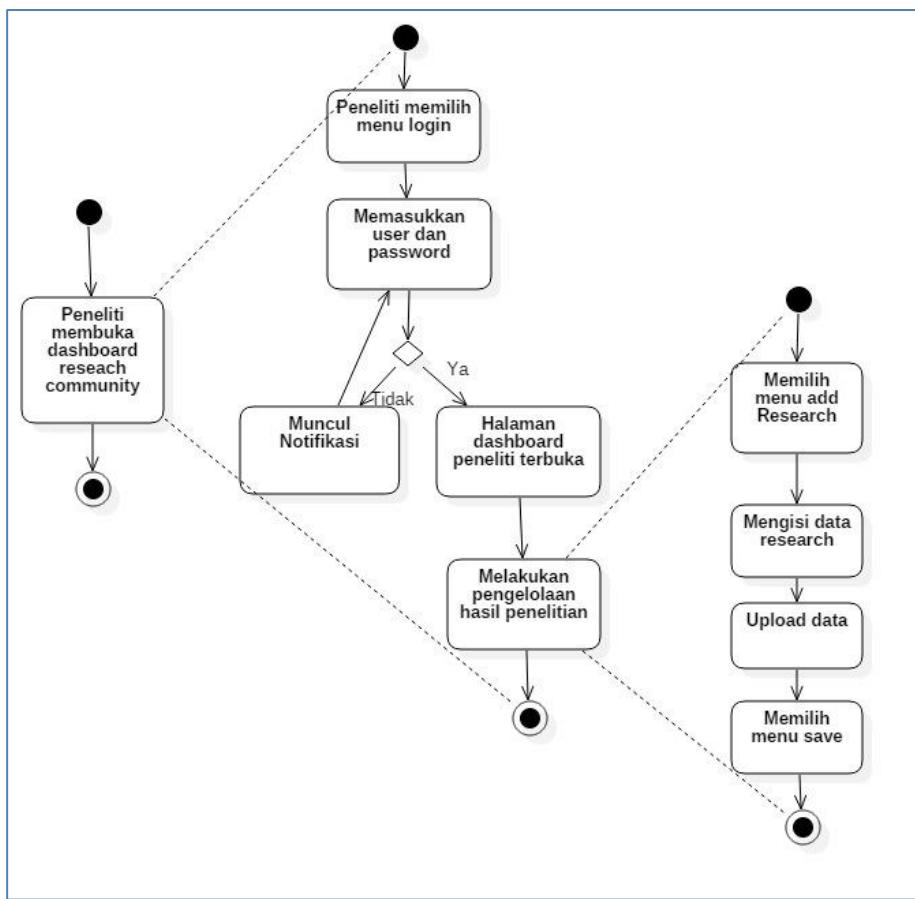
I. Elemen – elemen berbasis skenario

Perangkat lunak dideskripsikan dari sudut penadang pengguna menggunakan pendekatan berbasis skenario. Contohnya adalah use case pada gambar 4.1 yang bersifat dasar dan diagram use case tersebut akan dikembangkan menjadi

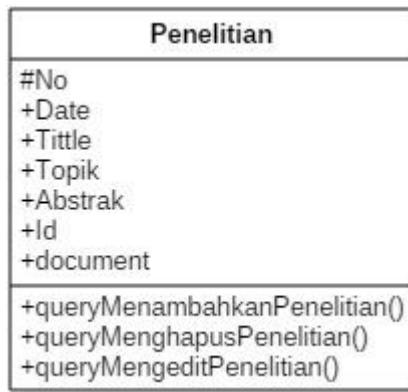
use case – use case berbasis pola yang akan lebih mudah dielaborasikan.

2. Elemen – elemen berbasis kelas

Masing – masing skenario penggunaan pada dasarnya mengimplikasikan sejumlah objek yang dapat dimanipulasi oleh perangkat lunak saat sebuah aktor melakukan interaksi dengannya. Objek – objek ini kemudian dapat dikelompokkan ke dalam kelas – kelas yaitu sebuah kumpulan dari sesuatu yang memiliki atribut – atribut yang serupa dan memiliki perilaku – perilaku yang umum sifatnya. Contohnya adalah diagram kelas UML. Diagram kelas itu memperlihatkan daftar atribut dan operasi – operasi yang dapat diterapkan untuk memodifikasi atribut – atribut yang dimiliki kelas tersebut. Sebagai tambahan pada diagram – diagram kelas , elemen – elemen pemodelan dianalisis lainnya memperlihatkan bagaimana kelas – kelas saling berkolaborasi satu dengan lainnya dan juga memperlihatkan relasi – relasi serta interaksi – interaksi yang terjadi di antara suatu kelas dengan kelas – kelas lainnya.



Gambar 4.3 Diagram aktivitas untuk pengelolaan penelitian pada LPPM Perguruan Tinggi



Gambar 4.4 Diagram Class untuk penelitian

3. Elemen – elemen Perilaku

Perilaku suatu sistem berbasis komputer bisa saja memiliki imbas tertentu pada rancangan tertentu yang dipilih dan mungkin juga memiliki imbas tertentu pada pendekatan implementasi yang akan diterapkan. Model – model kebutuhan harus menyediakan elemen – elemen pemodelan yang memperlihatkan perilaku perangkat lunak yang akan dikembangkan. Diagram state UML adalah suatu metode yang memperlihatkan perilaku sistem dengan memperhatikan state-nya . Sebuah state pada dasarnya merupakan setiap modus perilaku yang dapat diobservasi secara eksternal. Diagram state memperlihatkan tindakan – tindakan tertentu yang diambil sebagai akibat terjadinya suatu

event tertentu. Diagram state merepresentasikan prilaku sistem secara keseluruhan.

4. Elemen – elemen berorientasi aliran
Informasi – informasi akan ditransformasi dalam bentuk aliran di dalam sistem berbasis komputer. Sistem menerima masukan – masukan dalam berbagai bentuk, menerapkan fungsi – fungsi untuk melakukan transformasi terhadap masukan – masukan itu dan menghasilkan keluaran – keluaran dalam bentuk beragam.

4.5.2. Pola – pola Analisis

Setiap orang yang telah menyelesaikan rekayasa kebutuhan untuk beberapa proyek perangkat lunak seringkali memperhatikan beberapa masalah yang terjadi berulangkali pada semua proyeknya yang memiliki ranah aplikasi yang bersifat spesifik. Pola – pola analisis ini menghadirkan solusi – solusi dalam ranah aplikasi yang dapat digunakan ulang saat melakukan pemodelan untuk banyak aplikasi.

4.6. Cara Menegosiasikan Kebutuhan – kebutuhan

Dalam konteks rekayasa kebutuhan yang ideal, pengenalan permasalahan , pengenalan permasalahan lebih lanjut dna elaborasi, sering sekali dapat digunakan untuk menentukan spesifikasi kebutuhan pelanggan secara cukup rinci sehingga tim perangkat lunak langsung dapat bekerja ke tahapan – tahapan selanjutnya dalam aktivitas – aktivitas rekayasa perangkat lunak. kenyataanya kita mungkin harus masuk ke

tahap negoisasi dengan satu atau lebih penyandang dana dan orang – orang yang berkepentingan. Sasaran dari negoisasi pada dasarnya adalah untuk mengembangkan suatu perencanaan proyek yang sesuai dengan keinginan para penyandang dana dan orang – orang yang berkepentingan semnetara pada saat yang bersamaan beusaha untuk melakukan penyesuaian – penyesuaian dengan batasan – batasan yang ada yang ditentukan pada tim perangkat lunak.

Boehm dalam (Roger S. Pressman, 2012) mendefinisikan sejumlah aktivitas negoisasi pada saat awal masing – masing iterasi peroses perangkat lunak. Alih – alih merupakan aktivitas komunikasi pelanggan tunggal. Berikut aktivitas yang dapat dilakukan pada negosiasi kebutuhan :

1. Melakukan identifikasi para stakeholder kunci untuk sistem atau subsistem yang akan dikembangkan
2. Menentukan kondisi menang yang akan didapatkan oleh mereka yang berkepentingan
3. Menegoisasikan kondisi menang oarang – orang yang berkepentingan untuk berusaha merekonsiliasikannya ke dalam sejumlah kondisi menang – menang untuk semua orang yang terlibat dalam negoisasi.

4.7. Cara Menvalidasi Kebutuhan – kebutuhan

Saat masing – masing elemen dari model kebutuhan dibuat, elemen – elemen itu harus diperiksa ketidakconsitenannya, harus diperiksa untuk mengetahui adanya pengabaian – pengabain serta ambiguitas. Kebutuhan – kebutuhan tersebut dilakukan tinjauan oleh orang – orang yang berkepentingan.

Suatu tinjauan model kebutuhan seharusnya dapat menjawab pertanyaan – pertanyaan berikut (Roger S. Pressman, 2012) :

1. Apakah masing – masing kebutuhan konsisten dengan sasaran untuk produk secara keseluruhan ?
2. Apakah semua kebutuhan telah dispesifikasi pada peringkat abstraksi yang semestinya ? yaitu apakah kebutuhan – kebutuhan menyediakan peringkat rincian teknis yang tidak sesuai untuk diakomodasi pada tahap ini ?
3. Apakah kebutuhan memang diperlukan atau kebutuhan itu sesungguhnya memperlihatkan fitur – fitur tambahan yang mungkin tidak terlalu penting bagi sasaran sistem ?
4. Apakah masing – masing kebutuhan menyatu dan tidak ambigu ?
5. Apakah masing – masing kebutuhan memiliki pengusul ? yaitu apakah masing – masing sumber kebutuhan itu tercatat ?
6. Apakah ada kebutuhan – kebutuhan tertentu yang bertentangan dengan kebutuhan – kebutuhan lainnya ?
7. Apakah masing – masing kebutuhan dapat dicapai dalam konteks lingkungan teknik yang akan digunakan oleh sistem atau produk yang akan dikembangkan?
8. Apakah masing – masing kebutuhan dapat diuji setelah masing – masing kebutuhan itu diimplementasikan /
9. Apakah model – model kebutuhan merefleksikan informasi, fungsi dan perilaku sistem yang akan dikembangkan dengan cara yang sesuai ?

10. Apakah model – model kebutuhan telah dipartisi sedemikian rupa sehingga kelak dapat secara progresif memberi informasi yang reinci tentang sistem atau perangkat lunak yang akan dikembangkan ?
11. Apakah pola – pola kebutuhan telah digunakan untuk menyederhanakan model – model kebutuhan ? apakah semua pola telah divalidasi secara semestinya ? apakah semua pola konsisten dengan kebutuhan – kebutuhan pelanggan ?

4.8. Metode Pengumpulan Data

4.8.I. Interview

Interview dalam rangka pengumpulan informasi adalah percakapan terarah dengan tujuan khusus menggunakan format tanya jawab. Interview juga merupakan usaha dalam mengeksplorasi kunci masalah interaksi antara manusia dengan komputer , kegunaan sistem , bagaimana menyenangkan sistem dan bagaimana sistem berguna dalam mendukung tugas individu. Berikut adalah langkah – langkah dalam mempersiapkan interview :

1. Pahami latar belakang organisasi , pahami latar belakang orang yang akan diwawancara dan organisasinya , dengan tujuan analisis dapat menyusun kosa kata yang sesuai dan memaksimalkan waktu dalam wawancara.
2. Tetapkan tujuan wawancara
3. Putuskan siapa yang akan diwawancara
4. Siapkan orang yang diwawancara
5. Tentukan jenis dan struktur pertanyaan

4.8.2. Joint Application Design

Merupakan teknik yang memungkinkan pengembang, manajemen dan kelompok pelanggan bekerja sama untuk membangun suatu produk. Pendekatan ini dapat memberikan manfaat yaitu menyingkat waktu , meningkatkan mutu hasil informasi dan menciptakan identifikasi lebih banyak pengguna dengan sistem informasi sebagai hasil proses – proses yang partisipatif.

Berikut adalah kondisi yang memungkinkan penggunaan Joint Application Design :

1. Kelompok pengguna menginginkan sesuatu yang baru untuk penyelesaian masalah yang ada.
2. Budaya organisasi yang mendukung perilaku penyelesaian masalah secara bersama – sama antar pegawai dari level yang berbeda.
3. Penganalisis memprediksi bahwa jumlah ide – ide yang dapat dihasilkan melalui wawancara empat – mata tidak sebanyak ide yang dihasilkan dari perluasan pengamatan kelompok.

Kelebihan dari penggunaan Joint Application Design adalah :

1. Menghemat waktu wawancara tradisional empat mata
2. Memungkinkan perkembangan yang cepat.
3. Pengembangan desain yang lebih kreatif.

Kekurangan dari penggunaan Joint Application Design adalah :

1. Joint Application Design membutuhkan komitmen waktu
2. Jika persiapan setiap sesi Joint Application Design tidak cukup memadai , atau bila laporan tindak lanjut serta dokumentasi untuk spesifikasi – spesifikasi tertentu tidak lengkap, membuat desain menjadi kurang memuaskan.
3. Keahlian – keahlian organisasi dan budaya yang diperlukan tidak cukup untuk dapat dikembangkan sehingga memungkinkan upaya – upaya bersama yang lebih produktif dalam menyusun Joint Application Design.

4.8.3. Menggunakan Kuesioner

Penggunaan kuesioner merupakan teknik pengumpulan informasi yang memungkinkan analis sistem untuk mempelajari sikap , keyakinan , perilaku dan karakteristik dari beberapa orang kunci dalam organisasi yang mungkin dipengaruhi oleh sistem saat ini dan yang diusulkan. Sikap adalah apa yang orang – orang dalam organisasi inginkan dalam hal ini sistem baru. Keyakinan adalah apa yang orang anggap benar sedangkan perilaku adalah apa yang anggota organisasi lakukan. Karakteristik adalah sifat orang atau benda.

Berikut adalah contoh kuesioner (Kenneth E. Kendall, 2010) :

53. What are the most frequent problems you experience with computer output?
A. _____
B. _____
C. _____

54. Of the problems you listed above, what is the single most troublesome?

55. Why?

Below are questions about yourself. Please fill in the blanks to the best of your ability.

67. How long have you worked for this company?
____ Years and ____ Months

68. How long have you worked in the same industry?
____ Years and ____ Months

69. In what other industries have you worked?

... or short answers.

Open-ended questions can ask the respondent for lists ...

... or detailed responses ...

The diagram illustrates a survey form with several questions and accompanying notes. Question 53 asks about frequent problems with computer output, with lines for A, B, and C. Question 54 asks for the most troublesome problem, with a single line. Question 55 asks why, with three lines. Below these is a general instruction to fill in to the best of one's ability. Questions 67 and 68 ask about work duration, with lines for years and months. Question 69 asks about other industries worked in, with a single line. Three yellow sticky notes provide context: one on the right says 'Open-ended questions can ask the respondent for lists ...' with an arrow pointing to question 53; another note below it says '... or detailed responses ...' with an arrow pointing to question 55; and a third note on the left says '... or short answers.' with an arrow pointing to the short answer lines for question 53.

Gambar 4.4 Contoh kuesioner

BAB 5 Pemodelan Spesifikasi Kebutuhan

Perangkat Lunak

Pada peringkat teknis, pekerjaan rekayasa perangkat lunak pada umumnya dimulai dengan sejumlah pekerjaan pemodelan yang menunjukkan spesifikasi kebutuhan perangkat lunak dan representasi perancangan perangkat lunak yang akan dikembangkan. Model – model spesifikasi kebutuhan perangkat lunak merupakan sejumlah model dan merupakan representasi awal perangkat lunak.

5.I. Analisis Kebutuhan

Analisis spesifikasi kebutuhan menghasilkan spesifikasi – spesifikasi dari karakteristik – karakteristik operasional yang akan dimiliki oleh perangkat lunak yang akan dikembangkan, yang dapat mengindikasikan antarmuka perangkat lunak dengan elemen – elemen sistem yang lain dan juga menetapkan batasan – batasan yang harus dihadapi perangkat lunak. Pekerjaan – pekerjaan pemodelan spesifikasi – spesifikasi kebutuhan pada dasarnya akan menghasilkan jenis model berikut (Roger S. Pressman, 2012) :

1. Model berbasis skenario, yang menggambarkan spesifikasi kebutuhan perangkat lunak dari berbagai sudut pandang aktor perangkat lunak.
2. Model data , yang menjelaskan ranah informasi untuk permasalahan yang akan diselesaikan

3. Model berorientasi kelas , yang memperlihatkan kelas – kelas dalam konteks pemograman berorientasi objek dan dengan cara bagaimana kelas – kelas tersebut saling bekerjasama untuk mencapai sasaran – sasaran spesifikasi – spesifikasi kebutuhan perangkat lunak.
4. Model berorientasi aliran, yang menggambarkan elemen – elemen fungsional sistem perangkat lunak dan yang menggambarkan bagaimana cara mereka melakukan transformasi terhadap data, saat data yang bersangkutan melintasi sistem.
5. Model prilaku, yang menggambarkan bagaimana perangkat lunak berperilaku terhadap event – event yang datang dari luar sistem

Model – model analisis diatas memberikan informasi kepada perancang perangkat lunak untuk diterjemahkan menjadi arsitektur sistem, antarmuka – antarmuka sistem dan perancangan berperingkat komponen. Terakhir model – model spesifikasi kebutuhan memungkinkan para pengembang dan para stakeholder melakukan penilaian kualitas ketika perangkat lunak selesai dikembangkan.

5.I.I. Filosofi dan Sasaran – sasaran secara keseluruhan

Di sepanjang pemodelan analisis untuk mendapatkan spesifikasi – spesifikasi kebutuhan perangkat lunak harus berfokus pada apa yang terjadi saat pengguna berinteraksi dalam suatu keadaan tertentu , objek – objek apa yang akan dimanipulasi oleh perangkat lunak, fungsi – fungsi apa yang harus dapat dilakukan oleh sistem perangkat lunak , perilaku apa yang akan diperlihatkan oleh perangkat lunak, antarmuka apa yang

didefiniskan dan batasan – batasan apa yang diterapkan. Model – model analisis untuk mendapatkan spesifikasi – spesifikasi kebutuhan perangkat lunak pada dasarnya harus mencapai 3 sasaran utama yaitu :

1. Untuk mendeskripsikan apa yang pelanggan inginkan
2. Menetapkan dasar bagi perancangan perangkat lunak
3. Untuk mendefinisikan sejumlah kebutuhan yang dapat divalidasi saat perangkat lunak dikembangkan.

Model analisis juga pada dasarnya harus bisa menjabatani kesenjangan – kesenjangan yang terjadi diantara deskripsi – deskripsi berperingkat sistem yang mendeskripsikan fungsionalitas – fungsionalitas sistem atau bisnis secara keseluruhan saat ia dicapai dengan menerapkan solusi – solusi perangkat lunak, perangkat keras, manusia dan elemen sistem yang lainnya, dengan perancangan perangkat lunak yang akan mendeskripsikan arsitektur perangkat lunak antar muka pengguna dan struktur peringkat komponen. Merupakan hal yang penting untuk melihat bahwa semua elemen yang ada dalam model spesifikasi kebutuhan akan dapat dilacak langsung ke bagian – bagian yang ada dalam model perancangan.

5.1.2. Aturan Analisis

Arlow dan Neustadt dalam (Roger S. Pressman, 2012) menyarankan sejumlah aturan yang seharusnya dipatuhi saat analis sistem membuat model – model analisis .

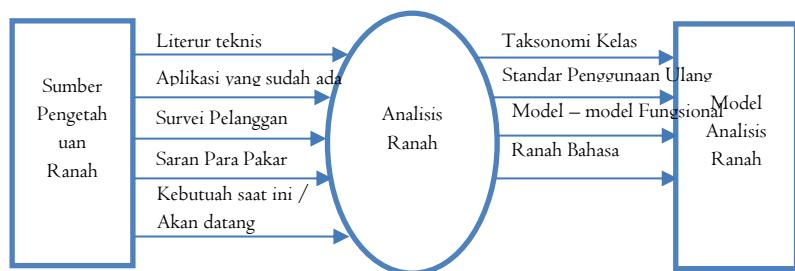
1. Model seharusnya berfokus pada kebutuhan – kebutuhan yang tampak di dalam ranah permasalahan atau ranah bisnis.
2. Masing – masing elemen yang ada di dalam model kebutuhan seharusnya menambahkan suatu pemahaman keseluruhan pada spesifikasi kebutuhan perangkat lunak dan menyediakan pendangan sekilas pada ranah informasi, fungsi dan prilaku sistem.
3. Tunda pertimbangan – pertimbangan yang berkait dengan infrastruktur dan model – model non – fungsional lainnya hingga tahap perancangan dimulai.
4. Minimalkan saling ketergantungan dalam sistem.
5. Upayakan agar analis sistem merasa pasti bahwa model – model kebutuhan memberikan nilai tertentu pada semua stakeholder.
6. Usahan agar model sesederhana mungkin.

5.1.3. Analisis Ranah

Firesmith dalam (Roger S. Pressman, 2012) mendeskripsikan analisis ranah sebagai proses mengidentifikasi , menganalisis dan menspesifikasi kebutuhan – kebutuhan umum dari suatu ranah aplikasi

yang sifatnya spesifik, agar suatu saat dapat digunakan pada berbagai proyek lain yang ada pada ranah aplikasi yang sama yang merupakan identifikasi , analisis, dan spesifikasi kemampuan – kemampuan yang bersifat umum dan dapat digunakan ulang dalam ranah aplikasi yang sifatnya spesifik, dalam artian objek – objek , kelas – kelas , bagian – bagian dan kerangka – kerangka kerja, yang bersifat umum dari suatu aplikasi yang bersifat spesifik.

Peran seorang analis ranah adalah untuk menyimpulkan dan mendefinisikan pola – pola analisis , kelas – kelas analisis, dan informasi yang terkait mungkin digunakan oleh orang – orang yang bekerja dengan cara yang serupa tetapi tidak harus berada pada aplikasi yang sama. Berikut gambaran analisis ranah.



Gambar 5.1 Input , Output untuk analisis ranah

5.1.4. Pendekatan – pendekatan untuk Pemodelan Spesifikasi Kebutuhan

Ada dua pendekatan untuk memodelkan spesifikasi kebutuhan yaitu :

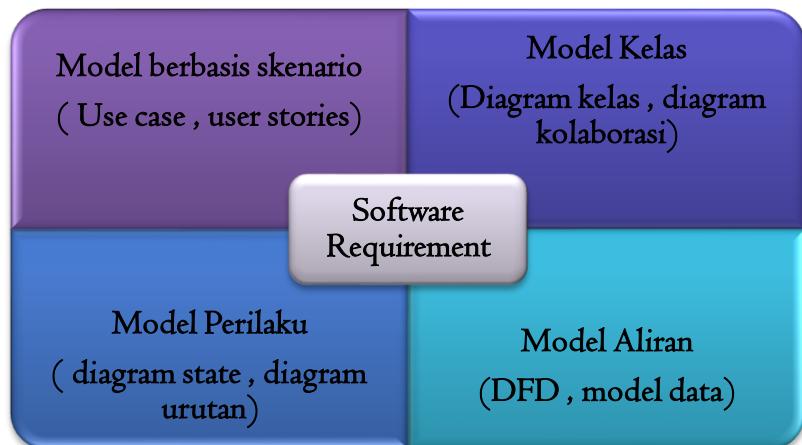
I. Analisis terstruktur

Analisis terstruktur memperlakukan data dan proses yang melakukan transformasi data tersebut sebagai entitas yang terpisah. Objek – objek data dimodelkan dengan cara mendefinisikan atribut – atributnya serta relasi – relasinya. Proses – proses yang dimanipulasi objek – objek data dimodelkan sedemikian rupa sehingga mereka dapat memperlihatkan bagaimana caranya mereka melakukan transformasi data saat objek – objek data mengalir di dalam sistem yang akan dikembangkan.

2. Analisis berorientasi objek

Pemodelan ini berfokus pada pendefinisian kelas – kelas dan cara bagaimana mereka saling bekerjasama satu dengan lainnya untuk memenuhi kebutuhan para pelanggan. UML (*Unified Modeling Process*) dan *Unified Process* merupakan perkakas dan proses yang bernuansa berorientasi objek.

Dua pendekatan tersebut memunculkan pilihan mana yang terbaik, walaupun pengembang perangkat lunak kebanyakan memilih dari salah satu pendekatan tersebut. Pengembang tidak seharusnya bertanya mana yang terbaik tetapi pengembang seharunya berfikir bagaimana kombinasi dari representasi pendekatan tersebut yang menyediakan bagi para stakeholder model spesifikasi kebutuhan yang terbaik dan yang merupakan jembatan yang paling efektif ke tahapan perancangan lunak.



Gambar 5.2 Elemen – elemen model analisis

Masing – masing unsur model kebutuhan memperlihatkan permasalahan dari sudut pandang yang berbeda. Elemen – elemen berbasis skenario memperlihatkan bagaimana interaksi yang kelak akan terjadi antara pengguna dengan perangkat lunak yang

akan dikembangkan dan juga memperlihatkan sejumlah aktivitas berurutan yang bersifat spesifik yang terjadi saat perangkat lunak digunakan. Elemen model berbasis kelas memodelkan objek – objek yang akan dimanipulasi oleh sistem , memodelkan operasi – operasi yang akan diterapkan pada objek – objek untuk melakukan manipulasi, memodelkan relasi yang terjadi antara objek satu dengan yang lainnya, dan memodelkan kolaborasi yang terjadi di antara kelas – kelas yang telah didefinisikan. Elemen – elemen perilaku memperlihatkan bagaimana event – event eksternal melakukan perubahan pada keadaan sistem atau kelas – kelas yang ada didalamnya. Terakhir elemen – elemen berorientasi aliran memperlihatkan perangkat lunak yang bertindak sebagai perilaku transformasi informasi , memperlihatkan bagaimana objek – objek data di transformasi saat mereka mengalir melintasi fungsi yang dimiliki sistem.

5.2. Pemodelan

5.2.I. Pemodelan Berbasis Skenario

Pemodelan berbasis skenario merupakan suatu cara untuk memahami para pengguna berinteraksi dengan perangkat lunak. Dengan memahami hal tersebut tim pengembang perangkat lunak dapat memahami kebutuhan pengguna dan melakukan analisis untuk melakukan pemodelan perancangan yang baik. Pemodelan spesifikasi kebutuhan pengguna menggunakan UML diawali dengan membentuk use

case – use case , diagram – diagram aktivitas dan diagram – diagram swimlane.

I. Membuat Use Case Awal

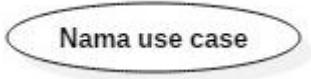
Cara kerja dari use case pada dasarnya adalah menangkap interaksi yang terjadi antara produsen informasi , pengguna informasi dengan perangkat lunak (Roger S. Pressman, 2012). Secara garis besar use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi – fungsi dan fitur – fitur tersebut (Rosa A. S, 2018).

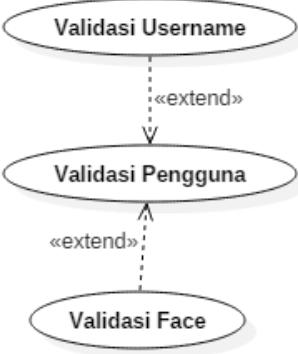
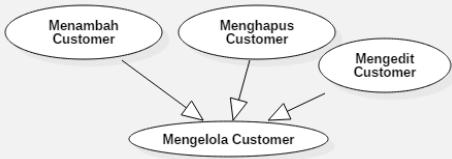
Syarat penamaan pada use case adalah nama didefinisikan sesederhana mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu (Rosa A. S, 2018) :

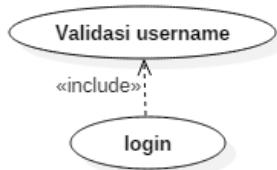
- a. Actor adalah orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat sendiri , jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
- b. Use case merupakan fungsionalitas yang disediakan sistem sebagai unit – unit yang saling bertukar pesan antar unit atau aktor

Berikut notasi – notasi atau simbol – simbol yang ada pada diagram use case (Rosa A. S, 2018):

Tabel 5.1 Simbol pada use case

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit – unit yang saling bertukar pesan antara unit atau aktor , di nyatakan dengan menggunakan kata kerja.
Aktor / actor  Nama Aktor	Orang , proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat sendiri , jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang
Asosiasi / association 	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor 
Ekstensi / extend	Relasi use case tambahan ke sebuah use case di mana use case yang ditambahkan dapat berdiri sendiri walaupun tanpa use case tambahan itu,

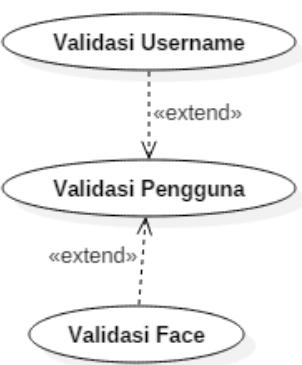
<p>«extend»</p> <p>-----></p>	<p>seperti prinsip inheritance pada pemrograman berorientasi objek.</p>  <pre> graph TD V1([Validasi Username]) -- "«extend»" --> V2([Validasi Pengguna]) V2 -- "«extend»" --> V3([Validasi Face]) </pre>
<p>Generalisasi / generalization</p> <p>→</p>	<p>Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya .</p>  <pre> graph TD V1([Menambah Customer]) --> V4([Mengelola Customer]) V2([Menghapus Customer]) --> V4 V3([Mengedit Customer]) --> V4 </pre>

<p>Menggunakan / Include</p> <p>«include»</p> 	<p>Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya sebagai syarat dijalankan use case ini</p> <p>Ada dua perspektif dalam penggunaan include dalam use case :</p> <ol style="list-style-type: none"> a. Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan .  <p>b. Include berarti use case yang ditambahkan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan.</p> 

	Kedua perspektif dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.
--	---

Use case ini nantinya akan menjadi kelas proses pada diagram kelas sehingga perlu dipertimbangkan penamaan yang dilakukan. Berikut aturan perubahan use case yang menjadi kelas proses (Rosa A. S, 2018).

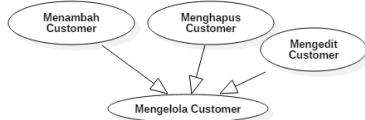
Tabel 5.2 Perubahan Use case menjadi kelas proses

Hubungan	Deskripsi
Extensi / extend 	Pada hubungan extend dapat diambil use case induknya untuk

dihasilkan kelas dengan metode berupa use case extend.

```
class ValidasiUsername {  
    //atribut  
    .....  
  
    procedure  
  
        validasiUserName()  
{  
    //proses  
    .....  
}  
  
    procedure  
  
        validasiFace()  
    //proses  
    .....  
}
```

Generalisasi/generalization



Pada hubungan generalisasi maka dapat hanya diambil use case umumnya dijadikan kelas dengan metode berupa use case khususnya.

```

class MengelolaCustomer {
//atribut
.....
procedure
MenambahCustomer()
//proses
.....
}

procedure
MenghapusCustomer()
//proses
.....
}
procedure
MengeditCustomer
.....

```

Use case yang berdiri sendiri



login

Metode yang mungkin bisa ada di dalam kelas proses login adalah sebagai berikut :

```
class Login {  
    //atribut  
    .....  
  
    procedure  
    Login(){  
        //proses  
        .....  
    }  
  
    procedure  
    Logout() {  
        //proses
```

Setiap use case di lengkapi dengan skenario use case . Skenario use case adalah flow jalannya proses use case dari sisi aktor dan sistem. Berikut adalah contoh format tabel dari skenario use case (Victoria University of Wellington, 2018):

Use Case Body

Title: Reserve books

Summary: This use case describes the interactions between a *Book borrower* and the library system when reserving books.

Actors: Book borrower (primary), Library Information System (secondary)

Creation date: 13/08/14

Modification date: 03/08/15

Version: 2.0

Person in charge: Noel Nelson

Main Success Scenario:

<u>Actions of actors:</u>	<u>Actions of system:</u>
1. Book borrower navigates to library website 2. Book borrower enters search parameters for book they wish to reserve 4. Book borrower selects book to reserve 6 Library IS confirms availability 8. Book borrower enters details 10. Library IS confirms details	3. System returns search results 5. System checks availability 7. System requests log in 9. System requests authentication from Library IS 11. System confirms log in, places book on reserve, and updates catalogue information 12. System notifies user

Alternative Scenarios:

A1: Username and/or password incorrect

The A1 scenario starts at point 9 of the main success scenario

<u>Actions of actors:</u>	<u>Actions of system:</u>
10. Library IS rejects username and/or password	11. Go to M.6

Error Scenarios:

E1: Book not available to reserve

The E1 scenario starts at point 5 of the main success scenario

<u>Actions of actors:</u>	<u>Actions of system:</u>
6. Library IS says book not available	7. System notifies user that book is not available to reserve at this time. The Use Case fails.

Gambar 5.3 Contoh Skenario Use Case

Tipe use case

Use case dapat dijelaskan baik pada tingkat abstrak (dikenal sebagai use case bisnis proses) atau pada tingkat implementasi khusus (dikenal sebagai use case sistem). Masing-masing dijelaskan secara lebih rinci di bawah ini (Inflectra, 2018) :

- a. Business Use Case - Juga dikenal sebagai "use case tingkat abstrak", kasus penggunaan ini ditulis dengan cara agnostik teknologi, hanya mengacu pada proses bisnis tingkat tinggi yang dijelaskan (misalnya "pengembalian buku") dan berbagai eksternal entitas (juga dikenal sebagai aktor) yang mengambil bagian dalam proses (misalnya "peminjam", "pustakawan", dll.). Kasus penggunaan bisnis akan menentukan urutan tindakan yang perlu dilakukan oleh bisnis untuk memberikan hasil yang berarti dan dapat diobservasi ke entitas eksternal.
- b. System Use Case - Juga dikenal sebagai "use case implementasi", kasus penggunaan ini ditulis pada tingkat detail yang lebih rendah daripada use case bisnis dan mengacu pada proses khusus yang akan dilakukan oleh bagian-bagian yang berbeda dari sistem. Misalnya kasus penggunaan sistem mungkin

"mengembalikan buku ketika terlambat" dan akan menggambarkan interaksi dari berbagai aktor (peminjam, pustakawan) dengan sistem dalam melaksanakan proses end-to-end.

2. Model – model UML yang melengkapi Use Case

a. Diagram Aktivitas

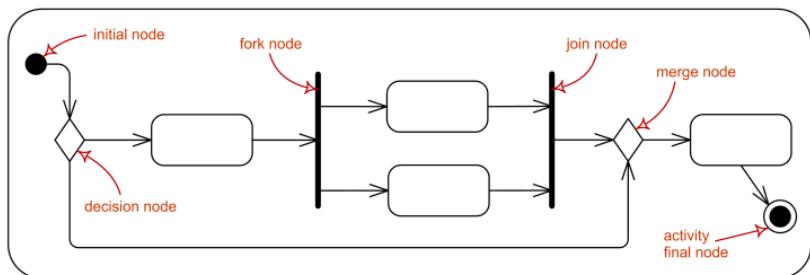
Diagram aktivitas (activity diagram) yang disediakan oleh UML melengkapi use case yang telah dibuat sebelumnya dengan memberikan representasi grafis dan aliran – aliran interaksi di dalam suatu skenario yang sifatnya spesifik (Roger S. Pressman, 2012).

Diagram aktivitas menggambarkan diagram alir (flowchart) dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Pada intinya diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dilakukan sistem. Diagram aktivitas dapat digunakan untuk mendefinisikan hal – hal berikut (Rosa A. S, 2018) :

- I) Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

- 2) Urutan atau pengelompokan tampilan dari sistem atau user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
- 3) Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
- 4) Rancangan menu yang ditampilkan pada perangkat lunak.

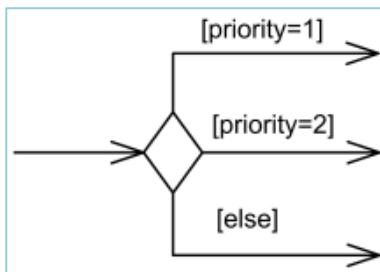
Berikut adalah notasi – notasi atau simbol – simbol yang digunakan pada diagram aktivitas (uml-diagrams.org, UML Activity Diagram Controls, 2018):



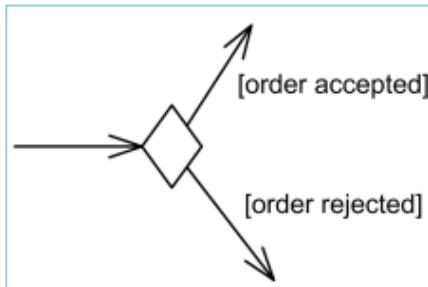
Gambar 5.4 Simbol Activity Diagram

- I) Initial Node merupakan node kontrol dimana aliran aktivitas dimulai

- 2) Decision Node merupakan node kontrol yang menerima token pada satu atau dua sisi yang masuk dan memilih satu ujung keluar dari satu atau lebih aliran keluar. Decision node diperkenalkan di UML untuk mendukung aktivitas kondisional. Contoh decision node :



Gambar 5. 5 Decision tiga cabang



Gambar 5.6 Decision node cabang tiga

- 3) Fork Node, merupakan node kontrol yang memiliki satu tepi masuk dan tepi keluar ganda dan digunakan untuk membagi aliran masuk menjadi beberapa aliran bersamaan. Fork node diperkenalkan untuk mendukung paralelisme dalam aktivitas.
- 4) Join Node ,merupakan node yang memiliki beberapa tepi masuk dan satu ujung keluar dan digunakan untuk mensinkronkan aliran bersamaan yang datang. Join node diperkenalkan untuk mendukung paralelisme dalam kegiatan.
- 5) Merge Node, merupakan node kontrol yang menyatukan beberapa aliran alternatif yang masuk untuk menerima aliran keluar tunggal. Tidak ada token yang bergabung. Penggabungan tidak boleh untuk menyingkronkan aliran bersamaan.
- 6) Activity Final Node , merupakan node kontrol akhir yang menghentikan semua aliran dalam suatu aktivitas.

b. Diagram Swimlane

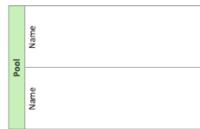
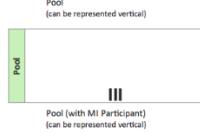
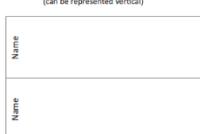
Diagram swimlane merupakan variasi dari diagram aktivitas yang memungkinkan untuk melihat aliran aktivitas – aktivitas yang dideskripsikan oleh use case pada saat yang sama memperlihatkan aktor mana yang melakukan kegiatan. Diagram Swimlane menggabarkan bagaimana berbagai aktir memanggil fungsi – fungsi tertentu yang sifatnya spesifik sehingga sesuai dengan kebutuhan – kebutuhan perangkat lunak. untuk notasi dari diagram swimlane sama dengan diagram aktivitas hanya ditambahkan swimlane yaitu container yang memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Swimlane diagram , sering disebut juga “Deployment Process Map” atau “Cross Functional Flowchart” adalah sebuah diagram yang merepresentasikan flow proses yang menggambarkan interaksi dari beberapa bagian yang berbeda dan bagaimana perkembangan proses melelui beberapa phase yang berbeda (binus, 2014).

Elemen – elemen yang terdapat dalam swimlane diagram adalah : (a) Process: Aktual proses dan flow , (b) Actors:

Orang, groups, teams, yang melakukan tahapan proses. (c) Phases: Menunjukkan tahapan dari project. (d) Symbols: Simbol fisik yang digunakan menggambarkan apa yang terjadi pada setiap tahapan dari proses (binus, 2014).

Pools and Lanes



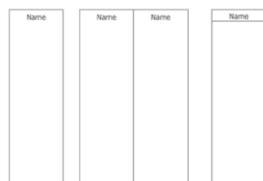
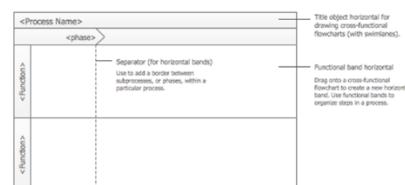
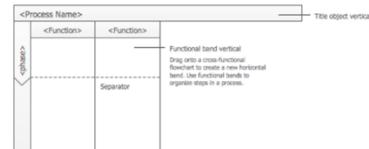
Flows

Message Flow →

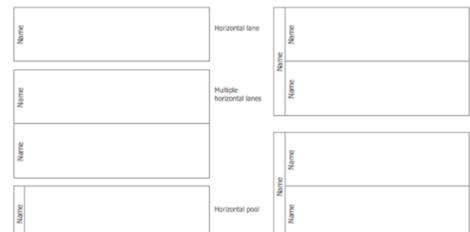
Smart Message Flow ↗

Initiating Message Flow with Decorator

Non-Initiating Message Flow with Decorator



Vertical pool



Horizontal pool with two lanes

Vertical pool with two lanes



Horizontal pool with multiple lanes

Vertical pool with multiple lanes

Gambar 5.7 Business Process Elements:
Swimlanes

Sumber : (conceptdraw, 2018)

5.2.2. Pemodelan Berbasis Kelas

Pemodelan berbasis kelas pada dasarnya memperlihatkan objek – objek yang dimanipulasi oleh perangkat lunak, memperlihatkan operasi – operasi yang akan diterapkan pada objek – objek untuk menghasilkan imbas tertentu pada manipulasi objek, memperlihatkan relasi – relasi antar objek serta memperlihatkan kolaborasi – kolaborasi yang terjadi di antara kelas – kelas yang didefinisikan. Elemen – elemen model berbasis kelas mencakup didalamnya elemen – elemen kelas dan objek – objek , atribut – atribut , operasi – operasi , model tanggung jawab kelas (Class Responsibility Collaborator – CRC) , diagram – diagram koaborasi dan peket – paket (Roger S. Pressman, 2012).

Ada beberapa karakteristik objek yaitu state (merupakan suatu kondisi atau keadaan dari objek yang mungkin ada, status dari objek akan berubah setiap waktu dan ditentukan oleh sejumlah properti dan relasi dengan objek lain), Behavior atau sifat (menentukan bagaimana objek merespon permintaan dari objek lain dan melambangkan setiap hal yang dapat dilakukan, sifat ini diimplementasikan dengan sejumlah operasi untuk objek), identity (setiap objek yang ada dalam suatu sistem adalah unik).

I. Identifikasi Kelas – kelas Analisis

Untuk mengidentifikasi kelas – kelas , dapat dimulai dengan melakukan proses identifikasi kelas dengan cara memeriksa skenario penggunaan perangkat lunak yang telah dikembangkan sebelumnya sebagai bagian dari model – model kebutuhan dan kemudian melakukan pemisahan berdasarkan tata bahasa pada usecase yang dikembangkan untuk perangkat lunak. kelas – kelas dapat ditentukan dengan cara menggarisbawahi setiap kata benda dan memasukkannya ke sebuah tabel sederhana. Kelas – kelas analisis pada umumnya memanifestasikan dirinya dalam beberapa cara sebagai berikut (Roger S. Pressman, 2012):

- a. Entitas – entitas eksternal yang menghasilkan atau menggunakan informasi yang akan digunakan perangkat lunak berbasis komputer.
- b. Sesuatu yang merupakan bagian dari ranah informasi untuk permasalahan.
- c. Kehadiran atau event yang terjadi di dalam konteks operasi sistem.
- d. Peran – peran yang dimainkan oleh orang – orang yang berinteraksi dengan perangkat lunak.
- e. Unit – unit organisasional yang relevan untuk perangkat lunak yang akan dikembangkan.

- f. Tempat yang meletakkan konteks tertentu bagi sistem dan fungsi keseluruhan sistem.
- g. Struktur yang mendefiniskan suatu kelas objek – objek

Kelas – kelas analisis dapat didefinisikan ketika awal pemodelan dengan memisahkan kata benda dalam narasi pemrosesan use case yang telah didapat. Dengan mengekstrasi kata benda – kata benda tersebut maka didapat kelas potensial. Dari kelas potensial tersebut maka diidentifikasi karakteristik dari kelas potensial tersebut.

Pada dasarnya langkah pertama dari pemodelan berbasis kelas adalah membuat definisi – definisi kelas dan keputusan – keputusan yang harus dibuat.

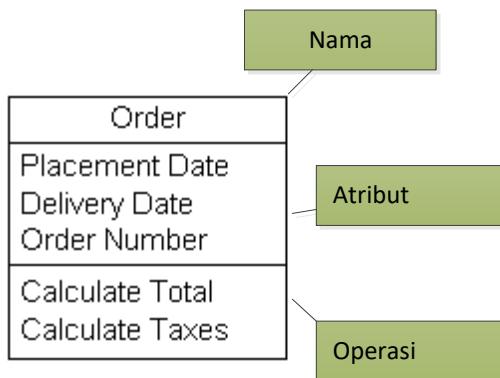
Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasikeadaan tersebut(metoda/fungsi). (Desy, class Diagram, 2018)

Dapat disimpulkan bahwa untuk menemukan class adalah kelompokan kata benda pada dokumentasi use case kemudian kategorikan

kata benda tersebut, setiap objek yang ditemukan pada kata benda yang ditemukan merupakan contoh dari beberapa class. Class memiliki tiga area pokok yaitu nama atau stereotype , atribut dan metoda atau operasi. (Desy, class Diagram, 2018)

Atribut dan operasi dapat memiliki salah satu yaitu (Desy, class Diagram, 2018):

- a. Private , tidak dapat dipanggil dari luar class yang bersangkutan
- b. Protected , hanya dapat dipanggil oleh class yang bersangkutan dan anak – anak yang mewarisiinya.
- c. Public, dapat dipanggil oleh siapa saja.



Gambar 5.8 Struktur class

2. Menentukan Atribut – atribut

Atribut mendefinisikan kelas – kelas yang telah dipilih untuk dimasukkan dalam model

spesifikasi kebutuhan perangkat lunak . Atribut – atribut merupakan sesuatu yang mendefinisikan kelas yaitu yang mengklasifikasi makna suatu kelas dalam konteks ruang permasalahan yang telah diketahui sebelumnya. Untuk mengembangkan sejumlah atribut yang bermakna untuk kelas analisis maka perlu diperlajari terlebih dahulu masing – masing use case dan memilih sesuatu yang cukup beralasan untuk dimiliki suatu kelas (Roger S. Pressman, 2012).

Sebuah class mungkin memiliki beberapa atribut, Atribut merepresentasikan beberapa properti dari sesuatu yang dimodelkan. (Desy, class Diagram, 2018)

3. Mengdefinisikan Operasi – operasi
Operasi – operasi pada dasarnya mendefinisikan perilaku suatu objek. Secara umum operasi – operasi dapat dikategorikan menjadi empat yaitu (Roger S. Pressman, 2012):
 - a. Operasi yang melakukan manipulasi data dengan cara – cara tertentu misalnya (menambah, menghapus, melakukan performatan ulang, memilih)
 - b. Operasi yang melaksanakan komputasi
 - c. Operasi yang mencoba menjawab tentang keadaan(state) suatu objek
 - d. Operasi yang melakukan pemantauan terhadap suatu objek dengan maksud

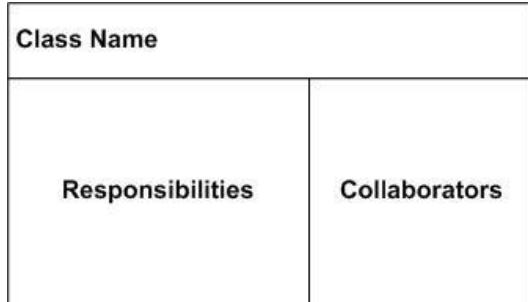
melakukan pengendalian atas objek yang bersangkutan.

Suatu operasi harus memiliki pengetahuan alamaiah tentang atribut – atribut dan asosiasi – asosiasi suatu kelas.

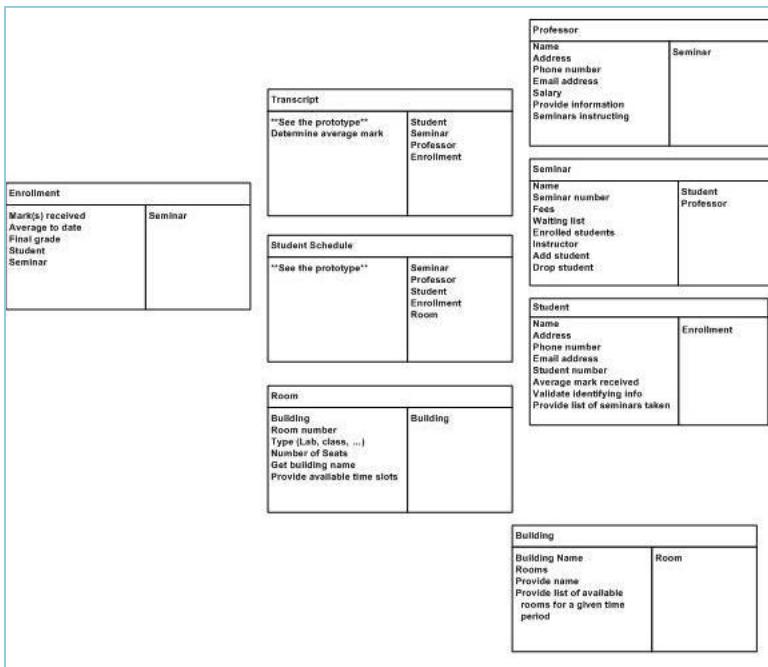
Operasi adalah abstraksi dari segala sesuatu yang dapat dilakukan pada sebuah objek dan berlaku untuk semua objek yang terdapat dalam class tersebut. Class mungkin memiliki beberapa operasi, biasanya pemanggilan operasi pada sebuah objek akan mengubah data atau kondisi dari objek tersebut.

4. Pemodelan CRC (*Class-Responsibility-Collaborator*)

Pemodelan CRC menyediakan cara untuk mengidentifikasi dan mengorganisasi kelas – kelas yang relevan untuk sistem atau relevan untuk menspesifikasi – menspesifikasi kebutuhan produk. Menurut Ambler dalam (Roger S. Pressman, 2012) mengemukaakan bahwa suatu model CRC merupakan kumpulan kartu – kartu yang masing – masing merepresentasikan kelas – kelas. Kartu CRC terbagi menjadi tiga bagian yaitu nama kelas, mendaftar tanggung jawab kelas, mendaftar kolaborator-kolaboratornya.



Gambar 5.9 CRC Card



Gambar 5.10 Contoh CRC Model

Sumber : (Ambler, 2018)

5. Menggambar Class

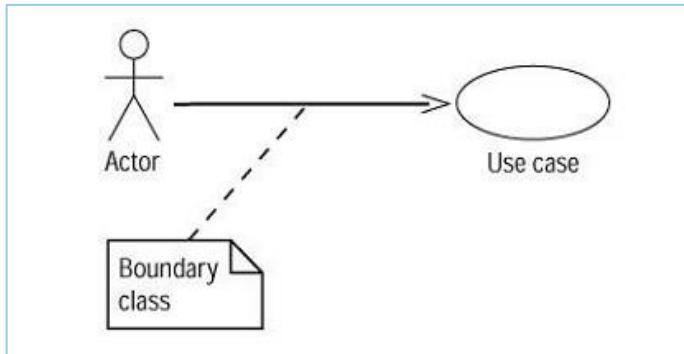
Ketika menggambarkan sebuah class, kita tidak perlu menampilkan seluruh atribut atau operasi , kerana sebagian kasus kita tidak dapat menampilkannya dalam sebuah gambar , karena terlalu banyaknya atribut atau operasinya, bahkan terkadang tidak perlu karena kurang relevannta atribut atau operasi yang akan ditampilkan. Sehingga kita dapat menampilkan atribut dan opersinya hanya sebagain . Kosongnya tempat pengisian bukan berarti tidak ada, karena itu dapat menambahkan tanda (...) pada akhir daftar yang menunjukkan bahwa masih ada atribut atau operasi yang lain.

6. Stereotype Class

Stereotype adalah sebuah mekanisme yang digunakan untuk mengkategorikan sebuah class. Sehingga memudahkan kita dalam mengorganisasikan reponsibility dari tiap – tiap class. Terdapat tiga stereotype utama dalam UML yaitu boundary ,entity dan control .

a. Boundary class adalah class yang terdapat batasan sistem dan dunia nyata, hal ini mencakup semua form, hardware interface . Boundary class dapat diidentifikasi dari use case diagram . Minimal terdapat satu buah boundary class dalam relasi actor dengan usecase.

Boundary class adalah yang mengakomodasi interaksi antara actor dengan sistem.

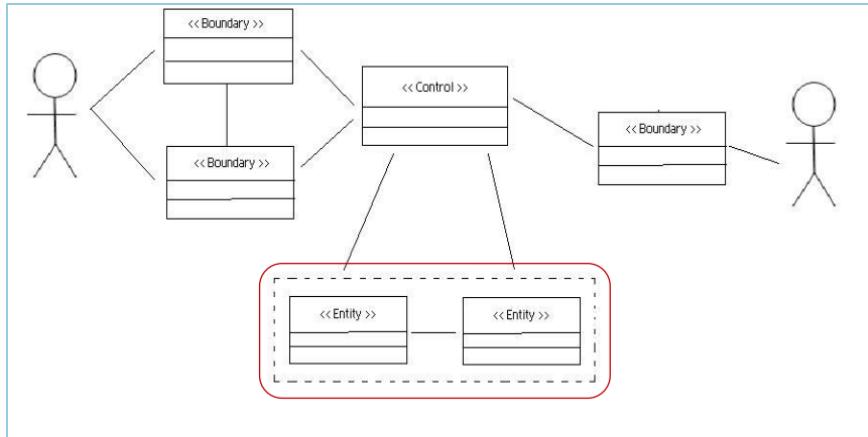


Gambar 5.II Boundary Class

b. Entity Class

Entity class menyimpan informasi yang mungkin akan disimpan ke sebuah storage . Class dengan stereotype entity dapat dipemukau di flow of event (scenario dari use case diagram) dan interaction diagram. Entity calss dapat di identifikasi dengan mencari kata benda yang ada pada flow of events . Selain itu, dapat juga diidentifikasi dari struktur database (dilihat dari nama – nama tabelnya). Sebuah entity class mungkin perlu dibuat untuk sebuah tabel. Bila sebuah tabel menyimpan informasi secara permanen, maka entity class akan

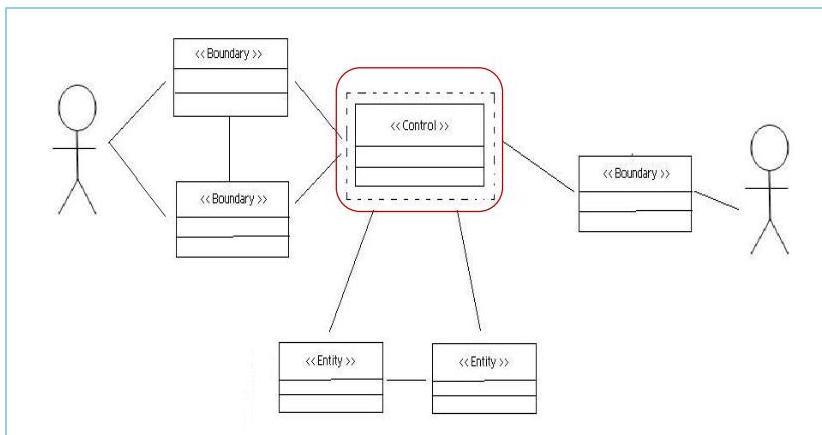
menyimpan informasi pada memory ketika sistem sedang running.



Gambar 5.12 Entity Class

c. Control Class

Control class bertanggung jawab dalam mengatur kelas – kelas yang lain. Control class juga bertanggung jawab dalam mengetahui dan menyampaikan business rule dari sebuah organisasi. Class ini menjalankan alternate flow dan mampu mengatasi error . karena alasan ini controll class sering disebut manager class.



Gambar 5.13 Control Class

7. Relationship

Relasi atau relationship menghubungkan beberapa objek sehingga memungkinkan terjadinya interaksi dan kolaborasi diantara objek – objek yang terhubung. Dalam pemodelan class diagram, terdapat tiga buah relasi utama yaitu association, aggregation dan generalization.

a. Asosiasi

Relasi asosiasi merupakan relasi struktural yang menspesifikasikan bahwa satu objek terhubung dengan objek lainnya. Relasi ini tidak menggambarkan aliran data, sebagaimana yang terdapat pada pemodelan desain pada analisa terstruktur. Relasi asosiasi dapat dibagi menjadi dua jenis yaitu :

- 1) Uni-directional association dan bi-directional association



Objek supir memiliki uni-directional association dengan objek taxi. Relasi uni-directional diatas memungkinkan objek supir untuk memanggil properti dari objek taxi. Namun tidak berlaku sebaliknya . Objek pesawat tidak dapat mengakses properti dari objek supir.

- 2) Bi-directional



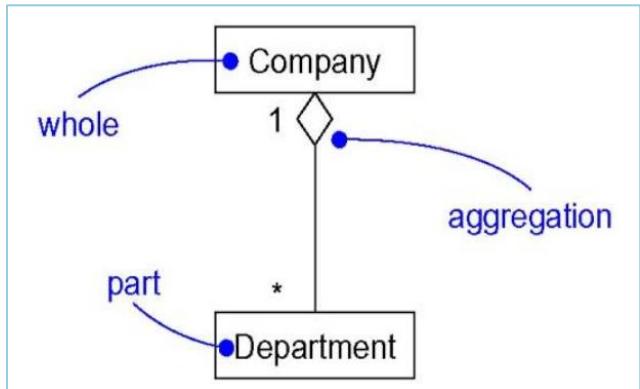
Objek supir dapat memanggil properti yang dimiliki oleh objek taksi. Begitu juga sebaliknya, objek taksi dapat memanggil properti dari objek supir

Hubungan association mempunyai dua titik. Salah satu titik bisa memiliki label untuk menjelaskan association tersebut. Panah association menggambarkan arah

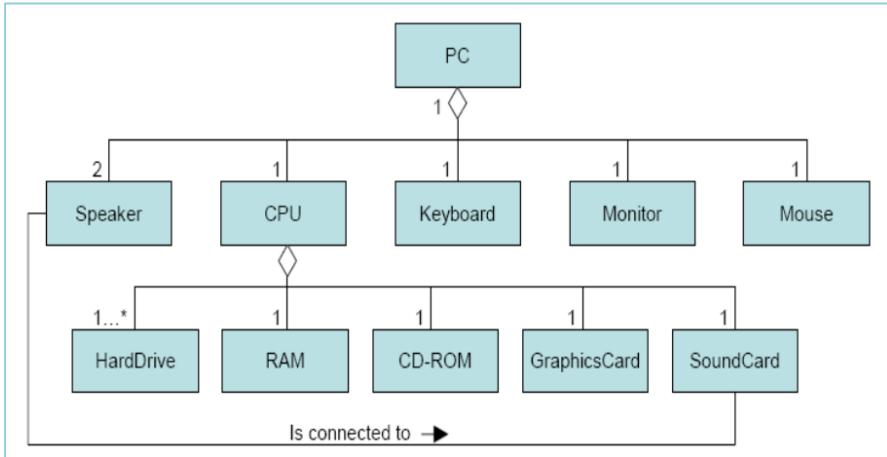
mana association dapat di transfer atau disusun.

b. Aggregation

Aggregation merupakan bentuk khusus dari asosiasi dimana induk terhubung dengan bagian – bagiannya. Aggregation merepresentasikan relasi “has-a”, artinya sebuah class memiliki atau terdiri dari bagian yang lebih kecil. Dalam UML, relasi aggregasi digambarkan dengan open diamond pada sisi yang menyatakan induk (whole).



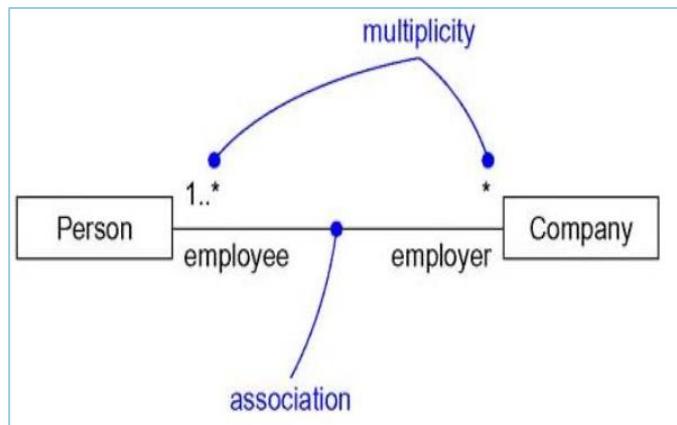
Gambar 5.14 Contoh Aggregation



Gambar 5.15 Contoh Aggregation dari
PC

Multiplicity

Multiplicity menentukan atau mendefinisikan banyaknya objek yang terhubung dalam suatu relasi. Indikator multiplicity terdapat pada masing – masing akhir garis relasi baik pada asosiasi.



Gambar 5.16 Contoh Multiplicity

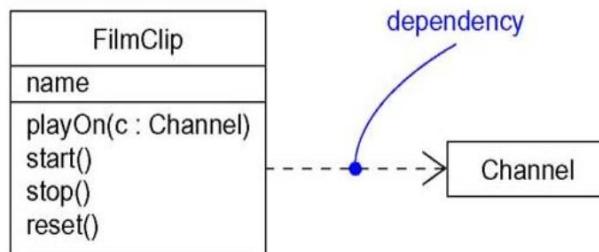
Multiplicity dari suatu titik association adalah angka kemungkinan bagian dari hubungan kelas dengan single instance(bagian) pada titik yang lain.

Tabel 5.1 Multiplicity

Multiplicity	Artinya
0..1	Nol atau satu bagian. Notasi n..m menerangkan n sampai m bagian
0..* atau *	Tak hingga pada jangkauan bagian (termasuk kosong)
1	Tepat satu bagian
1..*	Sedikitnya hanya satu bagian

c. Dependency

Dependency merupakan sebuah relasi yang menyebutkan bahwa perubahan pada satu class, maka akan mempengaruhi class lain yang menggunakannya, tetapi tidak berlaku sebaliknya. Pada umumnya relasi dependency dalam konteks Class Diagram digunakan apabila terdapat satu class yang menggunakan class lain sebagai argumen dari sebuah method.



Gambar 5.17 Contoh dependency

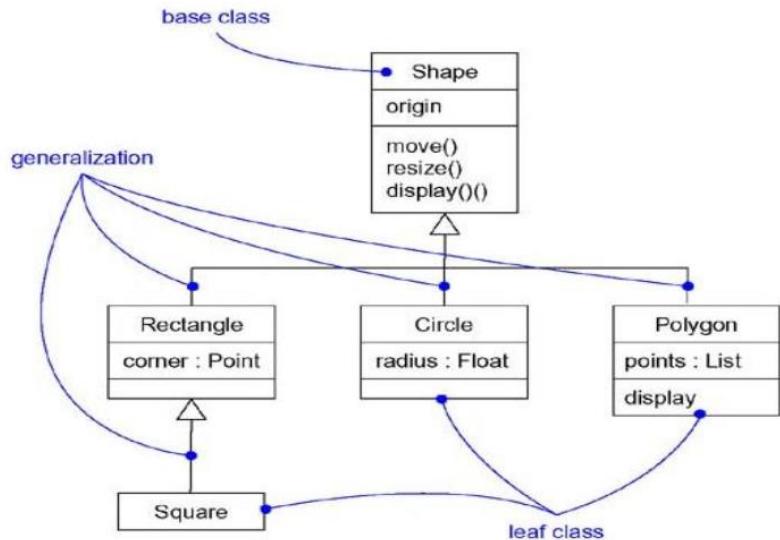
Inheritance

Inheritance merupakan salah satu karakteristik dalam pemrograman berorientasi objek, dimana class mewarisi /inherit sifat – sifat (dalam hal ini atribut dan operasi) dari class lain yang merupakan parent dari class tersebut. Class yang menurunkan sifat – sifatnya disebut superclass sedangkan class yang

mewarisi sifat dari superclass disebut subclass. Inheritance disebut juga hierarki “is-a” (adalah sebuah) atau “kind-of” (sejenis). Subclass dapat memiliki atau menggunakan atribut dan operasi tambahan yang hanya berlaku pada tingkat hierarkinya. Karena inheritance relationship bukan merupakan relationship diantara objek yang berbeda , maka relationship ini tidak diberi nama. Begitu pula dengan penamaan multiplicity.

d. Generalization

Generalisasi merupakan relasi antar class dengan makan generalisasi-spesialisasi (umum – khusus).



Gambar 5.18 Contoh Generalization

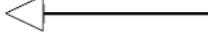
8. Simbol – simbol dalam Class Diagram

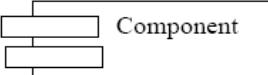
Berikut adalah simbol – simbol yang digunakan dalam menggambar Class Diagram.

Tabel 5.2 Simbol – simbol dalam class diagram

SIMBOL	NAMA	KETERANGAN
<div style="border: 1px solid black; padding: 10px; width: fit-content;"> Site Config +sqlIDNS:string +Adminemail:String </div>	Class	<p><i>Class</i> adalah blok - blok pembangun pada pemrograman berorientasi obyek.</p> <p>Sebuah class digambarkan sebagai sebuah kotak yang</p>

		terbagi atas 3 bagian. Bagian atas adalah bagian nama dari <i>class</i> . Bagian tengah mendefinisikan property/atribut <i>class</i> . Bagian akhir mendefinisikan method method dari sebuah <i>class</i> .
<u>1..n Owned by 1</u>	Assosiation	Sebuah asosiasi merupakan sebuah <i>relationship</i> paling umum antara 2 <i>class</i> , dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 <i>class</i> . Garis ini bisa melambangkan tipe-tipe <i>relationship</i> dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah <i>relationship</i> (Contoh: One-to-one, one-to-many, many-to-many).
	Composition	Jika sebuah <i>class</i> tidak bisa berdiri sendiri dan harus merupakan bagian dari <i>class</i> yang lain, maka <i>class</i> tersebut memiliki relasi <i>Composition</i> terhadap <i>class</i> tempat dia bergantung tersebut. Sebuah <i>relationship composition</i> digambarkan sebagai garis

		dengan ujung berbentuk jajaran genjang berisi/solid.
	Dependency	Kadangkala sebuah <i>class</i> menggunakan <i>class</i> yang lain. Hal ini disebut <i>dependency</i> . Umumnya penggunaan <i>dependency</i> digunakan untuk menunjukkan operasi pada suatu <i>class</i> yang menggunakan <i>class</i> yang lain. Sebuah <i>dependency</i> dilambangkan sebagai sebuah panah bertitik-titik.
	Aggregation	<i>Aggregation</i> mengindikasikan keseluruhan bagian <i>relationship</i> dan biasanya disebut sebagai relasi “mempunyai sebuah” atau “bagian dari”. Sebuah <i>aggregation</i> digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi/tidak solid.
	Generalization	Sebuah relasi <i>generalization</i> sepadan dengan sebuah relasi <i>inheritance</i> pada konsep berorientasi obyek. Sebuah <i>generalization</i> dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid

		yang mengarah ke kelas “parent”-nya/induknya.
	komponen	Sebuah komponen melambangkan sebuah entitas software dalam sebuah sistem. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel disebelah kirinya.
	Depedency	Sebuah Dependency digunakan untuk menotasikan relasi antara dua komponen. Notasinya adalah tanda panah putus-putus yang diarahkan kepada komponen tempat sebuah komponen itu bergantung.

Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosial , dan lain-lain . Class diagram memberikan pandangan secara luas dari suatu sistem dengan menunjukkan kelas – kelasnya dan hubungan mereka. Diagram class bersifat statis yaitu menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan. Class diagram dapat membantu memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe

diagram yang paling dalam pemodelan system berbasis object-orientes. Class Diagram memperlihatkan sekumpulan class, interface dan collaborations dan relasi didalamnya. Selama proses analisa, class diagram memperhatikan aturan – aturan dan tanggung jawab entitas yang menentukan perilaku sistem . Selama tahap desain, class diagram berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. Kita memodelkan class diagram untuk memodelkan static design view dari suatu sistem. (Desy, class Diagram, 2018)

5.2.3. Pemodelan Berorientasi Aliran

Walaupun pemodelan berorientasi data saat ini dianggap merupakan teknik yang kadaluwarsa oleh kebanyakan rekayawan perangkat lunak. Meskipun DFD (Data Flow Diagram) serta diagram – diagram dan informasi – informasi yang terkait bukan bagian dari UML, pada dasarnya tetap dapat digunakan untuk melengkapi diagram – diagram UML dan menyediakan wawasan tambahan yang berkaitan dengan kebutuhan – kebutuhan dna aliran – aliran data yang terjadi pada sistem.

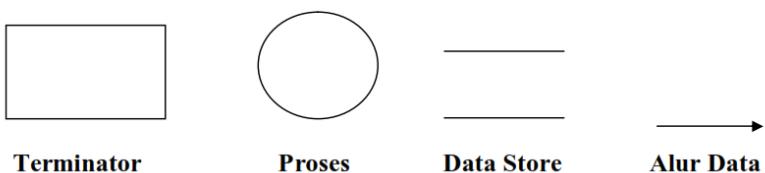
DFD memperlihatkan gambaran tentang masukan proses keluaran dari suatu perangkat lunak yaitu objek – objek data mengalir ke dalam perangkat lunak, kemudian objek – objek data itu akan ditransformasi oleh elemen – elemen pemrosesan, dan objek – objek data hasilnya akan mengalir keluar dari perangkat lunak.

objek – objek data dalam penggambaran DFD biasanya direpresentasikan menggunakan panah berlabel, dan transformasi – transformasi biasanya direpresentasikan menggunakan lingkaran – lingkaran yang biasa disebut dengan gelembung. DFD pada dasarnya digambarkan dalam bentuk hierarki yaitu DFD yang pertama disebut sebagai DFD tingkat 0 atau konteks yang menggambarkan sistem secara keseluruhan. DFD – DFD berikutnya merupakan penghalusan dari diagram konteks, yang memberikan gambaran yang semakin rinci dari diagram konteks, dalam hal ini akan berlanjut ke peringkat – peringkat selanjutnya.

I. Membuat Model Aliran Data -- Data Flow Diagram (DFD)

a. Komponen Data Flow Diagram

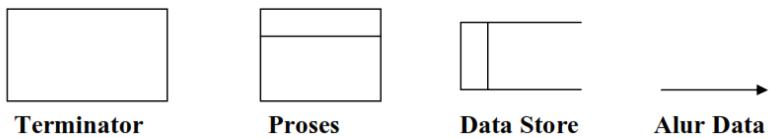
Menurut Yourdan dan Demarco



Gambar 5.19 Komponen DFD menurut
Ypurdan dan Demarco

Sumber : (Darmastuti, 2018)

Menurut Gene dan Serson

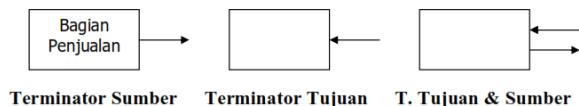


Gambar 5.20 Komponen DFD menurut Gene dan Serson

Sumber : (Darmastuti, 2018)

I) Terminator

Terminator mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Terdapat dua jenis terminator yaitu terminator sumber dan terminator tujuan

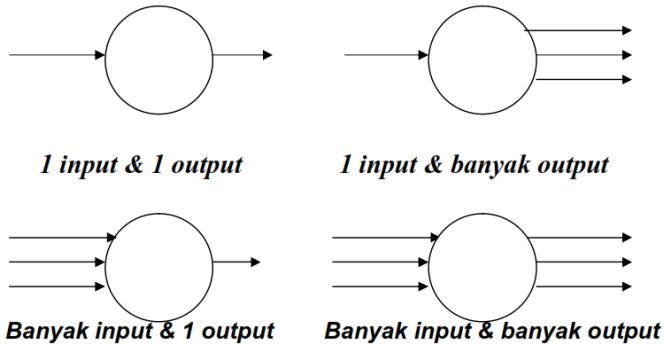


Terminator Sumber Terminator Tujuan T. Tujuan & Sumber

Gambar 5.21 Jenis Terminator

2) Proses

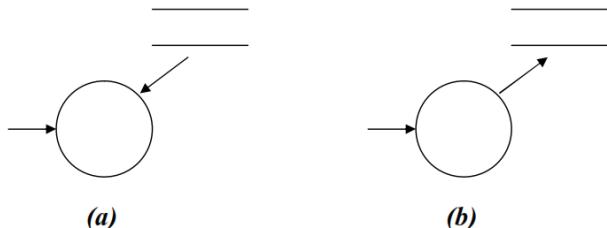
Komponen proses menggambarkan bagian dari sistem yang mentransformasikan input menjadi output. Ada empat kemungkinan yang dapat terjadi dalam proses sehubungan input dan output :



Gambar 5.22 hubungan input output pada komponen proses

3) Data Store

Komponen data store digunakan untuk membuat model sekumpulan paket data dan diberi nama dengan kata benda jamak. Alur data yang menghubungkan data store dengan suatu proses mempunyai pengertian yaitu (a) alur data dari data store yang berarti sebagai pembacaan satu paket tunggal data, (2) alur data ke data store yang berarti sebagai pengupdatetan data.



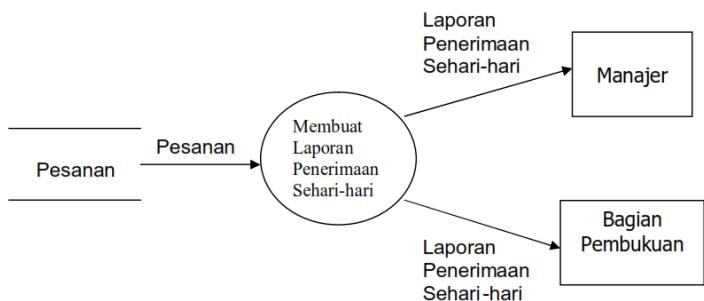
Gambar 5.23 Implementasi Data Store

4) Data Flow atau Alur data

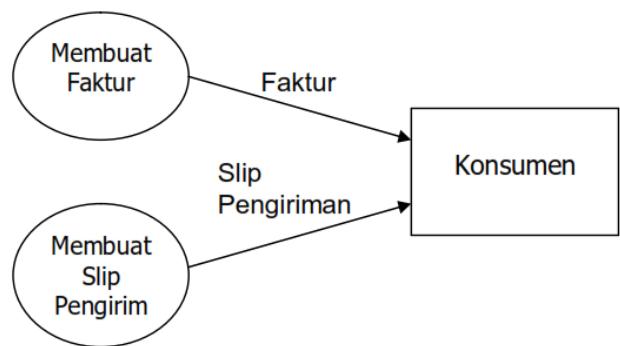
Suatu data flow digambarkan dengan anak panah yang menunjukkan arah menuju ke dan keluar dari suatu proses. Alur data digunakan untuk menerangkan perpindahan data atau paket data dari satu bagian sistem ke bagian lainnya. Ada empat konsep dalam penggambaran alur data :



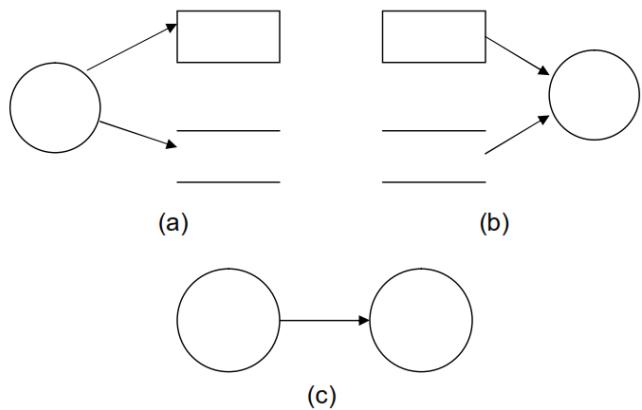
Gambar 5.24 Konsep Paket Data



Gambar 5.25 Konsep Alur Data
Menyebar

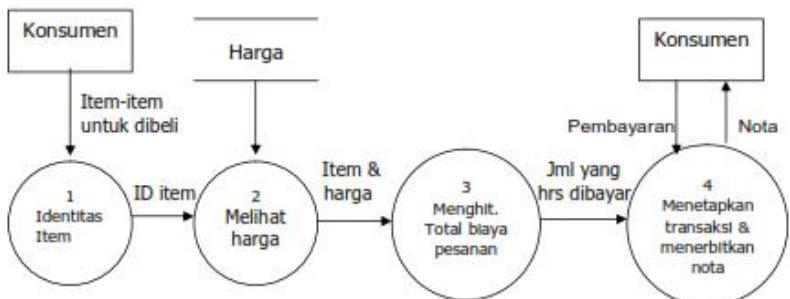


Gambar 5.26 Konsep Alur Data
Mengumpul

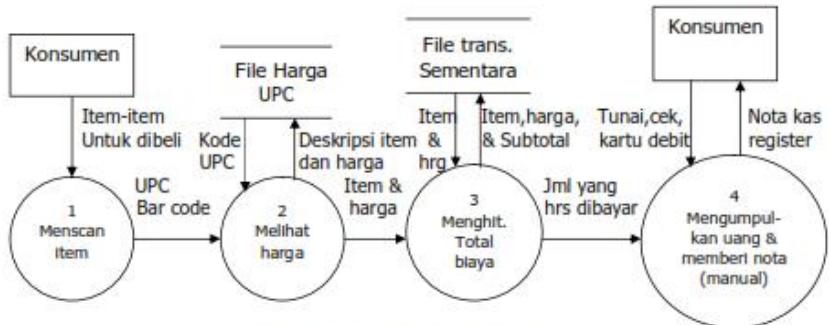


Gambar 5.27 Konsep Sumber atau
Tujuan Alur Data

- b. Bentuk Data Flow Diagram
 Terdapat dua bentuk DFD yaitu diagram alur data fisik (DADF) dan diagram alur data logic (DADL) . DADF digunakan untuk menggambarkan sistem yang ada atau sistem yang lama , sedangkan DADL digunakan untuk menggambarkan sistem yang baru atau usulan.



(a) Diagram Alur Data Logika



(b) Diagram Alur Data Fisik

Gambar 5.28 DADL dan DADF

Sumber : (Darmastuti, 2018)

c. Penggambaran DFD

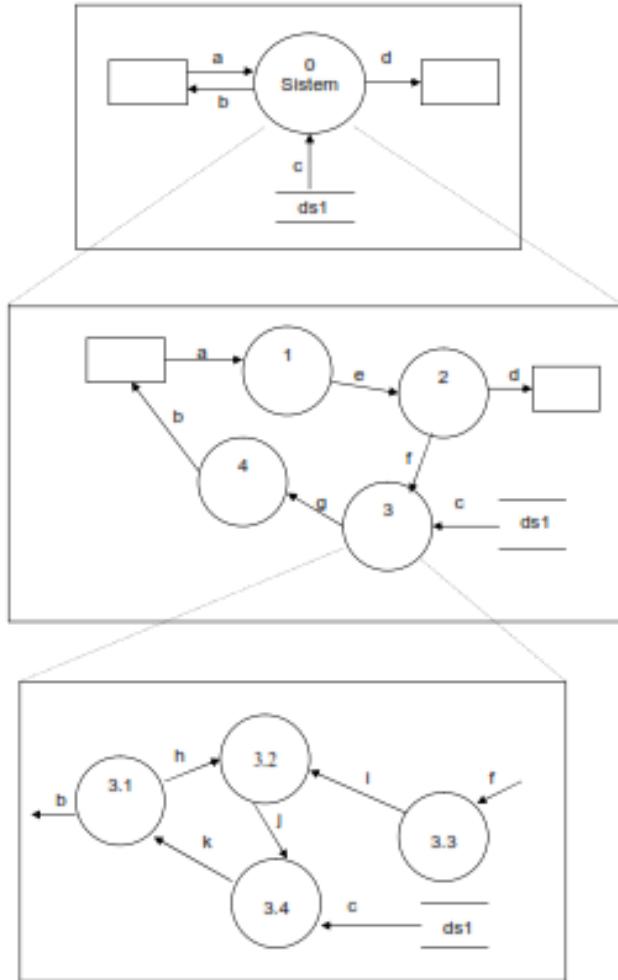
Secara garis besar langkah untuk membuat DFD adalah sebagai berikut :

- I) Identifikasi terlebih dahulu semua entitas luar yang terlibat di sistem

- 2) Identifikasi semua input dan output yang terlibat dengan entitas
- 3) Buat diagram konteks, diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya. Cara membuat diagram konteks :
 - a) Tentukan nama sistemnya
 - b) Tentukan batasan sistemnya
 - c) Tentukan terminator apa saja yang ada dalam sistem
 - d) Tentukan apa yang diterima atau diberikan terminator dari atau ke sistem
 - e) Gambarakan diagram konteks
- 4) Buat diagram level zero, diagram ini adalah dekomposisi dari diagram konteks. Cara membuat diagram level zero :
 - a) Tentukan proses utama yang ada pada sistem
 - b) Tentukan apa yang diberikan atau diterima masing – masing proses ke atau dari sistem sambil

- memperhatikan konsep keseimbangan
- c) Apabila diperlukan munculkan data store (master) sebagai sumber atau tujuan alur data
 - d) Gambarkan diagram level zero.
- 5) Buat diagram level satu, diagram ini merupakan dekomposisi dari diagram level zero. Cara mengambarkan diagram level zero :
- a) Tentukan proses yang lebih kecil dari proses utama yang ada di level zero
 - b) Tentukan apa yang diberikan atau diterima dari proses ke atau dari sistem dan perhatikan konsep keseimbangan
 - c) Apabila diperlukan munculkan data store (trasaksi) sebagai sumber maupun tujuan alur data.
 - d) Gambarkan DFD level satu
- 6) DFD level dua, tiga dan seterusnya . Diagram ini

merupakan dekomposisi dari level sebelumnya . Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program . Aturan yang digunakan sama dengan level satu.



Gambar 5.29 levelisasi DFD

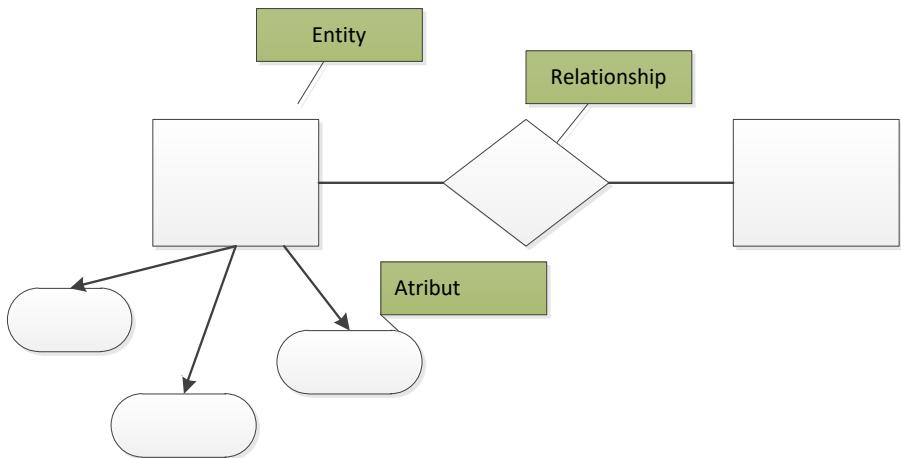
2. Model Data

Jika kebutuhan – kebutuhan perangkat lunak mencakup didalamnya kebutuhan untuk membuat , memperluas atau bersinggungan dengan basis data atau jika struktur data yang kompleks harus dibentuk dan dimanipulasi, tim perangkat lunak dapat menggunakan pemodelan data. Rekayasaan perangkat lunak dapat mendefinisikan semua objek data yang akan diproses didalam perangkat lunak, mendefinisikan relasi antar objek dengan menggunakan pemodelan data yaitu ERD (Entity Relationship Diagram).

Entity Relationship Diagram

Entity relationship diagram adalah suatu model penyajian data dengan menggunakan entity dan relationship. ERD menggambarkan model konseptual untuk menggambarkan struktur logis dari basisdata berbasis grafis yang bertujuan agar database dapat dipahami dan dirancang dengan mudah.

a. Simbol – simbol ERD



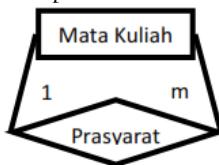
Gambar 5.30 Simbol – simbol ERD

- 1) Entity , objek yang dapat dibedakan dalam dunia nyata .
- 2) Relationship , hubungan yang terjadi antara satu atau lebih entity
- 3) Atribut , karakteristik dari setiap entity yang menyediakan penjelasan detail mengenai entity tersebut. Nilai dari atribut adalah data aktual atau informasi yang disimpan pada suatu atribut di dalam entity, dimana tiap atribut memiliki domain tersendiri. Jenis

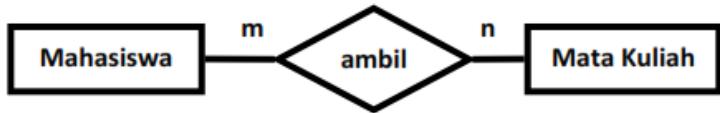
– jenis atribut yang digunakan dalam ERD :

- a) Key , atribut yang digunakan untuk menentukan suatu entity secara unik
- b) Atribut simple , atribut sederhana yang tidak dapat dibagi dalam beberapa bagian
- c) Atribut komposit , atribut yang dapat dibagi dalam beberapa bagian. Contohnya adalah alamat yang dapat dibagi menjadi kota, propinsi , negara.
- d) Atribut single-valued, atribut yang memiliki paling banyak satu nilai untuk setiap baris data.
- e) Multi-valued attributes , atribut yang dapat diisi dengan lebih satu nilai tetapi sejenisnya sama. Contohnya nomor telp. Alamat, gelar
- f) Atribut turunan, atribut yang diperoleh dari pengolahan dari atribut lain yang berhubungan.

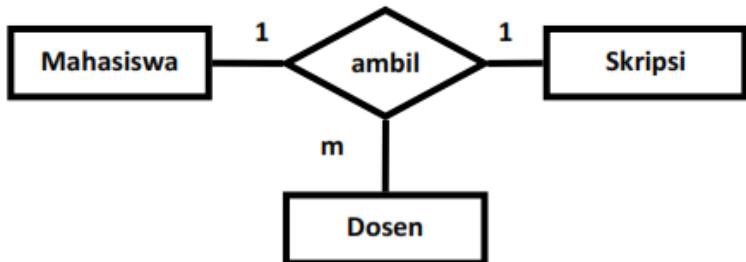
- g) Atribut Key , attribut yang dapat dijadikan kunci untuk mencari data dalam relasi .
- b. Derajat Relasi
- Derajat relasi menunjukkan banyaknya himpunan entitas yang saling berelasi . jenis derajat himpunan relasi adalah (Satrio Agung W, 2011):
- 1) Unary degree(derajat satu) melibatkan sebuah entitas yang berelasi dengan dirinya sendiri
 - 2) Binary degree(derajat dua) himpunan relasi melibatkan dua himpunan entitas. Secara umum himpunan relasi dalam sistem basis data adalah binary.
 - 3) Ternary degree(derajat tiga) himpunan relasi memungkinkan untuk melibatkan lebih dari dua himpunan entitas



Gambar 5.31 Unary degree



Gambar 5.32 Binary degree

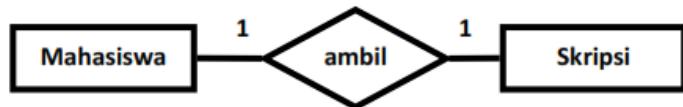


Gambar 5.33 Ternary degree

c. Pemetaan Kardinalitas Relasi
 Pemetaan kardinalitas relasi menggambarkan banyaknya jumlah maksimum entitas dapat berelasi dengan entitas pada himpunan entitas yang lain. Untuk himpunan relasi biner pemetaan kardinalitasnya dapat dibagi menjadi (Satrio Agung W, 2011):

- I) One to one, sebuah entity hanya dapat berelasi dengan satu buah objek di entity yang lain.
- 2) One to many, sebuah entity dapat berelasi dengan banyak objek di entity yang lain.

- 3) Many to one, banyak entity akan berelasi dengan satu objek yang sama pada entity yang lain.
- 4) Many to many, banyak entity yang akan berelasi dengan banyak objek di entity yang lain.



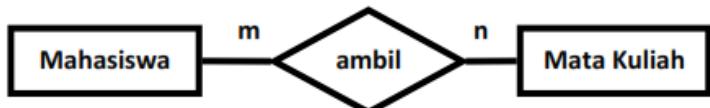
Gambar 5.34 One to one



Gambar 5.35 One to many



Gambar 5.36 Many to one



Gambar 5.37 Many to many

5.2.4. Pemodelan Berbasis Perilaku

Model perilaku menggambarkan bagaimana perangkat lunak akan berperilaku dalam menghadapi event – event yang datang dari arah luar atau bagaimana perangkat lunak akan berperilaku terhadap rangsangan – rangsangan yang muncul dari arah luar. Berikut langkah – langkah untuk membuat model ini : (a) melakukan evaluasi terhadap semua use case untuk secara penuh memahami urutan – urutan interaksi yang terjadi di dalam sistem perangkat lunak, (b) mengidentifikasi event – event yang mengendalikan urutan interaksi – interaksi dan memahami bagaimana event – event itu berhubungan dengan objek – objek yang bersifat spesifik, (c) membuat diagram – diagram yang memperlihatkan urutan – urutan untuk masing – masing use case, (d) mengembangkan diagram state untuk perangkat lunak yang akan dikembangkan, (e) meninjau model perilaku untuk memverifikasi akurasi dan konsistensinya.

I. Mengidentifikasi Event – event Menggunakan Use Case

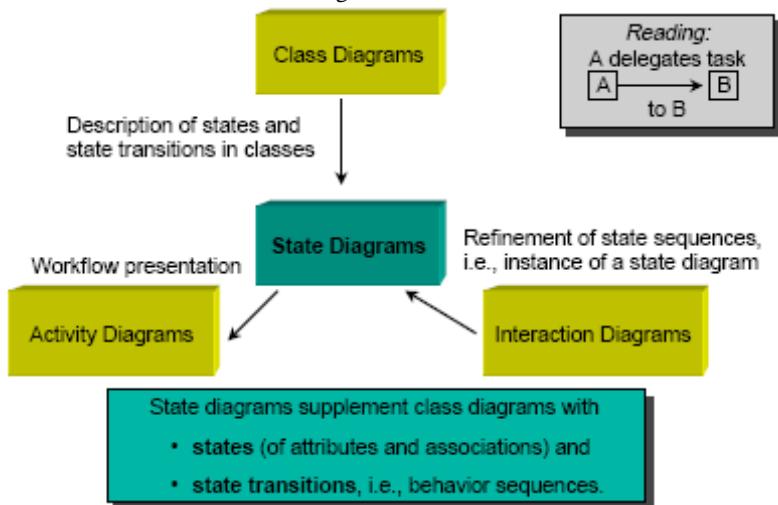
Use case pada dasarnya merepresentasikan suatu urutan aktivitas – aktivitas yang terjadi di antara aktor – aktor dan perangkat lunak yang sedang dirancang atau kembangkan, dan event terjadi saat sistem dan aktor – aktornya bertukar informasi. Jadi kita dapat mengidentifikasi event – event dengan melihat aktivitas – aktivitas use

case serta melihat event pertukaran informasi atau data yang ada pada aktivitas use case tersebut.

2. Representasi Keadaan (*State*)

Pada pemodelan berbasis perilaku ada dua karakteristik keadaan (state) yang perlu dipertimbangkan yaitu (a) keadaan dari masing – masing kelas , saat kelas – kelas itu melaksanakan fungsinya masing – masing, (b) keadaan sistem saat sistem itu diobservasi dari luar saat sistem itu melaksanakan fungsinya. Komponen dari pemodelan prilaku dalam UML adalah diagram state dan sequence diagram / diagram urutan aksi – aksi.

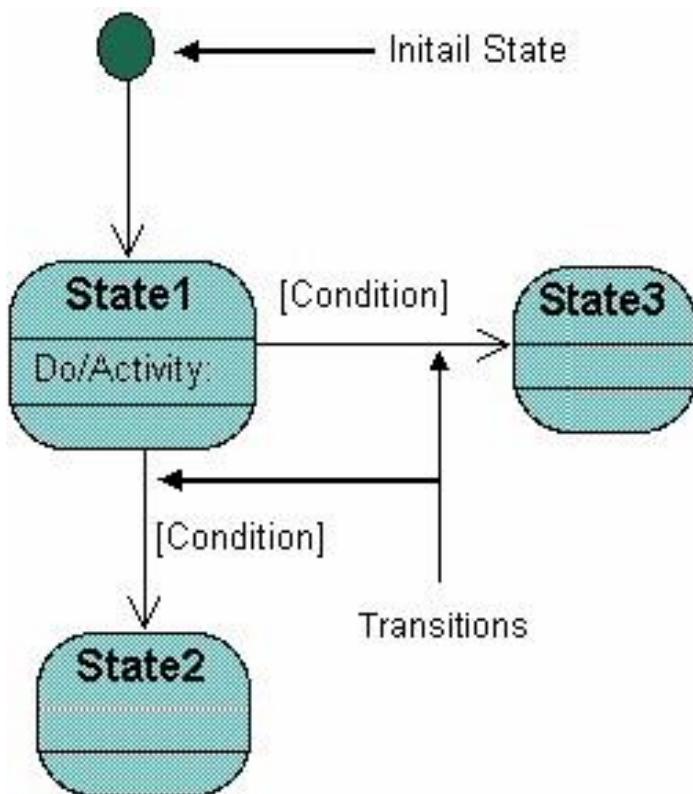
a. State Diagram



Gambar 5.38 Peran State Diagram
dalam UML

Sumber : (Gunadarma.ac.id, 2018)

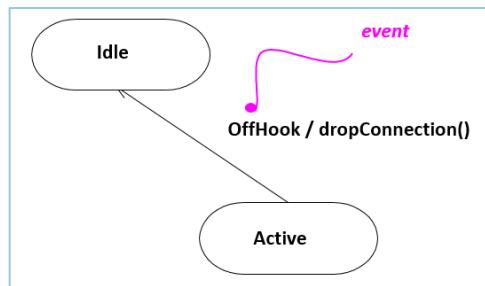
Diagram state merupakan diagram untuk menggabarkan behavior yaitu perubahan keadaan di suatu class berdasarkan event dan message yang dikirim dan diterima oleh class tersebut. Setiap diagram state hanya boleh memiliki satu start state (initial state) dan boleh memiliki satu atau lebih dari satu stop states.



Gambar 5.38 Contoh diagram state
Sumber : (Gunadarma.ac.id, 2018)

State merupakan abstraksi dari nilai – nilai atribut dan asosiasi dari sebuah objek, state merepresentasikan konsisi dari sebuah objek pada periode waktu tertentu dan berhubungan dengan suatu interbal waktu antara dua event. Respon

terhadap event dapat tergantung kepada state suatu objek. Event merupakan spesifikasi dari suatu kejadian tertentu, dan kita dapat memodelkan sesuatu kejadian sebagai event.



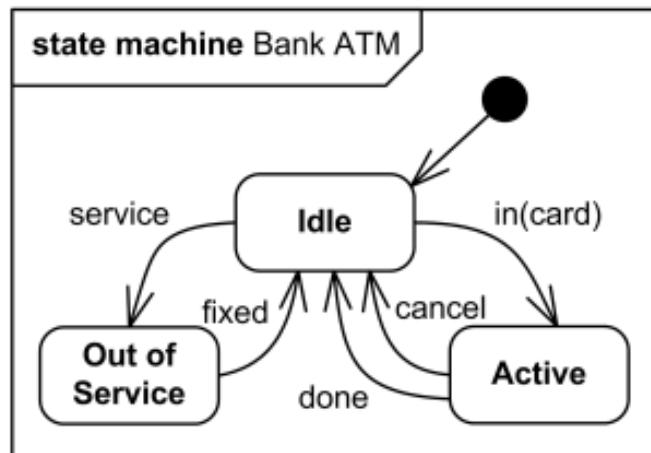
Gambar 5.39 Contoh event
Sumber : (Gunadarma.ac.id, 2018)

Event dapat dikategorikan menjadi dua yaitu internal event (event yang berasal dari dan menuju ke objek pada sistem), eksternal event (event yang berasal dari aktor ke sistem atau sebaliknya).

State Machine Diagram

State machine adalah behavior yang menggambarkan urutan state dari objek sepanjang waktu hidupnya. State Machine Diagram digunakan untuk menggambarkan perubahan status atau

transisi status dari sebuah mesin atau sistem atau objek (Rosa A. S, 2018).



Gambar 5.40 Contoh State Diagram

Sumber : (uml-diagrams.org, State Machine Diagrams, 2018)

Berikut ini komponen – komponen dasar yang ada dalam state machine diagram :

Tabel 5.3 Komponen – komponen dasar pada state machine

NO	GAMBAR	NAMA	KETERANGAN
I		<i>State</i>	State atau status adalah keadaan sistem pada waktu tertentu. State dapat berubah

			jika ada event tertentu yang memicu perubahan tersebut
2		<i>Initial State</i>	Start atau initial state adalah state atau keadaan awal pada saat sistem mulai hidup.
3		<i>Final State</i>	Final State adalah state keadaan akhir dari daur hidup suatu sistem
4	Event 	<i>Event</i>	Event adalah kegiatan yang menyebabkan berubahnya status mesin.

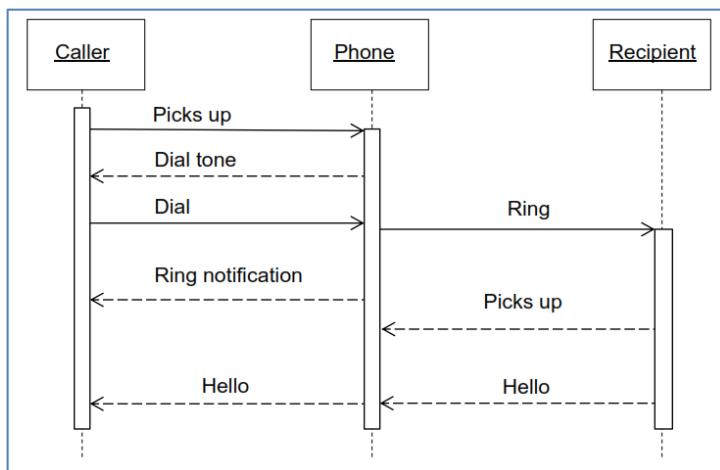
b. Sequence Diagram

Sequence diagram menggambarkan perilaku objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Untuk membuat sequence diagram perlu diketahui terlebih dahulu objek – objek yang terlibat dalam use case dan skenario perannya.

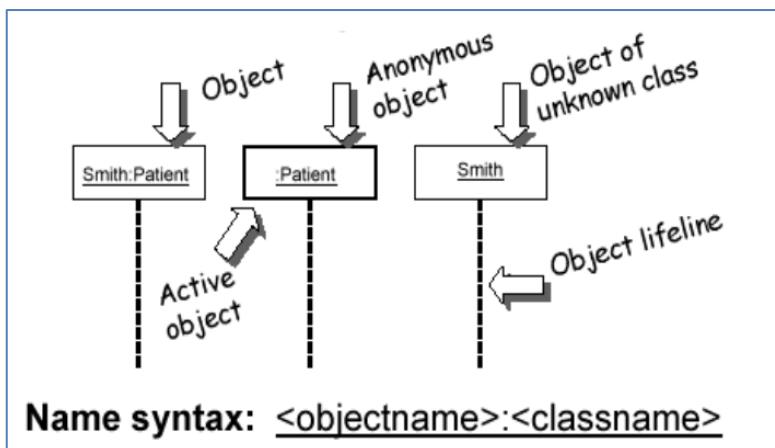
Bagian kunci dari diagram sequence adalah sebagai berikut (Fowler, 2018) :

- 1) Participant yaitu objek atau entitas yang melakukan kegiatan dalam diagram.
- 2) Message yaitu komunikasi antara participant dengan objek
- 3) The axes dalam diagram sequence :
 - a) Horizontal yaitu objek atau participant yang melakukan kegiatan
 - b) Vertical yaitu waktu

Berikut adalah contoh diagram sequence dari panggilan telepon :

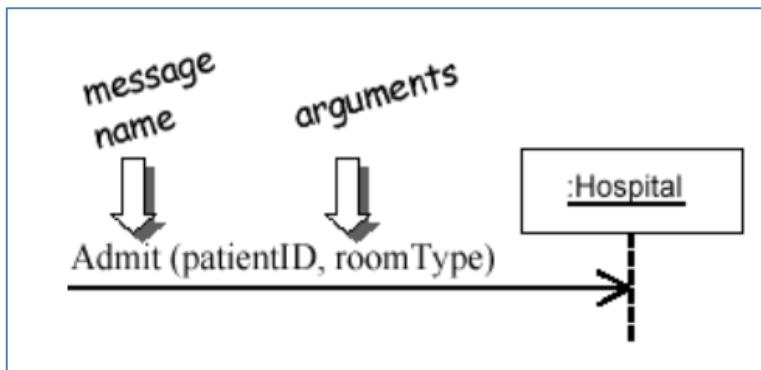


Gambar 5.41 Contoh diagram sequence
dari perilaku panggilan telepon
Sumber : (Fowler, 2018)



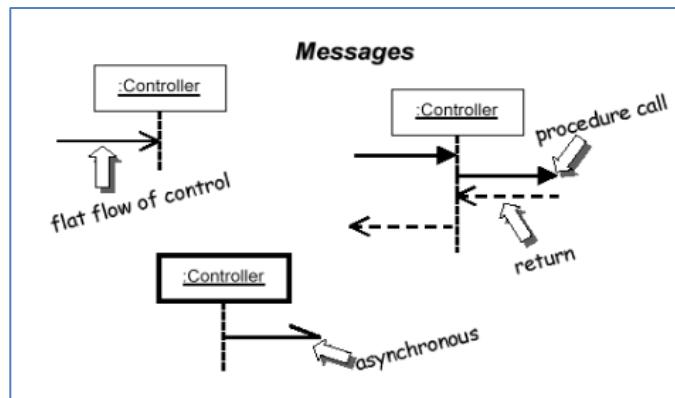
Gambar 5.42 Implementasi objek dalam
syntax

Sumber : (Fowler, 2018)



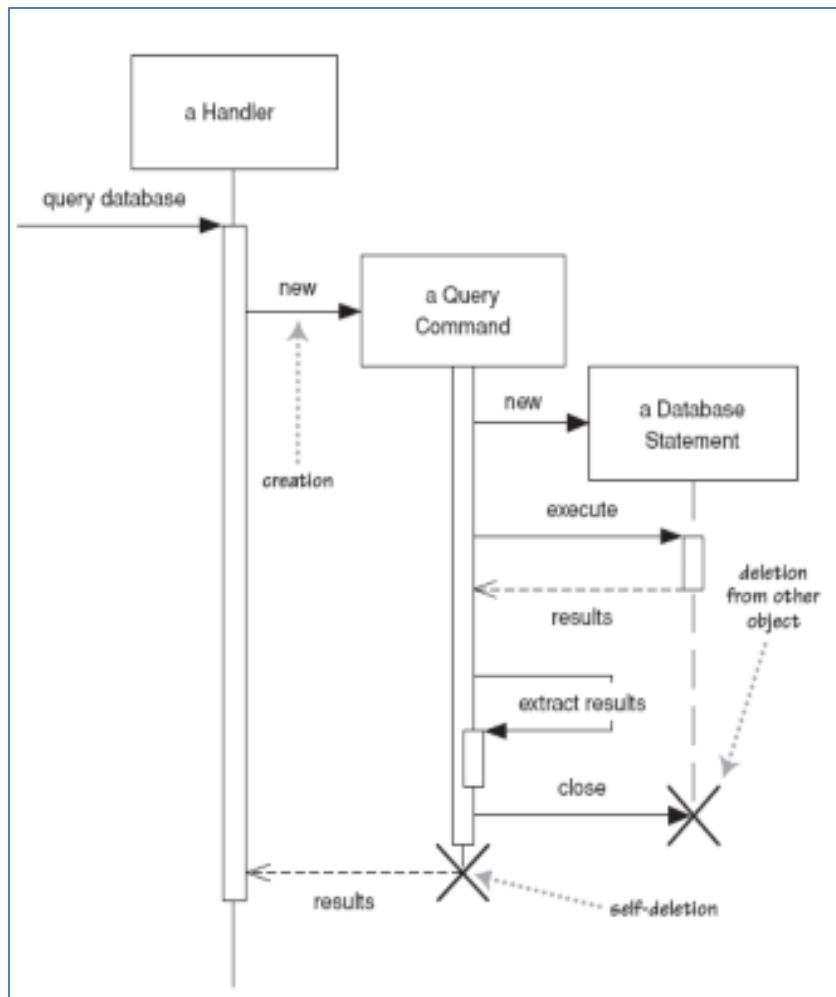
Gambar 5.43 Pesan antar objek

Sumber : (Fowler, 2018)



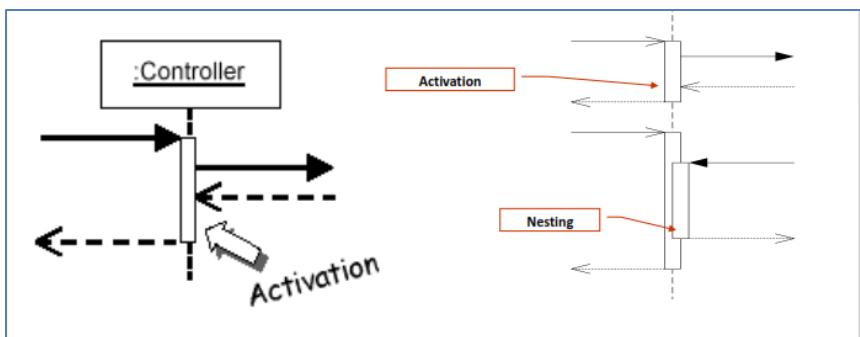
Gambar 5.44 Pesan yang dilanjutkan

Sumber : (Fowler, 2018)



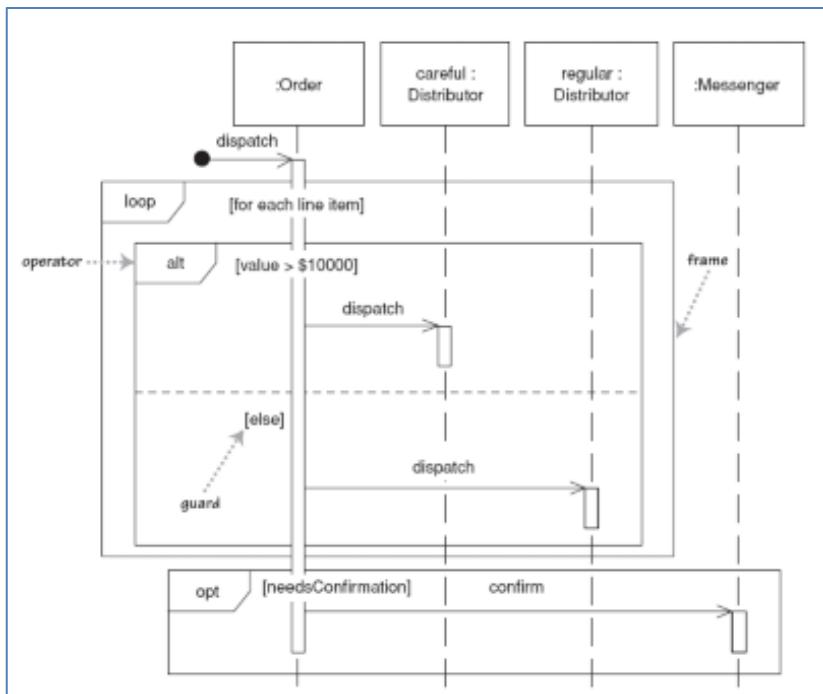
Gambar 5.45 Life time objek

Sumber : (Fowler, 2018)



Gambar 5.46 Activation

Sumber : (Fowler, 2018)



Gambar 5.47 Contoh loop dan selection
Sumber : (Fowler, 2018)

5.2.5. Pemodelan Kebutuhan – kebutuhan untuk Aplikasi – aplikasi Web

- I. Masukan Pemodelan Kebutuhan pada aplikasi WEB

Proses perangkat lunak generik versi cepat dapat digunakan pada rekayasa aplikasi – aplikasi web. Masukan – masukan untuk model – model kebutuhan pada aplikasi web adalah informasi

yang terkumpul melalui aktivitas – aktivitas komunikasi.

2. Keluaran dari Pemodelan Kebutuhan
Analisi kebutuhan pada aplikasi web menyediakan suatu mekanisme yang representasikan dan mengevaluasi isi – isi dan fungsi – fungsi yang dimiliki suatu web, merepresentasikan interaksi – interaksi yang dilakukan pengguna dan merepresentasikan lingkungan dan infrastruktur dimana aplikasi web itu berada. Masing – masing dari karakteristik dapat di representasikan sebagai sebuah model. Ada lima jenis kelas model yang utama untuk aplikasi web yaitu :
 - a. Model isi
Mengidentifikasi isi yang ada dalam aplikasi web. Model isi memuat di dalamnya elemen – elemen struktural yang menyediakan suatu bentuk tampilan untuk kebutuhan isi dari aplikasi web.
 - b. Model interaksi
Mendefinisikan cara para pengguna akan berinteraksi dengan aplikasi web. Model interaksi dapat disusun ke dalam elemen – elemen *use case*, *squence diagram*, *state diagram* dan prototipe – prototipe antar muka pengguna.
 - c. Model fungsional
Mendefinisikan operasi – operasi yang akan diterapkan pada isi aplikasi web dan

- mendeskripsikan fungsi – fungsi pemrosesan isi yang diperlukan oleh pengguna. Model fungsional menyelesaikan dua elemen proses yaitu fungsionalitas yang dapat diobservasi oleh pengguna dan operasi – operasi yang dimuat di dalam kelas – kelas analisis yang mengimplementasikan perilaku – perilaku yang berhubungan dengan kelas – kelas analisis yang mengimplementasikan perilaku – perilaku yang berhubungan dengan kelas – kelas analisis. Model fungsional dapat di susun kedalam diagram aktivitas.
- d. Model navigasi, mendefinisikan strategi navigasi keseluruhan untuk aplikasi web
 - e. Model konfigurasi, mendeskripsikan lingkungan serta infrastruktur dimana aplikasi web akan berada.

BAB 6 Konsep Perancangan

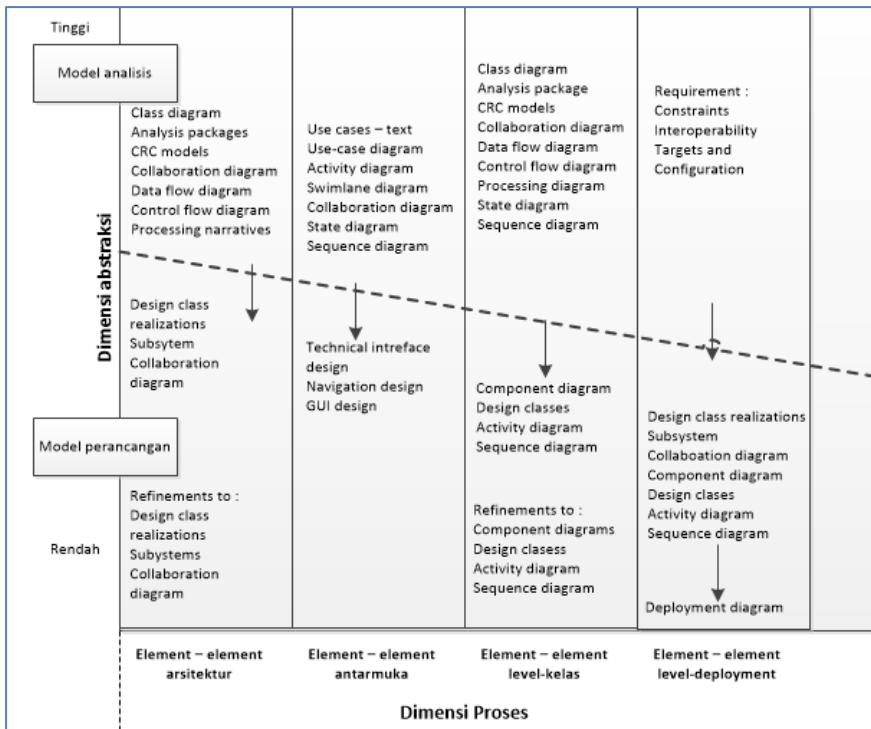
Perangkat Lunak

Perancangan perangkat lunak menyediakan secara rinci hal – hal yang terkait dengan arsitektur perangkat lunak , struktur – struktur data yang digunakan dalam perangkat lunak, antarmuka – antarmuka , komponen – komponen dan subsistem – subsistem yang diperlukan untuk implementasi perangkat lunak. Sasaran dari perancangan perangkat lunak adalah untuk menghasilkan suatu model atau representasi perangkat lunak yang memperlihatkan kekuatan , komoditi serta kenyamanan.

Perancangan arsitektur perangkat lunak mendefinisikan relasi antara elemen – elemen struktural utama dari suatu perangkat lunak, mendefinisikan gaya arsitekturnya dan pola – pola perancangan untuk mencapai kebutuhan yang telah dianalisis sebelumnya. Representasi dari arsitektur perangkat lunak adalah kerangka kerja suatu sistem berbasis komputer . Sedangkan rancangan antarmuka mengdeskripsikan cara komunikasi perangkat lunak dengan pengguna atau sistem lain.

Perancangan perangkat lunak merupakan proses yang bersifat interatif , spesifikasi – spesifikasi kebutuhan perangkat lunak diterjemahkan menjadi suatu blueprint untuk mengkonstruksi perangkat lunak. *Blueprint* memperlihatkan tampilan secara menyeluruh dari suatu perangkat lunak.

Sedangkan model perancangan perangkat lunak dapat dilihat menggunakan dua dimensi yang berbeda . berikut dimensi – dimensi model perancangan .



Gambar 6.1 Dimensi – dimensi model perancangan

Sumber : (Roger S. Pressman, 2012)

6.I. Perancangan Arsitektural

Menurut Bas, Clements dan Kazman dalam (Roger S. Pressman, 2012) mengemukakan bahwa Arsitektur sistem

pada dasarnya adalah struktur sistem yang menggabungkan komponen – komponen perangkat lunak , menggabungkan properti – properti yang tampak dari komponen – komponen itu , serta mendeskripsikan hubungan antarkomponen itu.

Arsitektur sistem menjelaskan suatu susunan sistem yang terdiri dari komponen software, atribut dari komponen dan hubungan antar komponennya. Komponen dapat berupa modul, database, middleware atau class. Atribut adalah ciri dan fungsi dari modul sedangkan hubungan antar komponen adalah cara antar komponen tersebut berkomunikasi.

Arsitektur sistem memungkinkan kita untuk melakukan analisis terhadap efektivitas perancangan yang disesuaikan dengan kebutuhan – kebutuhan yang telah dinyatakan sebelumnya, memungkinkan untuk melakukan pertimbangan alternatif – alternatif arsitektural dan memungkinkan mengurangi risiko – risiko yang berhubungan dengan konstruksi perangkat lunak.

6.I.I. Strukur Arsitektur

Strukur arsitektur terdiri dari :

1. Struktur fungsional
Struktur fungsional merupakan komponen yang mewakili fungsi atau proses .
2. Struktur implementasi
Struktur implementasi merupakan komponen yang mewakili package , class, object , procedure, function atau method.
3. Struktur konkurensi

Struktur konkurensi merupakan komponen yang mewakili unit – unit yang tersusun secara paralel atau konkuren kerjanya.

4. Struktur fisik

Struktur fisik merupakan komponen hardware dan software yang ada.

5. Struktur pengembangan

Struktur pengembangan merupakan struktur yang didefinisikan berupa komponen , produk kerja dan sumber informasi lain yang dibutuhkan selama pengembangan software.

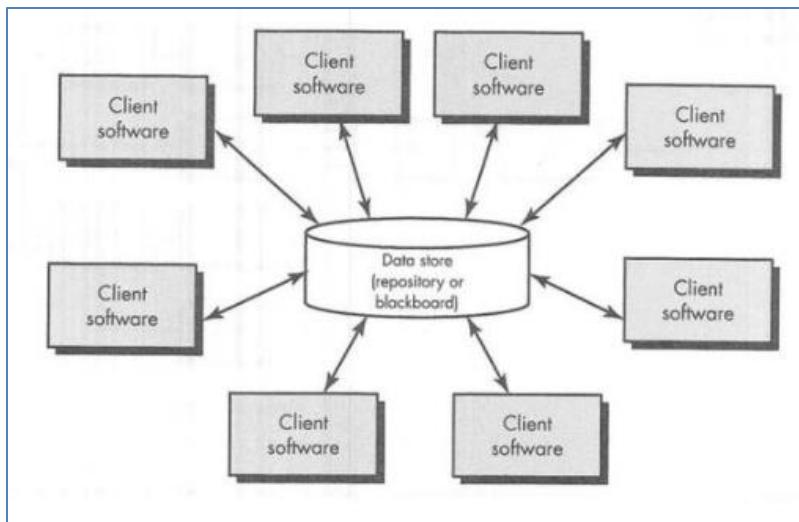
Setiap struktur diatas mewakili arsitektur software dari sudut pandang yang berbeda – beda , mengekspose informasi yang digunakan tim pengembang selama proses desain dan coding berlangsung.

6.1.2. Klasifikasi Gaya Arsitektur

Gaya arsitektur terbagi menjadi dua yaitu :

I. Data-centered Architecture

Suatu data store menjadi pusat di antara komponen lain yang mengaksesnya dalam rangka untuk update, penambahan , penghapusan atau memodifikasi data yang ada dalam tempat penyimpanan data.

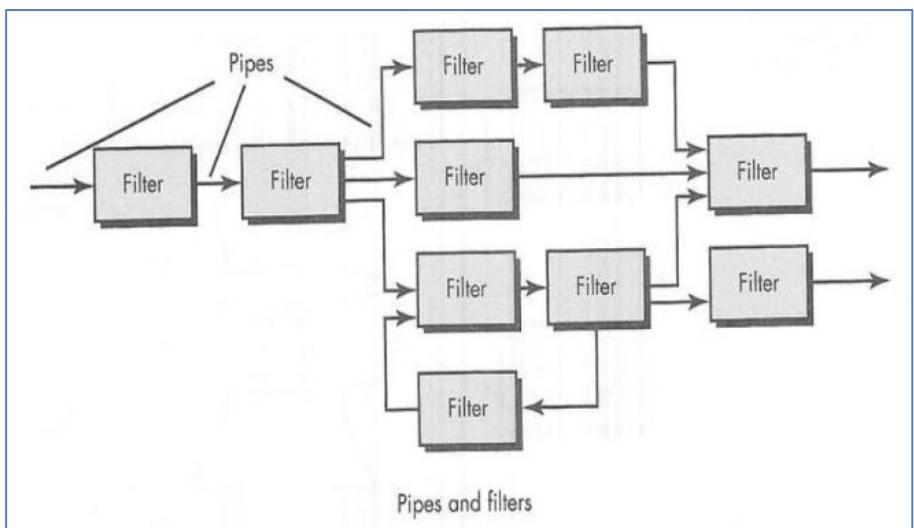


Gambar 6.2 Data-centered Architecture

Sumber : (Roger S. Pressman, 2012)

2. Data-flow Architecture

Arsitektur ini dimanfaatkan untuk menggabarkan input daya yang diubah melalui serangkaian perhitungan dan manipulasi untuk menjadi output.

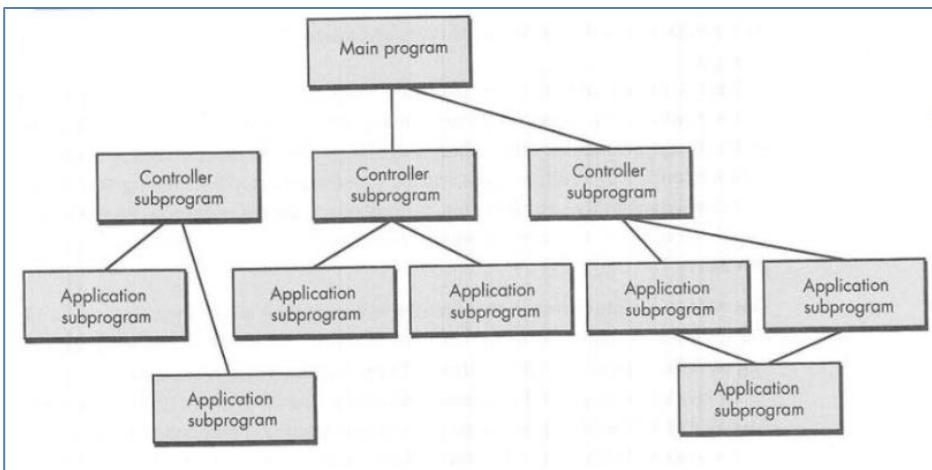


Gambar 6.3 Data-flow Architecture

Sumber : (Roger S. Pressman, 2012)

3. Call and Return Architecture

Menggambarkan struktur program yang disusun secara hirarki. Program dibagi menjadi beberapa sub program yang terdiri dari program utama dan beberapa sub program. Komponen mewakili sub program atau program utama.

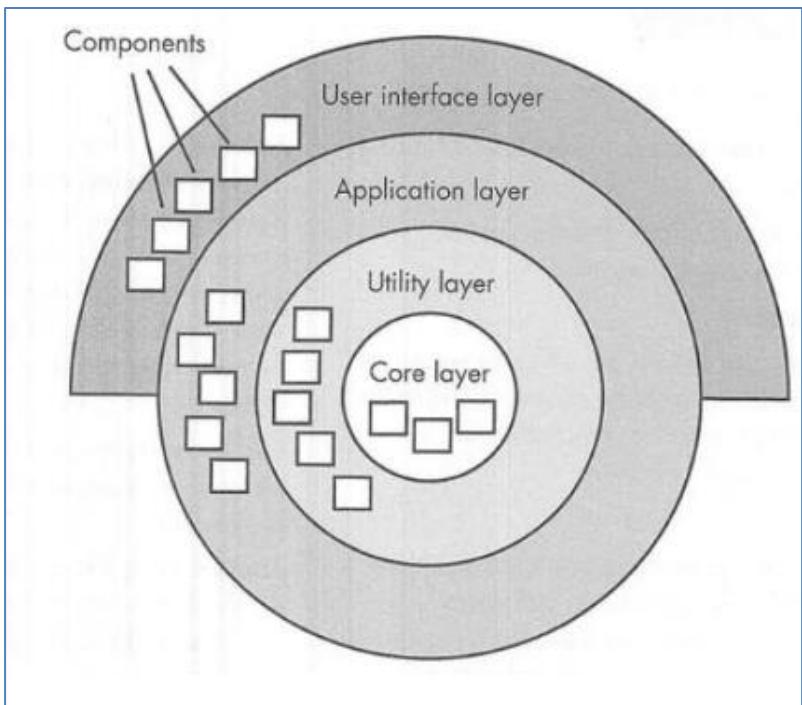


Gambar 6.4 Call and Return Architecture

Sumber : (Roger S. Pressman, 2012)

4. Layered Architecture

Setiap lapisan menjalankan operasinya dan makin ke dalam komponennya makin mendekati perintah mesin. Yang terluar adalah lapisan yang paling dekat dengan pengguna.



Gambar 6.5 Layered Architecture

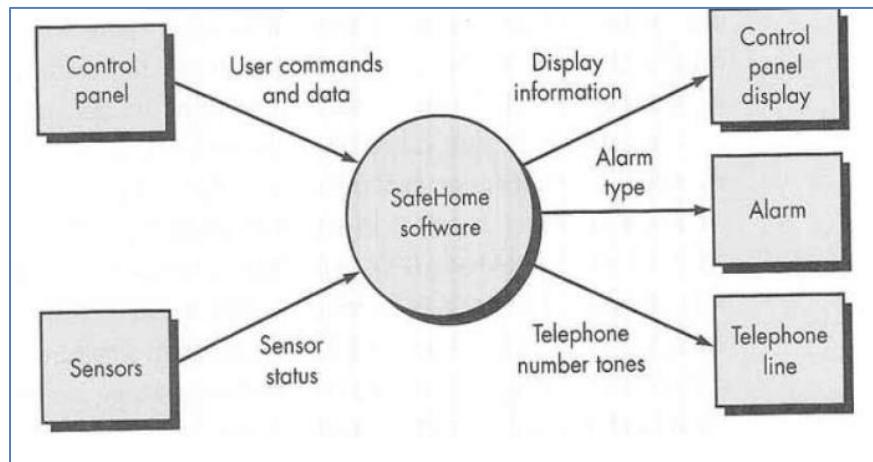
Sumber : (Roger S. Pressman, 2012)

6.1.3. Pemetaan Arsitektural Transformasi Menggunakan Aliran Data

Berikut adalah contoh yang diambil dari buku Pressman yang mentransformasikan dari DFD ke struktur call and return architecture dengan study case safe home security function.

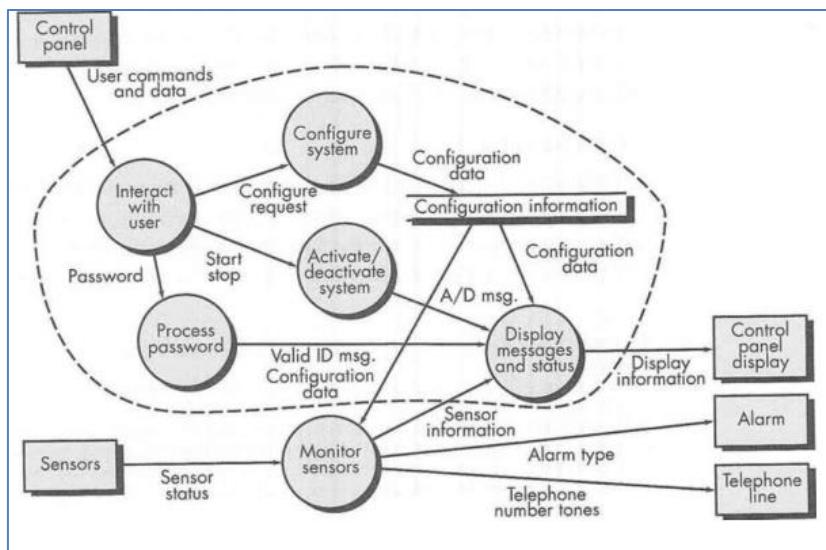
Langkah – langkah memetakan diagram – diagram aliran data ke dalam arsitektur perangkat lunak :

- I. Lakukan peninjauan pada model sistem yang paling mendasar .



Gambar 6.6 Context-level DFD untuk fungsi keamanan SafeHome

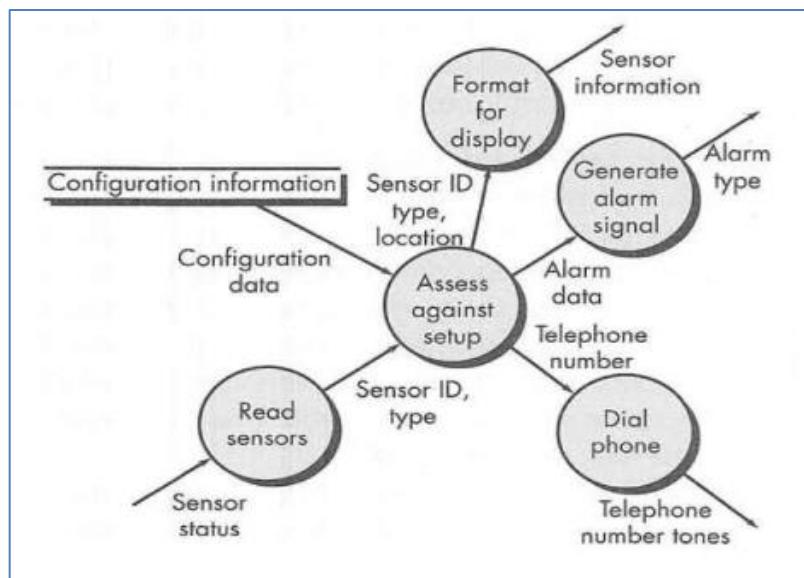
Sumber : (Roger S. Pressman, 2012)



Gambar 6.7 Level 1 DFD untuk fungsi keamanan SafeHome

Sumber : (Roger S. Pressman, 2012)

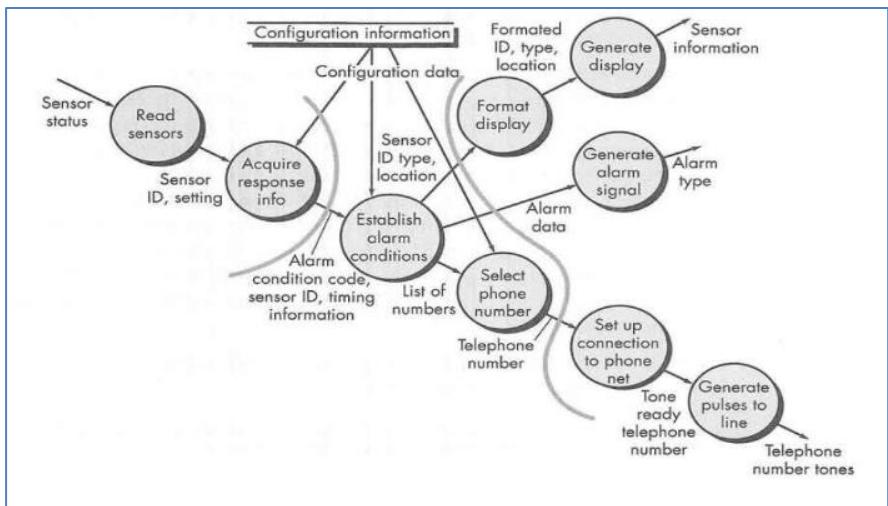
2. Lakukan peninjauan dan lakukan penghalusan diagram aliran data untuk perangkat lunak



Gambar 6.8 Level 2 DFD yang menghaluskan transformasi pemantauan sensor - sensor

Sumber : (Roger S. Pressman, 2012)

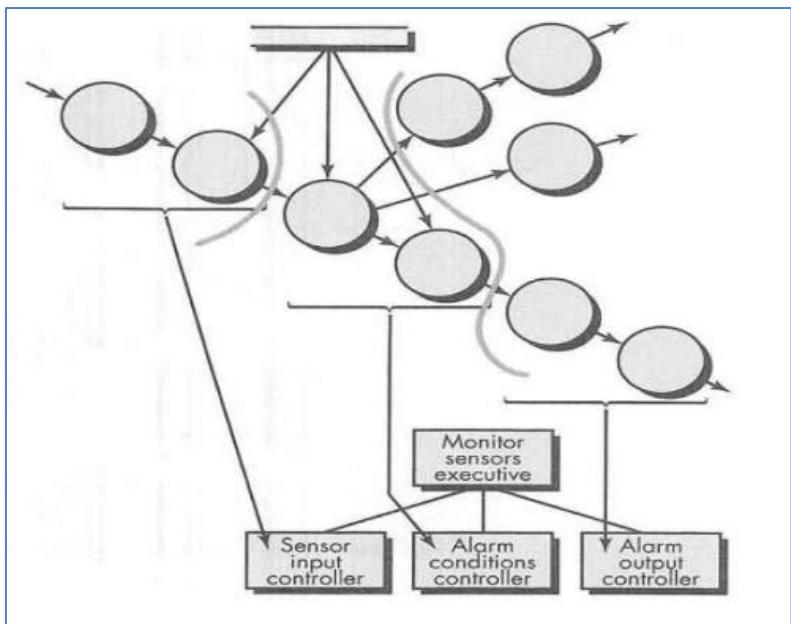
3. Menentukan apakah diagram aliran data memiliki karakteristik – karakteristik transformasi atau aliran transaksi.



Gambar 6.8 Level 3 DFD untuk pemantauan sensor – sensor dengan tambahan batasan – batasan aliran.

Sumber : (Roger S. Pressman, 2012)

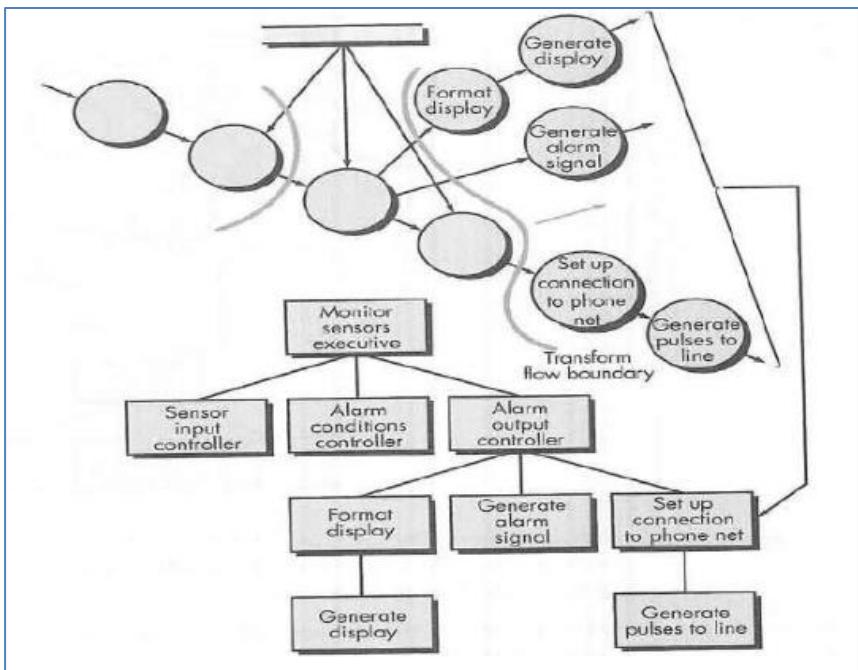
4. Pisahkan pusat transformasi dengan cara menspesifikasi batasan – batasan aliran yang masuk dan keluar
5. Lakukan faktorisasi peringkat pertama



Gambar 6.9 Faktorisasi peringkat pertama
untuk pemantauan sensor - sensor

Sumber : (Roger S. Pressman, 2012)

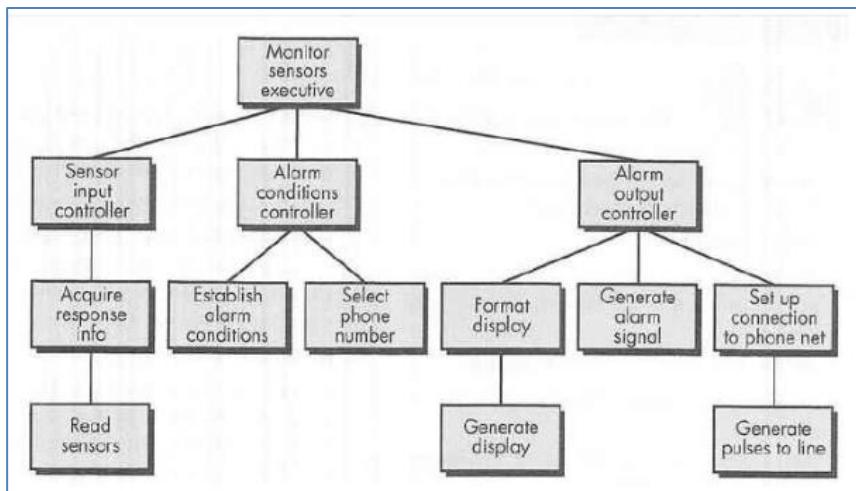
6. Melakukan faktorisasi peringkat kedua



Gambar 6.10 Faktorisasi peringkat kedua
untuk pemantauan sensor - sensor

Sumber : (Roger S. Pressman, 2012)

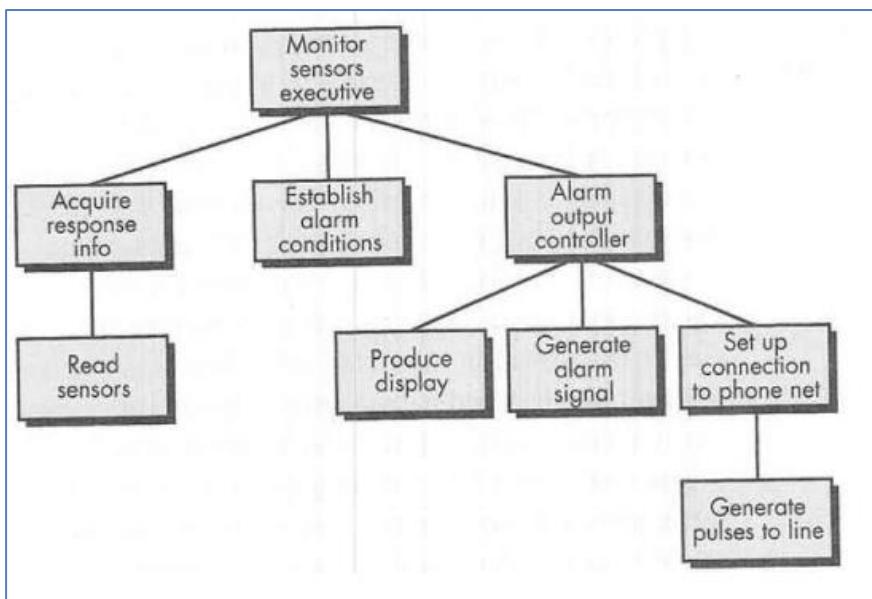
7. Perhalus arsitektur iterasi pertama menggunakan metode perancangan heuristic untuk meningkatkan kualitas perangkat lunak



Gambar 6.II struktur iterasi pertama untuk

pemantauan sensor - sensor

Sumber : (Roger S. Pressman, 2012)



Gambar 6.12 struktur program pemantauan
sensor – sensor yang telah mengalami
penghalusan.

Sumber : (Roger S. Pressman, 2012)

6.2. Perancangan pada Peringkat Komponen

Perancangan pada peringkat komponen terjadi setelah iterasi pada perancangan arsitektural diselesaikan. Tujuan dari perancangan pada peringkat komponen ini adalah untuk menterjemahkan model perancangan menjadi perangkat lunak operasional. Merupakan hal yang mungkin untuk merepresentasikan perancangan pada peringkat komponen menggunakan bahasa pemrograman tertentu. Suatu pendekatan

yang sering digunakan untuk merepresentasikan perancangan peringkat komponen adalah menggunakan berbagai representasi perantara yang dapat dengan mudah diterjemahkan menjadi kode – kode sumber yang ditulis menggunakan bahasa pemograman tertentu.

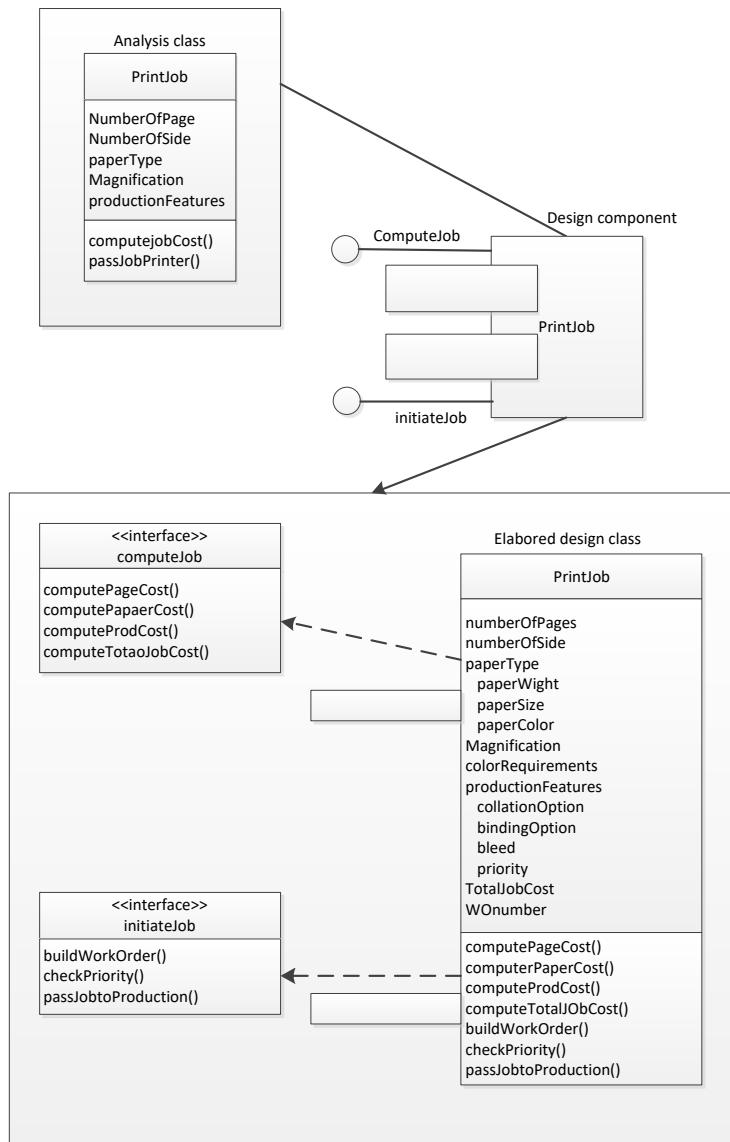
Komponen merupakan bangunan perangkat lunak yang bersifat modular . Komponen mengisi arsitektur sistem dan mengakibatkan memainkan peran dalam pencapaian sasaran dari kebutuhan sistem yang dikembangkan. Ada tiga pandangan terkait suatu komponen dan penggunaannya pada saat pemodelan perancangan berlangsung (Roger S. Pressman, 2012) :

Pandangan Berorientasi Objek

Suatu komponen memuat di dalam sejumlah kelas yang saling berkolaborasi. Masing – masing kelas di dalam suatu komponen telah dielaborasi dengan lengkap untuk melibatkan di dalamnya semua atribut dan operasi yang relevan untuk implemnetasinya.

Atribut kelas analisis

PrintJob didefiniskan sebagai komponen yang ada di dalam arsitektur perangkat lunak dia di presentasikan menggunakan notasi komponen UML sebagai berikut .



Gambar 6.12 Elaborasi sebuah komponen rancangan.

Sumber : (Roger S. Pressman, 2012)

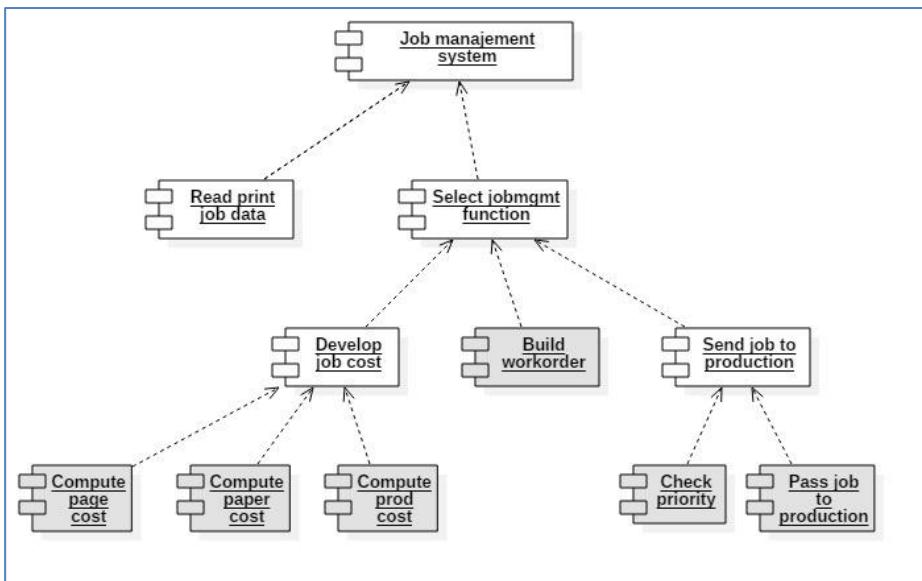
PrinJob memiliki dua antarmuka yaitu :

- a. computeJob , yang menyediakan kemampuan perhitungan biaya pekerjaan pencetakan
- b. initiateJob, yang akan melewatkkan pekerjaan pencetakan ke fasilitas produksi.

Pandangan Tradisional

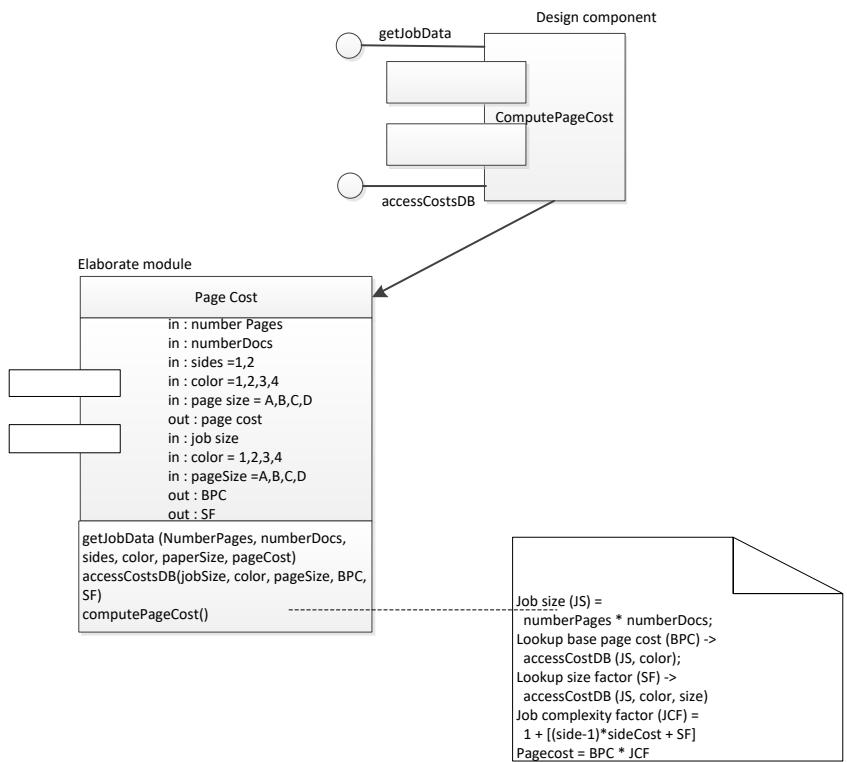
Sebuah komponen tradisional , sering disebut sebagai modul , yang berada dalam arsitektur perangkat lunak. Peran pentingnya adalah :

- a. Sebagai komponen kendali yang mengkoordinasi pemanggilan semua komponen ranah permasalahan yang lain.
- b. Sebuah komponen ranah permasalah yang mengimplemnetasikan secara lengkap atau secara bagian fungsi yang diperlukan untuk pelanggan
- c. Sebuah komponen infrastruktur yang bertangung jawab untuk fungsi – fungsi yang mendukung pemrosesan yang diperlukan dalam ranah pemasalahan.



Gambar 6.13 Diagram struktur untuk sistem tradisional.

Sumber : (Roger S. Pressman, 2012)



Gambar 6.14 Perancangan pada peringkat komponen untuk ComputerPageCost.

Sumber : (Roger S. Pressman, 2012)

Pandangan yang Berhubungan dengan Proses
 Komponen merupakan suatu katalog dari perencanaan yang sudah terbukti dari komponen – komponen pada peringkat kode yang telah tersedia pada saat perancangan berlangsung.

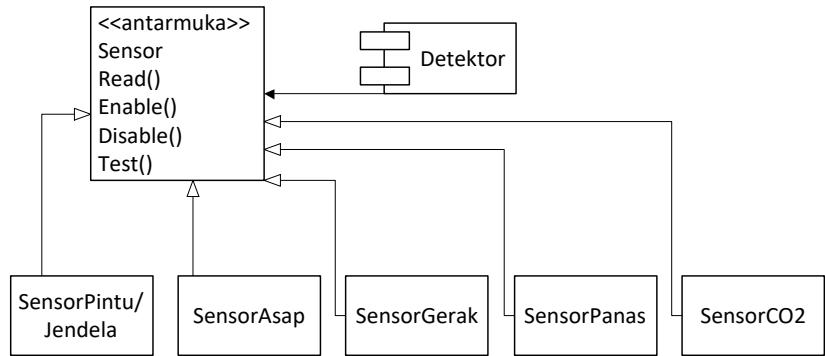
6.2.I. Perancangan Komponen – komponen Berbasis Kelas

Pada pendekatan rekayasa perangkat lunak berbasis objek, perancangan pada peringkat komponen difokuskan pada elaborasi kelas – kelas yang bersifat spesifik terhadap ranah permasalahan dan pada definisi penghalusan kelas – kelas infrastruktur yang ada dalam model kebutuhan. Deskripsi detail dari atribut – atribut , operasi – operasi dan antarmuka – antramuka yang digunakan oleh kelas – kelas diperlukan sebagai syarat melakukan aktivitas konstruksi.

I. Prinsip – prinsip Perancangan Dasar

Ada empat prinsip dasar yang dapat diterapkan dalam perancangan peringkat-komponen, dimana prinsip ini didasari tujuan untuk membuat perancangan – perancangan yang memungkinkan dilakukannya perubahan – perubahan dan mengurangi imbas saat perubahan itu terjadi. Berikut adalah prinsip – prinsip dasar :

- a. OCP (*Open-Closed Principle*) , sebuah modul (komponen) seharusnya terbuka untuk perluasan tetapi tertutup bagi modifikasi.



Gambar 6.15. OCP

Sumber : (Roger S. Pressman, 2012)

- b. Liskov Substitution Principle (LSP), subkelas – subkelas seharusnya dapat didistribusi untuk kelas – kelas dasar mereka. Bahwa komponen – komponen yang menggunakan kelas dasar seharusnya dapat terus berfungsi dengan baik jika suatu kelas yang diturunkan dari kelas dasar dimasukkan pada komponen – komponen itu.
- c. Dependency Inversion Principle (DIP), bergantung pada abstaksi – abstraksi jangan bergantung pada kesatuan. Bahwa semakin sebuah komponen bergantung pada komponen lainnya , maka semakin

sukar komponen tersebut untuk diperluas.

- d. Interface Segregation Principle (ISP), banyak antarmuka-antarmuka yang bersifat spesifik terhadap klien lebih baik daripada satu antarmuka besar yang memiliki kegunaan yang bersifat umum. Pada prinsip ini disaran kan agar kita membuat antarmuka yang bersifat khusus untuk melayani setiap kategori klien.

2. Kohesi

Kohesi adalah komponen – komponen yang berpikiran tunggal. Ada beberapa jenis kohesi yang disampaikan oleh lethbridge dan laganiere dalam (Roger S. Pressman, 2012) :

- a. Fungsional, diperlihatkan oleh operasi – operasi , peringkat kohesi ini sesungguhnya terjadi saat sebuah komponen melakukan komputasi sesuai dengan yang diharapkan kemudian mengembalikan suatu hasil tertentu.
- b. Lapisan , diperlihatkan oleh paket – paket, komponen – komponen dan kelas – kelas , jenis kohesi ini terjadi ketika lapisan yang lebih tinggi mengakses layanan – layanan lapisan yang lebih bawah, tetapi lapisan yang lebih bawah tidak mengakses lapisan yang lebih tinggi.

c. Komunikasional, semua operasi yang mengakses data yang sama didefinisikan dalam satu kelas.

3. Derajat Keterhubungan Antarkomponen

Derajat keterhubungan antarkomponen adalah pengukuran kualitatif tentang derajad bagaimana kelas – kelas saling terhubung satu dengan yang lainnya. Derajat keterhubungan dapat memanifestasikan dirinya dengan berbagai cara . Lethbridge dan laganiere dalam (Roger S. Pressman, 2012) mendefinisikan kategori – kategori sebagai berikut :

- a. Keterhubungan isi, terjadi saat suatu komponen secara paksa memodifikasi data yang bersifat internal dalam komponen yang lainnya.
- b. Keterhubungan umum, terjadi saat sejumlah komponen secara bersamaan menggunakan peubah yang bersifat global.
- c. Keterhubungan kendali , terjadi saat operasi A() memanggil operasi B() dan kemudian A() melewatkkan tanda kendali ke operasi B()
- d. Keterhubungan data, terjadi saat operasi – operasi melewatkkan string yang berukuran panjang sebagai argumen data.
- e. Keterhubungan pemanggilan rutin, terjadi saat suatu operasi memanggil operasi yang lainnya.

- f. Keterhubungan penggunaan tipe data, terjadi saat komponen A menggunakan tipe data yang didefinisikan ke dalam komponen B.
- g. Keterhubungan inklusi atau impor , terjadi saat komponen A mengimpor atau melibatkan sebuah paket atau isi dari komponen B.
- h. Keterhubungan eksternal, terjadi saat suatu komponen berkomunikasi atau berkolaborasi dengan komponen - komponen infrastruktur.

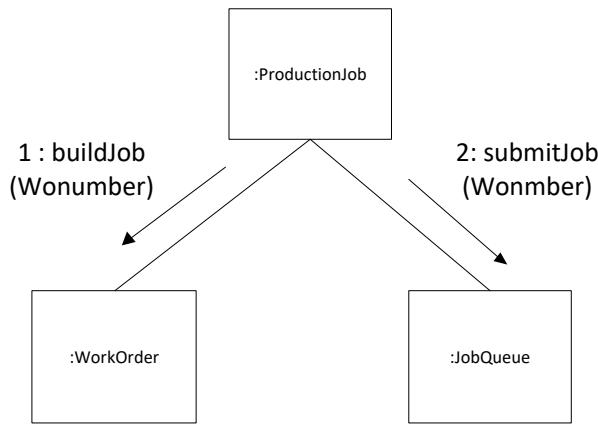
6.2.2. Melakukan Perancangan pada Peringkat Komponen

Langkah – langkah perancangan pada peringkat komponen adalah sebagai berikut (Roger S. Pressman, 2012) :

1. Mengidentifikasi semua kelas – kelas perancangan yang berhubungan dengan ranah permasalahan.
2. Mengelaborasi semua kelas – kelas perancangan yang berhubungan dengan infrastruktur.
3. Mengelaborasi semua kelas – kelas yang tidak diperlukan sebagai kelas – kelas yang dapat digunakan ulang
4. Spesifikasi rincian pesan saat kelas – kelas atau komponen berkolaborasi.
5. Mengidentifikasi antarmuka – antarmuka yang sesuai untuk masing – masing komponen

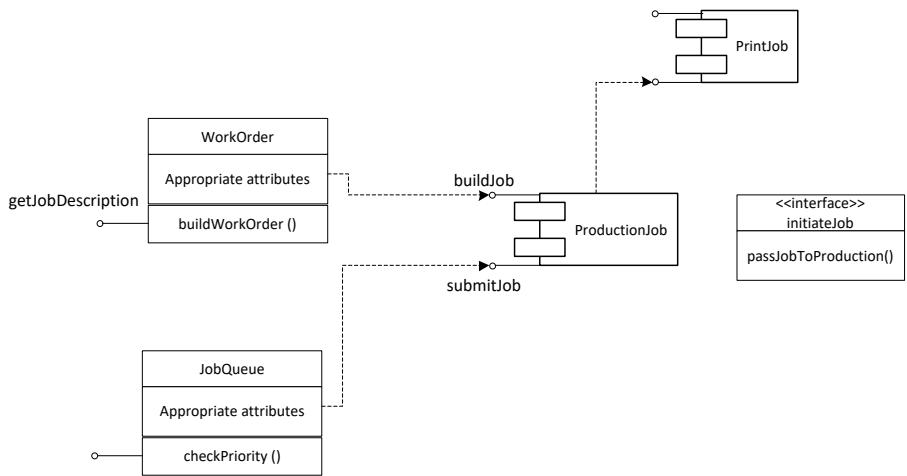
6. Elaborasi atribut – atribut dan definisikan tipe data dan struktur data yang diperlukan untuk mengimplementasikan
7. Deskripsikan aliran pemrosesan di dalam masing – masing operasi secara rinci.
8. Mendeskripsikan sumber – sumber data presisten (basis data dan berkas – berkas dan mengidentifikasi kelas – kelas yang dibutuhkan untuk mengelolanya)
9. Mengembangkan dan mengelaborasikan representasi – representasi perilaku untuk kelas atau komponen.
10. Elaborasi diagram penyebaran komponen (deployment) untuk menyediakan rincian implementasi tambahan.
11. Lakukan refaktorisasi setiap representasi perancangan pada peringkat komponen dan pertimbangkanlah suatu alternatif – alternatif yang mungkin.

Berikut adalah contoh – contoh representasi dari perancangan pada peringkat komponen :



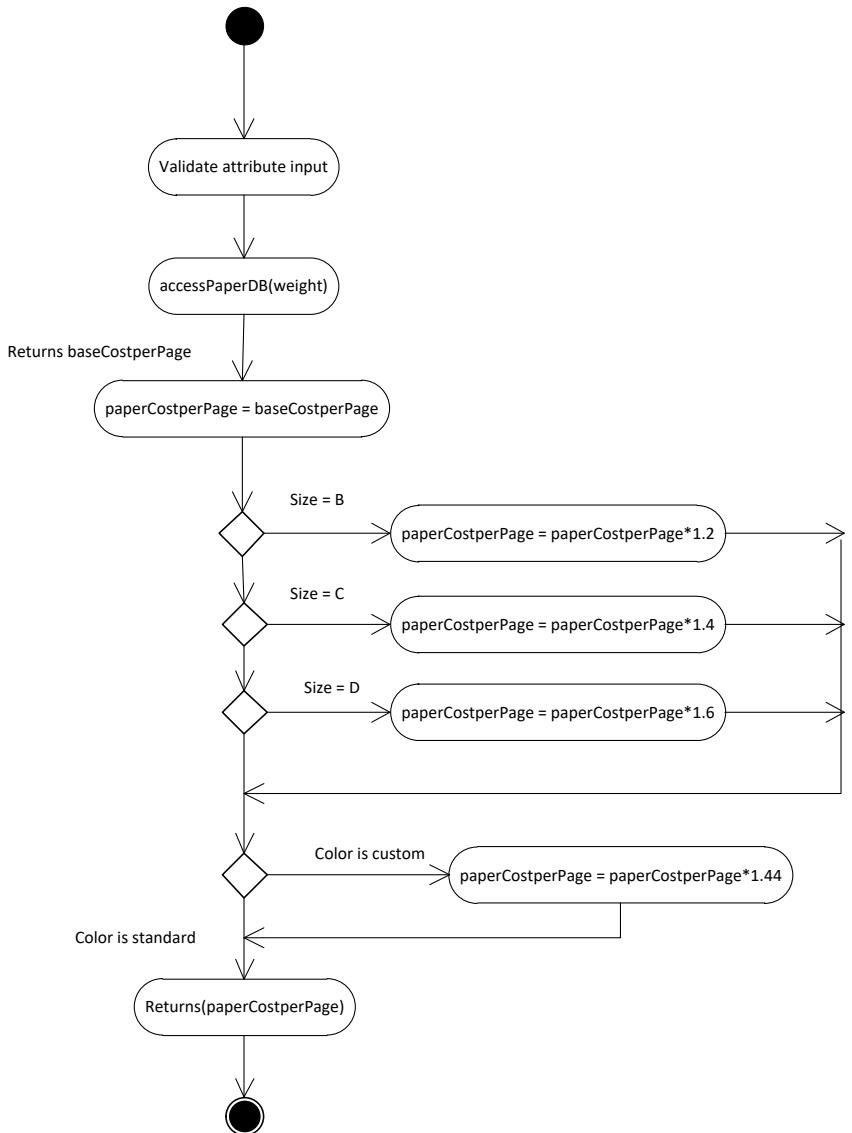
Gambar 6.16. Diagram Kolaborasi dengan pengiriman pesan.

Sumber : (Roger S. Pressman, 2012)



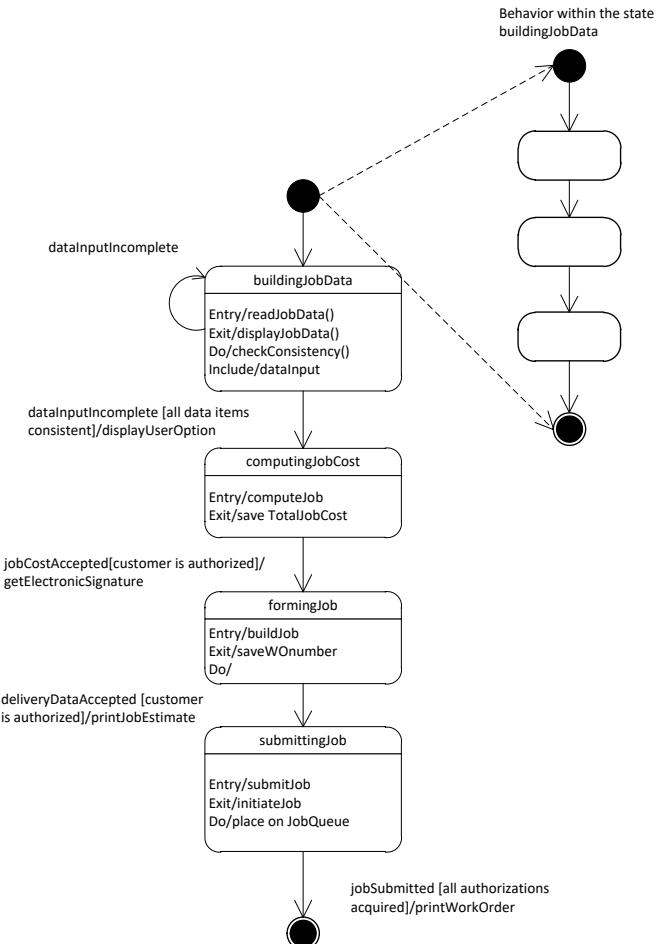
Gambar 6.18. Faktorisasi antarmuka dan
definisi kelas – kelas untuk PrintJob

Sumber : (Roger S. Pressman, 2012)



Gambar 6.19 Diagram aktivitas UML untuk
computePaperCost()

Sumber : (Roger S. Pressman, 2012)



Gambar 6.20 Statechart untuk kelas printJob
Sumber : (Roger S. Pressman, 2012)

6.2.3. Perancangan Peringkat Komponen untuk Aplikasi – aplikasi Web

- I. Perancangan isi pada Peringkat Komponen
Perancangan isi pada peringkat komponen pada dasarnya berfokus pada objek – objek isi dalam arti cara pengemasan yang digunakan dalam aplikasi – aplikasi web yang akan dilihat oleh pengguna akhir.
2. Perancangan Fungsional pada Peringkat Komponen
Fungsionalitas aplikasi web dapat dikembangkan sebagai urutan komponen – komponen secara paralel dimana arsitektur informasinya dapat digunakan untuk memastikan bahwa semua konsisten. Itu dapat dilakukan dengan cara mempertimbangkan baik model – model kebutuhan maupun arsitektur informasi awal dan kemudian memeriksa bagaimana fungsionalitas berakibat tertentu pada interaksi pengguna dengan aplikasi web yang akan dikembangkan, berakibat tertentu pada informasi yang akan ditampilkan dan berakibat pada pekerjaan – pekerjaan yang akan dilaksanakan oleh pengguna.

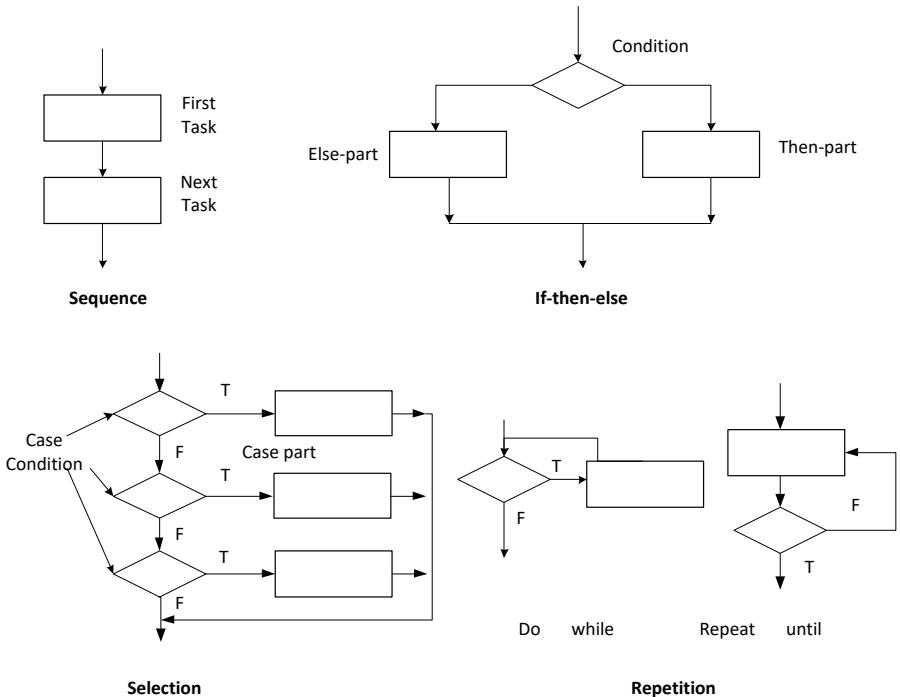
6.2.4. Perancangan Komponen – komponen Tradisional

Dasar perancangan pada peringkat komponen untuk komponen – komponen tradisional telah ditemukan lama oleh edsgar dijkstra dan kawan – kawan. Mereka

mengusulkan sejumlah konstruksi logika yang terbatas darimana setiap program komputer itu dapat dibentuk. Konstruksi memberikan tekanan pada pemeliharaan ranah – ranah fungsional. Konstruksi bersifat *squence*, *condition*, *repetition*. Setiap program komputer , bagaimanapun kompleksnya secara teknis program tersebut dapat dirancang dan diimplementasikan menggunakan sifat konstruksi tersebut yaitu *squence* , *condition*, *repetition* .

I. Notasi Perancangan Secara Grafis

Untuk dapat merepresentasikan konstruksi untuk perancangan dan implementasi dapat digunakan diagram aktivitas yang memungkinkan merepresentasikan perintah *squence* , *condition*, *repetition*. Dimana diagram akativitas merupakan turunan dari diagram alir *flowchart*.



Gambar 6.21 Konsruksi diagram alir

Sumber : (Roger S. Pressman, 2012)

2. Bahasa Perancangan Program

Bahasa perancangan pemrograman disebut sebagai bahasa inggris terstruktur , yang menggabungkan struktur logika bahasa pemograman dengan kemampuan bahasa inggris untuk membentuk pernyataan ekspresif yang berbentuk bebas. Sintak pada bahasa perancangan program melibatkan konstruksi – konstruksi untuk pendefinisian komponen yang

dapat digunakan untuk mendeskripsikan antarmuka – antarmuka, dapat juga digunakan untuk melakukan pendeklarasian data, dapat juga digunakan untuk melakukan penstrukturkan kondisi, untuk mengkontruksi pengulangan , dan dapat digunakan untuk mnegkonstruksi masukan-keluaran.

6.3. Perancangan Antarmuka Pengguna

Perancangan antarmuka pengguna berprinsip untuk menciptakan media komunikasi yang efektif antara pengguna dengan komputer. Karena jika suatu perangkat lunak sukar digunakan maka akan menuntun pada kesalahan – kesalahan. Antarmuka pengguna sangat penting karena membentuk persepsi para pengguna terhadap suatu perangkat lunak.

6.3.I. Aturan – aturan Emas

Teo Mandel dalam (Roger S. Pressman, 2012) menekankan tiga aturan emas dalam perancangan antarmuka pengguna :

- I. Tempatkan pengguna sebagai pengendali
Untuk menempatkan pengguna sebagai pengendali perlu dipahami prinsip – prinsip sebagai berikut :
 - a. Definisikan modus – modus interaksi dengan tidak memaksakan pengguna untuk melakukan aksi – aksi yang tidak diperlukannya dan tidak dikehendakinya.
 - b. Sediakan interaksi yang bersifat fleksible
 - c. Memungkinkan interaksi pengguna yang dapat diinterupsi atau dibatalkan

- d. Menyediakan modus interaksi yang bersifat lanjut dan memungkinkan interaksi dicustomisasi
 - e. Sembunyikan hal – hal yang bersifat sangat teknis dari para pengguna biasa
 - f. Rancang interaksi langsung dengan objek – objek tampak di layar monitor pengguna.
2. Kurangi beban memori di komputer pengguna
Untuk mengurangi beban memori di komputer pengguna perlu dipahami prinsip – prinsip sebagai berikut :
- a. Mengurangi permintaan memori jangka pendek dikomputer pengguna
 - b. Menetapkan nilai – nilai default yang bermakna
 - c. Mendefinikan tombol – tombol pemercepat (shortcut) yang intuitif
 - d. Tata letak visual antarmuka seharusnya berbasis pada metafora – metafira yang memang ada di dunia nyata
 - e. Tampilan informasi – informasi dalam bentuk yang progresif
3. Buat antarmuka sedemikian rupa sehingga bersifat konsisten
Untuk membuat antarmuka yang bersifat konsisten perlu dipahami prinsip – prinsip sebagai berikut :
- a. Memungkinkan pengguna untuk meletakkan pekerjaan saat ini ke dalam suatu konteks yang bermakna

- b. Memelihara konsistensi melintas sejumlah aplikasi yang serupa
- c. Jika model – model interaktif sebelumnya telah dibuat berdasarkan harapan – harapan pengguna, jangan membuat perubahan – perubahan apa pun kecuali ada alasan yang kuat untuk melakukannya.

6.3.2. Analisis dan Perancangan Antarmuka Pengguna

Proses dari analisis dan perancangan antarmuka pengguna diawali dengan pembuatan model yang sebagaimana terlihat dari luar sistem. Prinsip pengembangan antarmuka dapat dibuat dengan cara menggambarkan pekerjaan – pekerjaan yang berorientasi pada manusia dan komputer untuk mendapatkan suatu fungsi tertentu untuk menyelesaikan permasalahan.

Ada empat model analisis yang digunakan dalam merancang antarmuka yaitu :

- 1. Model pengguna, digunakan untuk mendapatkan profil para pengguna akhir dari sistem.
- 2. Model perancangan
- 3. Model mental / perspsi sistem, merupakan gambaran sistem yang di kepala para pengguna akhir .

4. Model implementasi, merupakan gabungan manifestasi yang tampak dari luar suatu sistem yang mendeskripsikan sintaks – sintaks dan sistematik – sistematik antarmuka pengguna.

Proses analisis pada perancangan untuk antarmuka pengguna bersifat iteratif yang direpresentasikan menggunakan mode spiral. Dalam model spiral tersebut ada empat aktivitas kerangka kerja yang jelas yaitu :

1. Pemodelan dan analisis antarmuka pengguna , aktivitas ini berfokus pada profil para pengguna yang akan berinteraksi dengan perangkat lunak.
2. Perancangan antarmuka pengguna, aktivitas ini mendefinisikan sejumlah objek antarmuka pengguna dan aksi – aksi yang memungkinkan pengguna untuk melakukan semua pekerjaan yang telah didefinisikan dengan tujuan untuk menyesuaikan dengan setiap sasaran kegunaan perangkat lunak yang sebelumnya telah didefinisikan.
3. Konstruksi antarmuka pengguna, kegiatan ini dimulai dengan membuat prototipe – prototipe yang memungkinkan skenario – skenario pengguna untuk di evaluasi.
4. Validasi antarmuka pengguna , aktivitas ini berfokus pada :
 - a. Kemampuan antarmuka pengguna untuk mengimplementasikan setiap pekerjaan pengguna secara benar, untuk mengakomodasi semua variasi pekerjaan, dan untuk mencapai semua kebutuhan –

- kebutuhan pengguna yang bersifat umum.
- b. Derajat kemudahan penggunaan antarmuka pengguna dan kemudahannya untuk dipelajari.
 - c. Penerimaan pengguna pada suatu antarmuka sebagai perkakas yang bermanfaat dalam pekerjaan mereka.

6.3.3. Analisis Antarmuka

Dalam analisis antarmuka ada empat hal yang harus dipahami yaitu :

I. Analisis Pengguna

Analisis pengguna dilakukan dengan memahami para pengguna perangkat lunak sebagaimana mereka akan menggunakan perangkat lunak yang akan dikembangkan oleh rekayasa perangkat lunak. Ada beberapa sumber yang dapat digunakan untuk dapat memahami para pengguna :

- a. Wawancara – wawancara dengan pengguna
- b. Asupan – asupan dari stakeholder terkait

2. Analisis pekerjaan dan pemodelan

Untuk menganalisis pekerjaan dan pemodelan dapat dilakukan dengan menggunakan teknik – teknik berikut :

- a. Use case
- b. Elaborasi pekerjaan
- c. Elaborasi objek
- d. Analisis aliran kerja

- e. Representasi hierarki
3. Analisis Tampilan
Kegiatan pada analisis pekerjaan dan pemodelan secara umum memicu dibuatnya presentasi beragam isi yang berbeda. Isi tampilan dapat berupa laporan – laporan, tampilan grafis, infomasi khusus , dan lain – lain.
4. Analisis lingkungan kerja
Analisis ini dilakukan untuk meneliti sedemikian rupa terkait aspek – aspek berkaitan dengan faktor – faktor lingkungan fisik, budaya tempat kerja yang mungkin memainkan peran sangat penting. Sehingga interaksi dengan sistem dapat berjalan dengan baik.

6.3.4. Langkah – langkah Perancangan Antarmuka

Ada banyak model perancangan antarmuka pengguna, pada dasarnya langkah – langkah yang digunakan adalah sebagai berikut :

Menggunakan informasi yang dikembangkan selama analisis antarmuka pengguna , definiskan objek – objek dna tindakan – tindakan atau operasi – operasi.

Definiskan event – event yang akan menyebabkan keadaan antarmuka pengguna berubah. Lakukan pemodelan untuk perilaku ini.

Perlihatkan masing – masing keadaan antarmuka pengguna seperti yang pada waktunya nanti akan dilihat oleh para pengguna.

Perkirakan bagaimana para pengguna akan menafsirkan keadaan sistem dari informasi – informasi yang disediakan melalui antarmuka pengguna. (Roger S. Pressman, 2012)

I. Menerapkan langkah – langkah perancangan antarmuka

Langkah dalam perancangan antarmuka pengguna diawali dengan mendefinisikan objek – objek antarmuka pengguna dan tindakan – tindakan yang akan di terapkan . aktivitas tersebut dapat direpresentasikan ke dalam use case. Setelah kegiatan tersebut dilakukan elaborasi secara iteratif pada masing – masing objek dan tindakan .

Setelah objek dan tindakan pada iterasi tertentu di definisikan dengan maksimal maka langkah selanjutnya salah proses perancangan tata letak objek pada layar sesungguhnya, pendefinisian teks diskriptif pada layar, dan pemberian judul pada jendela – jendela dan item – item menu.

2. Permasalahan – permasalahan yang berkaitan dengan perancangan.

Ketika perancangan antarmuka pengguna dilakukan , biasanya ada beberapa permasalahan umum yaitu (Roger S. Pressman, 2012) :

a. Waktu tanggap

Waktu tanggap diukur dari titik dimana para pengguna melakukan tindakan – tindakan pengendalian tentu hingga perangkat lunak menanggapinya dengan

- menghasilkan keluaran. Jika tanggapannya lama maka pengguna akan kecewa.
- b. Fasilitas – fasilitas bantuan
Hampir semua pengguna perangkat lunak membutuhkan fasilitas bantuan. Untuk itu banyak sekarang perangkat lunak yang dilengkapi dengan manual pengguna atau online support.
 - c. Penangan kesalahan – kesalahan
Pesan – pesan kesalahan merupakan peringatan buruk bagi pengguna , karena akan membuat pengguna merasa kecewa. Untuk itu perlu dirancang pesan kesalahan yang dapat meningkatkan kualitas perangkat lunak dan yang dapat mengurangi rasa kecewa pengguna.
 - d. Menu – menu dan label peritah – perintah
Ada beberapa perintah yang mungkin berbeda cara antara aplikasi yang satu dengan yang lain, perbedaan – perbedaan ini akan membuat pengguna bingung. Untuk itu akan lebih baik menggunakan tombol menu dengan representasi gambar sehingga akan memudahkan pengguna dalam pengoperasiannya.
 - e. Kemudahan suatu aplikasi untuk diakses
Kemudahan akses untuk para pengguna yang mungkin memiliki keterbatasan – keterbatasan secara fisik sangat penting. Untuk itu diperlukan sebuah panduan

- dan perancangan yang mempertimbangkan pengguna.
- f. Internasionalisasi
Untuk menjawab globalisasi tentunya sebuah perangkat lunak harus dapat menyediakan konteks bahasa – bahasa yang dapat dimengerti oleh seluruh pengguna dari berbagai negara.

6.3.5. Perancangan Antarmuka Aplikasi – aplikasi Web

Ada beberapa hal penting yang harus diketahui dalam merancang antarmuka aplikasi – aplikasi web yaitu (Roger S. Pressman, 2012):

Antarmuka pengguna untuk aplikasi – aplikasi web harus dapat memberikan tanda – tanda tertentu pada objek – objek yang pernah diakses dan memberikan informasi pada para pengguna tentang hal – hal yang berkaitan dengan lokasi mereka pada hierarki isi yang dimiliki oleh aplikasi web yang bersangkutan.

Antarmuka pengguna untuk aplikasi – aplikasi web seharusnya dapat selalui membantu pengguna untuk memahami pilihan – pilihan mereka saat ini yaitu fungsi – fungsi yang tersedia , tautan – tautan yang dapat digunakan , dan isi – isi apa yang relevan.

Antarmuka pengguna untuk aplikasi – aplikasi web seharusnya memfasilitasi navigasi atau

penelusuran yang akan dilakukan oleh para penggunanya.

Ada beberapa prinsip yang dapat menjadi panduan dalam perancangan antarmuka adalah sebagai berikut (Roger S. Pressman, 2012) :

1. Antisipasi, suatu aplikasi web seharusnya dirancang sedemikian rupa sehingga ia dapat mengantisipasi pergerakan pengguna di waktu – waktu berikutnya.
2. Komunikasi, antarmuka pengguna yang dimiliki suatu aplikasi web seharusnya mengkomunikasikan status dari setiap aktivitas yang dilakukan oleh para pengguna.
3. Konsistensi, penggunaan kontrol – kontrol navigasi , menu – menu , ikon – ikon dan estetika seharusnya bersifat konsisten di seluruh bagian aplikasi web.
4. Otonomi terkendali, antarmuka pengguna seharusnya dapat memfasilitasi pergerakan para pengguna ke seluruh bagian aplikasi web, tetapi antarmuka pengguna yang bersangkutan seharusnya melakukannya dengan cara menyesuaikannya dengan konvensi – konvensi navigasi yang telah ditetapkan sebelumnya untuk suatu aplikasi web.
5. Efisiensi, perancangan suatu aplikasi web dan antarmuka penggunanya seharusnya mengoptimalkan tingkat efisiensi pekerjaan pengguna, bukan mengoptimalkan tingkat efisiensi pengembang yang merancang dan

membangunnya atau mengoptimalkan tingkat efisiensi lingkungan klien-server yang mengeksekusinya.

6. Fleksibilitas, antarmuka pengguna seharusnya bersifat cukup fleksibel sehingga memungkinkan banyak pengguna dapat menyelesaikan pekerjaan – pekerjaannya secara langsung dan memungkinkan pengguna – pengguna yang lainnya dapat mengeksplorasi aplikasi web menggunakan cara yang bersifat acak.
7. Fokus, antarmuka pengguna aplikasi web seharusnya tetap fokus pada pekerjaan – pekerjaan yang akan dilakukan oleh para pengguna.
8. Hukum Fitt, waktu untuk mencapai suatu target tertentu sesungguhnya merupakan fungsi dari jarak dan ukuran target itu.
9. Objek – objek antarmuka manusia, sejumlah besar pustaka yang berisi objek – objek antarmuka manusia yang dapat digunakan ulang telah dikembangkan untuk aplikasi – aplikasi web. Gunakanlah !
10. Pengurangan waktu pemrosesan, alih – alih membuat para pengguna menunggu berjalannya beberapa operasi internal untuk diselesaikan , aplikasi – aplikasi web seharusnya menggunakan fasilitas dan kemampuan sistem operasi untuk mengerjakan beberapa pekerjaan sekaligus sedemikian rupa sehingga para pengguna bisa melanjutkan pekerjaanya yang lain sambil menunggu suatu operasi internal dilaksanakan

hingga selesai oleh aplikasi web yang bersangkutan.

11. Kemudahan dipelajari , suatu antarmuka pengguna yang dimiliki oleh aplikasi – aplikasi web seharusnya dirancang sedemikian rupa sehingga mudah dipelajari dan setelah dipelajari akan meminimalkan waktu untuk melakukan pembelajaran kembali saat aplikasi – aplikasi web yang bersangkutan kembali di kunjungi di masa – masa selanjutnya.
12. Metafora – metafora , suatu antarmuka untuk aplikasi web yang menggunakan metafora interaksi akan bersifat lebih mudah untuk dipelajari dan akan lebih mudah digunakan, sepanjang metafora interaksi yang digunakan sesuai untuk aplikasi web yang bersangkutan serta sesuai juga dengan para penggunanya.
13. Memelihara integritas produk kerja, suatu produk kerja harus secara otomatis disimpan sehingga apa yang ada di dalamnya tidak akan hilang jika terjadi kesalahan – kesalahan .
14. Kemudahan dibaca , semua informasi yang ditampilkan melalui antarmuka pengguna suatu aplikasi web seharusnya mudah dibaca oleh pengguna – pengguna yang berusia muda maupun meraka yang sudah tua.
15. Melacak keadaan pengguna, jika dimungkinkan , keadaan suatu interaksi pengguna seharunya dapat dilacak dan disimpan sedemikian rupa sehingga seorang pengguna bisa keluar dari aplikasi web dan dikemudian hari dapat masuk

kembali ke aplikasi web yang sama pada titik yang pernah di tinggalkan sebelumnya.

16. Navigasi yang tampak, sebuah antarmuka pengguna yang dimiliki oleh aplikasi web yang dirancang dengan baik pada dasranya menyediakan ilusi – ilusi seolah – olah para pengguna aplikasi web berada di tempat yang sama dengan pekerjaan – pekerjaan yang diserahkan pada mereka.

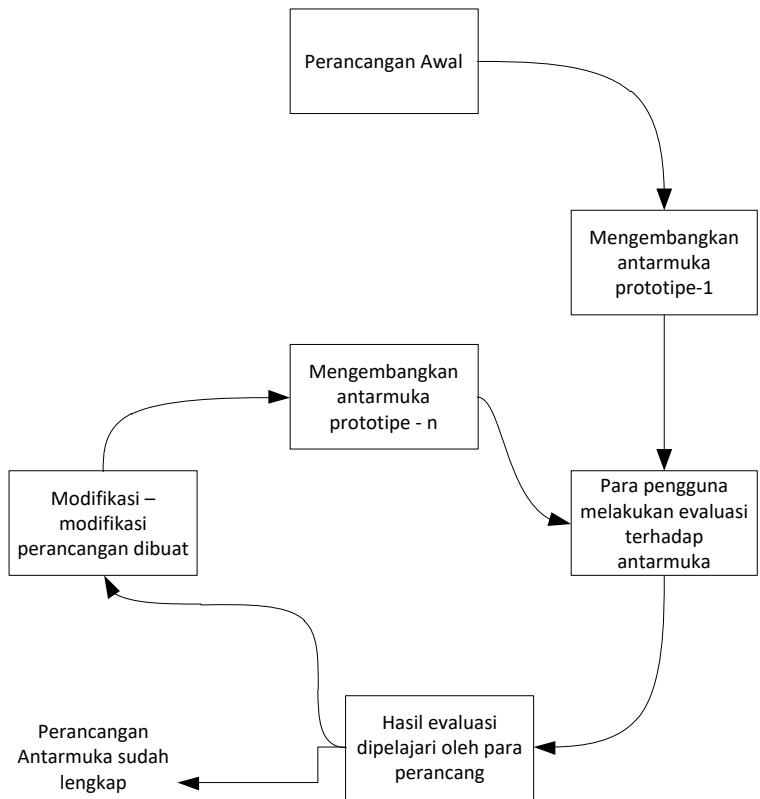
Sedangkan aliran kerja perancangan antarmuka aplikasi – aplikasi web dapat dikerjakan sebagai berikut (Roger S. Pressman, 2012) :

1. Lakukan peninjauan terhadap informasi – informasi yang ada di dalam model – model keutuhan dna jika dierlukan lakukan penghalusan – penghalusan.
2. Kembangkan sebuah sketsa kasar tentang tata letak antarmuka aplikasi web.
3. Petakan sasaran – sasaran pengguna ke dalam aksi – aksi antarmuka secara spesifik.
4. Definisikan himpunan pekerjaan para pengguna yang berhubungan dengan masing – masing aksi.
5. Buat gambar layar untuk masing – masing aksi pada antarmuka pengguna.
6. Lakukanlah penghalusan pada tata letak antarmuka pengguna dna gambaran – gambaran layar yang menggunakan pertimbangan – pertimbangan estetika.

7. Identifikasi objek – objek antarmuka pengguna yang diperlukan untuk mengimplementasikan antarmuka pengguna
8. Kembangkan sebuah representasi prosedural interaksi pengguna dengan antarmuka pengguna
9. Kembangkan sebuah representasi perilaku untuk antarmuka pengguna
10. Deskripsikan tata letak antarmuka pengguna untuk masing – masing keadaan
11. Perbaiki dan tinjau model perancangan antarmuka pengguna.

6.3.6. Evaluasi Terhadap Perancangan Antarmuka Pengguna

Setelah membuat prototipe antarmuka pengguna yang bersifat operasional , maka langkah selanjutnya adalah mengevaluasi perancangan antarmuka pengguna. Evaluasi dapat menggunakan beberapa teknik misalnya menggunakan kuesioner – kuesioner. Evaluasi menggunakan pendekatan prototipe antarmuka pengguna memiliki siklus sebagai berikut :



Gambar 6.22 Siklus evaluasi perancangan antarmuka

Sumber : (Roger S. Pressman, 2012)

6.4. Perancangan Aplikasi – aplikasi Web

Pada dasarnya perancangan aplikasi – aplikasi web membutuhkan dua aspek yaitu aspek teknis dan non teknis. Ada beberapa hal penting yang perlu diperhatikan dalam perancangan aplikasi – aplikasi web yaitu penetapan interface aplikasi – aplikasi web, rancangan estetika untuk interface pengguna, pendefinisan struktur arsitektur aplikasi web secara keseluruhan, pengembangan isi dna fungsionalitas pada arsitektur aplikasi, dan perencanaan navigasi untuk aplikasinya.

6.5.I. Kualitas Perancangan Aplikasi – aplikasi Web

Sebenarnya , jika berbicara kualitas , kebanyakan orang akan memiliki perspektif yang berbeda – beda . Namun ada hal – hal yang pasti harus diuji untuk menentukan kualitas aplikasi web tersebut. Olsina dalam (Roger S. Pressman, 2012) mengembangkan pohon penilaian kualitas sebagai berikut :



Gambar 6.23 Siklus evaluasi perancangan antarmuka

Sumber : (Roger S. Pressman, 2012)

6.5.2. Sasaran – sasaran Perancangan

Sejumlah sasaran perancangan yang dapat diterapkan pada sebuah aplikasi web adalah sebagai berikut :

1. Kesederhanaan

Content dalam sebuah website haruslah bersifat informatif , ringkas , tidak berlebih dan menggunakan metode penyampaian yang sesuai dengan informasi yang akan disampaikan.

2. Konsistensi

Estetika , arsitektural aplikasi, modus – modus interaksi, navigasi dan tampilan isi harus konsisten, karena jika tidak konsisten maka akan membingungkan .

3. Identitas

Suatu aplikasi pastilah ditunjukkan untuk pengguna tertentu , maka sebagai perancang seharusnya menentapkan identitas aplikasi yang sedang dikembangkan , agar memiliki karakteristik.

4. Ketangguhan

Ketangguhan merupakan relevansi antara isi dan fungsi dari aplikasi dengan kebutuhan pengguna.

5. Kemudahan untuk melakukan navigasi dalam aplikasi web

Aplikasi – aplikasi web seharusnya dirancang sedemikian rupa sehingga tampilan dan hasilnya dapat mudah diramalkan.

6. Daya Tarik Visual

Perancangan pengguna, pengaturan warna , keseimbangan teks, grafik dan media – media lainnya sangat memiliki kontribusi pada daya tarik visual.

7. Kompatibilitas

Aplikasi web akan di implementasikan dilingkungan eksekusi aplikasi yang berbeda – beda, maka aplikasi web harus di rancang sesuai dengan lingkungan eksekusi aplikasi.

6.5.3. Model Perancangan Berbentuk Piramida untuk Aplikasi – aplikasi Web

Berikut adalah piramida perancangan untuk aplikasi web :



Gambar 6.24 Piramida perancangan untuk
aplikasi web

Sumber : (Roger S. Pressman, 2012)

I. Perancangan Antarmuka untuk Aplikasi – aplikasi Web

Dalam perancangan antarmuka aplikasi web memiliki sasaran sebagai berikut :

- a. Menetapkan suatu jendela yang konsisten untuk meletakkan isi – isi dan fungsionalitas – fungsionalitas yang disediakan oleh antarmuka pengguna.
- b. Memadukan para pengguna melalui serangkaian interaksi dengan aplikasi yang dikembangkan
- c. Mengorganisasikan pilihan – pilihan navigasi dan isi – isi yang dapat dilihat oleh para pengguna.

2. Perancangan Estetika

Perancangan estetika merupakan tambahan artistik yang digunakan untuk melengkapi aspek -aspek teknis dari perancangan aplikasi web. Ada beberapa panduan tata letak secara umum yang mungkin dapat dipertimbangkan oleh para perancangan antar muka berbasis web :

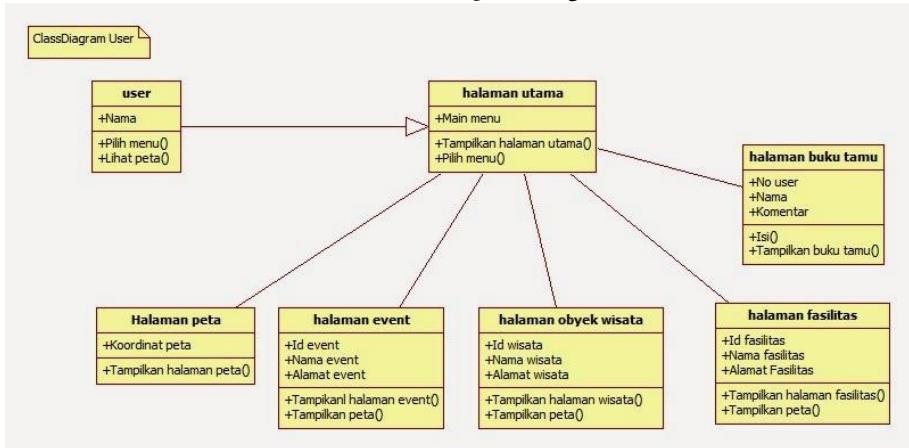
- a. Jangan takut dengan spasi kosong , terlalu banyak isi dalam setiap bagian web , akan mempersulit user dalam memilih informasi
- b. Lakukan penekanan pada isi,
- c. Lakukan pengelompokan fitur – fitur navigasi, isi dan fungsi

d. Pertimbangkan resolusi layar

3. Perancangan Isi

Untuk merancang isi maka perlu dilakukan objek – objek yang terdapat pada website tersebut. Biasanya hubungan antar objek satu dengan objek yang lainnya didefinisikan sebagai bagian model kebutuhan untuk aplikasi web. Hubungan – hubungan asosiasi agregasi yang dikenal dalam pemodelan menggunakan UML dapat digunakan untuk merepresentasikan relasi yang terjadi antara objek isi.

Berikut adalah contoh perancangan isi :



Gambar 6.25 Representasi perancangan objek isi

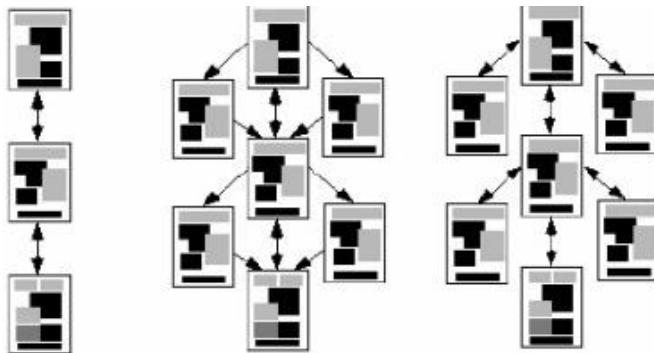
Sumber : (Hamdani, 2014)

4. Perancangan Arsitektural

Perancangan arsitektur web terkait dengan sasaran yang ditetapkan untuk aplikasi tersebut ,

terkait dengan isi yang di *publish*, dan terkait dengan user yang akan menggunjunginya. Dalam perancangan arsitektural terdapat empat struktur isi yaitu :

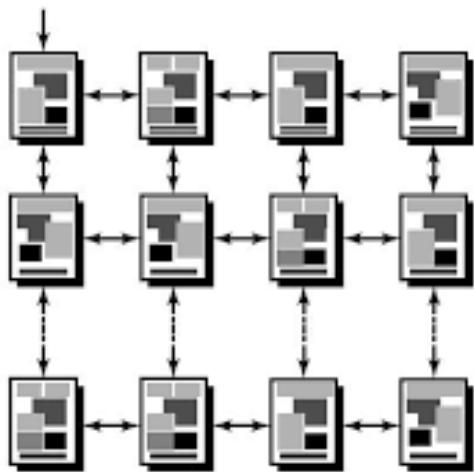
- a. Struktur linier , dilakukan saat interaksi pengguna dengan aplikasi web secara umum yang memperlihatkan urutan yang diramalkan.



Gambar 6.26 Struktur Linier

Sumber : (Roger S. Pressman, 2012)

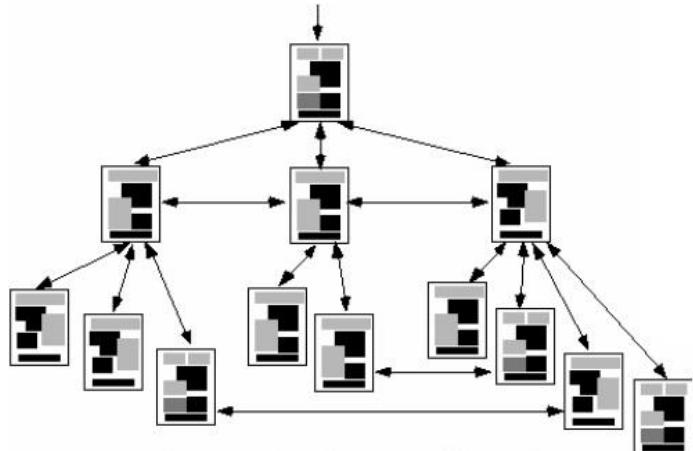
- b. Struktur grid, merupakan suatu pilihan arsitektural yang dapat diterapkan saat isi aplikasi web dapat diorganisasikan menjadi beberapa struktur dimensi.



Gambar 6.27 Struktur Grid

Sumber : (Roger S. Pressman, 2012)

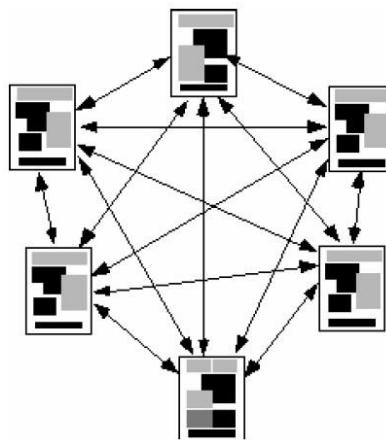
- c. Struktur hirarki , merupakan struktur yang memungkinkan aliran kendali terjadi sepanjang cabang – cabangnya.



Gambar 6.28 Struktur Hirarki

Sumber : (Roger S. Pressman, 2012)

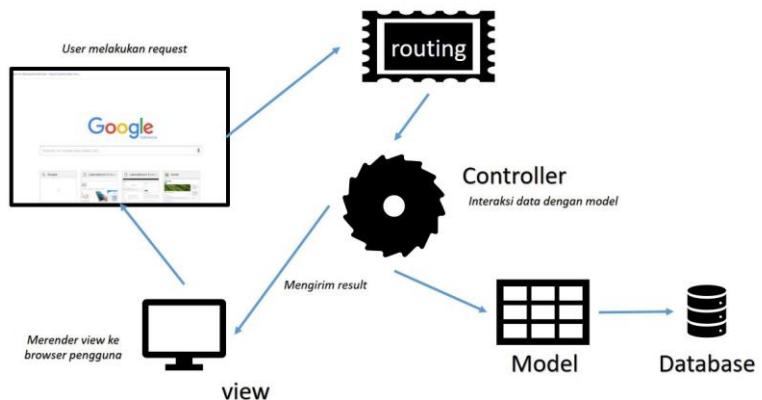
- d. Struktur jaringan, struktur yang memungkinkan terjadinya fleksibilitas penelusuran yang baik, tetapi pada saat yang sama , mungkin saja menimbulkan kebingungan pada sebagian pengguna.



Gambar 6.29 Struktur Jaringan

Sumber : (Roger S. Pressman, 2012)

Berikut adalah contoh arsitektur aplikasi web :



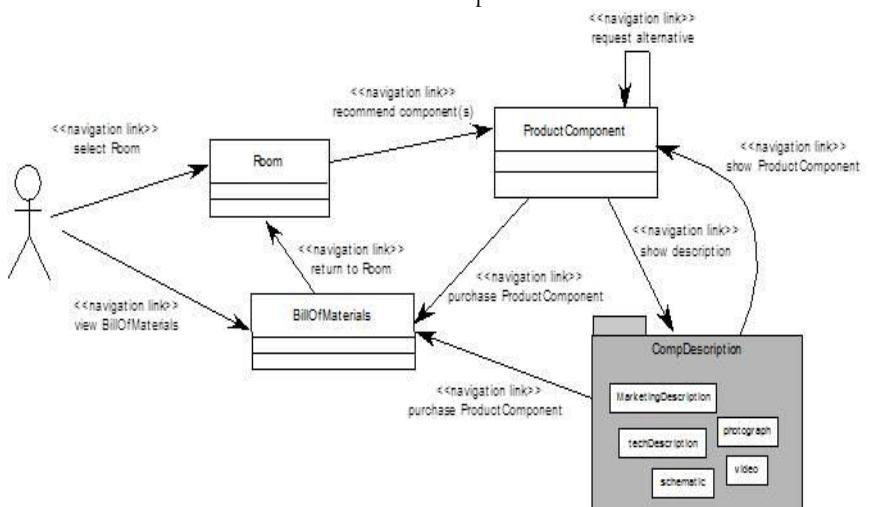
Gambar 6.30 Arsitektur MVC

Sumber : (Dara, 2017)

5. Perancangan Navigasi

Perancangan navigasi bertujuan untuk mengidentifikasi lintasan - lintasan navigasi yang memungkinkan para pengguna mengakses isi dan fungsi - fungsi aplikasi web yang sedang dikembangkan. Dalam perancangan navigasi kita kenal *Navigation Semantic Unit* (NSU) yaitu sejumlah informasi dan struktur navigasi yang terkait yang saling berkolaborasi untuk memenuhi sejumlah kebutuhan pengguna yang bersifat khusus.

Berikut adalah contoh pembuatan NSU :



Gambar 6.31 Pembuatan NSU

Sumber : (Roger S. Pressman, 2012)

6.5.4. Metode Perancangan Hypermedia Berorientasi Objek

Metode perancangan hypermedia berorientasi objek (*Object-Oriented Hypermedia Design Method*) ditemukan oleh Daniel Schwabe dan rekan kerjanya yang pada dasarnya metode ini terdiri dari empat aktivitas perancangan yang berbeda yaitu perancangan konseptual , perancangan struktur navigasi , perancangan antarmuka yang bersifat abstrak, dan implementasinya untuk secara aktual menghasilkan suatu aplikasi yang utuh. Berikut adalah inti dari metode perancangan hypermedia beroorientasi objek :

Tabel 6.1 Intisari dari metode perancangan hypermedia berorientasi objek

	Perancangan konseptual	Perancangan struktur navigasi	Perancangan antarmuka secara abstrak	Implementasi
Produk-produk kerja	Kelas – kelas, subsistem – subsistem, relasi – relasi , atribut – atribut	Tautan – tautan simpul, struktur – struktur akses, konteks – konteks navigasional, transformasi –	Objek – objek antarmuka abstrak, tanggapan – tanggapan atas event – event yang bersifat eksternal, transformasi – transformasi	Aplikasi web yang dapat dieksekusi

		transformasi navigasional		
Mekanisme – mekanisma perancangan	Klasifikasi, komposisi, agregasi, generalisasi, spesialisasi	Pemetaan antar objek-objek konseptual dan navigasi	Pemetaan antara anvgasi dan objek – objek yang akan melaksanakan pemrosesan tertentu	Sumber daya yang disediakan oleh target lingkungan
Perhatian – perhatian perancangan	Pemodelan ranah aplikasi secara semantik	Melakukan pengambilan pada akun profil pengguna dan pekerjaan . menekankan pada aspek – aspek kognitif	Pemodelan objek – objek yang melakukan pemrosesan – pemrosesan tertentu, mengimplementasikan metafora – metafora yang dipilih. Mendeskripsikan antarmuka untuk objek – objek navigasional	Kebenaran , kinerja apalikasi , kelengkapan

Sumber Tabel : (Roger S. Pressman, 2012)

BAB 7 Manajemen Kualitas

Perangkat Lunak

7.I. Konsep – konsep Kualitas

Kualitas merupakan sebuah ukuran terhadap suatu produk , layanan , atau yang lainnya, termasuk untuk perangkat lunak . Perangkat lunak yang tidak berkualitas akan merusak organisasi yang menggunakan Menghilangkan kesempatan untuk melakukan transaksi yang berimbang pada keuntungan organisasi, mengakibatkan tingginya biaya pemeliharaan serta mengakibatkan rendahnya kepuasan pelanggan.

7.I.I. Kualitas ?

Jika kita berbicara tentang kualitas , maka kita akan mendapatkan dua pendekatan terkait kualitas tersebut . Pendekatan pertama berfokus pada standar produk dan jasa yang berarti kesesuaian dengan spesifikasi kebutuhan pelanggan , kesesuaian dengan tujuan dan manfaat di kembangkannya perangkat lunak , menghantarkan perangkat lunak tanpa cacat , dan selalu baik dalam melakukan pengembangan perangkat lunak dari tahap awal. Pendekatan kedua berfokus pada standar pelanggan yaitu kepuasan pelanggan , memenuhi kebutuhan pelanggan dan menyenangkan pelanggan.

Kualitas perancangan merujuk pada karakteristik – karakteristik produk yang dispesifikasi oleh para perancang. Kualitas kesesuaian berfokus pada derajad dimana implemnetasi mengikuti perancangan dan menghasilkan perangkat lunak yang sesuai dengan sasaran kinerja dan kebutuhan. Kualitas perangkat lunak mencakup derajat dimana perancangan memenuhi fungsi – fungsi dan fitur – fitur yang dispesifikasikan melalui model – model kebutuhan. Robert Glass dalam (Roger S. Pressman, 2012) mengemukakan bahwa :

kepuasan pelanggan = produk yang sesuai + kualitas yang bagus + diserahkan sesuai anggaran dan jadwal

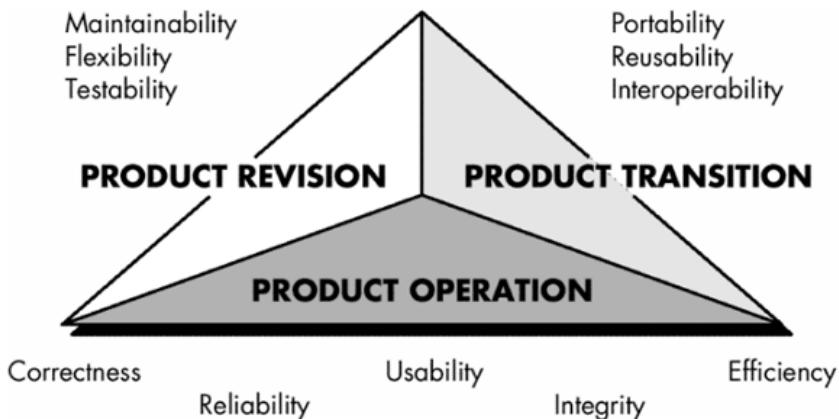
7.1.2. Kualitas Perangkat lunak

Pressman mengemukaan kualitas perangkat lunak ditekankan pada suatu proses perangkat lunak yang efektif , suatu produk yang yang bermanfaat sehingga tercipta nilai pada produsesn maupun pengguna perangkat lunak. Sedangkan Garvin mengemukakan kualitas ditinjau pada kualitas kinerja , kualitas fitur, keandalan, kesesuaian, durabilitas, kemampuan untuk melakukan layanan , estetika , dan persepsi. Berikut adalah beberapa perspektif pengukuran secara kualitatif :

Menurut Mc Call ada beberapa faktor yang mempengaruhi kualitas yaitu :

1. Kebenaran , bagaimana program akan memberikan hasil sesuai dengan spesifikasi yang ditetapkan sebelumnya dan memenuhi sasaran – sasaran pelanggan.
2. Keandalan, bagaimana suatu program diharapkan dapat melakukan fungsi – fungsi tertentu sesuai dengan tingkat ketelitian yang diinginkan.
3. Efisiensi , jumlah sumber daya komputasi dan kode yang diperlukan progarm untuk mampu melaksanakan fungsinya secara baik dan benar.
4. Integritas, bagaimana akses ke perangkat lunak atau ke data oleh orang – orang yang tidak terotorisasi dapat dikendalikan.
5. Penggunaan , besarnya usaha yang diperlukan untuk mempelajari , mengoperasikan, menyediakan input dan menafsirkan output untuk suatu program.
6. Kemampuan untuk dipelihara , besarnya usaha yang diperlukan untuk melokalisasi dan membetulkan kesalahan – kesalahan yang dapat ditemukan dalam program.
7. Fleksibilitas, besarnya usaha yang diperlukan untuk memodifikasi suatu program yang bersifat operasional.
8. Kemampuan untuk menghadapi pengujian, besarnya usaha yang diperlukan untuk melakukan pengujian atas suatu program dengan tujuan untuk memastikan bahwa program itu melaksanakan fungsi yang diharapkan.
9. Portabilitas, besarnya usaha yang diperlukan untuk mentransfer program .

10. Penggunaan ulang, bagaimana suatu program dapat digunakan ulang di aplikasi lainnya.
11. Interoperabilitas, besarnya usaha yang diperlukan untuk menggantikan bagian suatu sistem dengan bagian sistem yang lainnya.



Gambar 7.1 Faktor – faktor kualitas perangkat lunak menurut McCall

Sumber : (Roger S. Pressman, 2012)

Sedangkan menurut ISO 9126 mengidentifikasi atribut kualitas kunci sebagai berikut :

- I. Fungsionalitas , Derajat tentang bagaimana perangkat lunak memenuhi kebutuhan yang telah ditetapkan sebelumnya dan memiliki subatribut – atribut kecocokan, akurasi, interoperabilitas, kesesuaian dan keamanan.

2. Kenadalan, jumlah waktu penggunaan perangkat lunak yang tersedia dan memiliki subatribut ketangan , toleransi kesalahan , kemampuan untuk melakukan pemulihan.
3. Kemudahan pengguna, derajad tentang bagaimana kemudahan perangkat lunak digunakan , dimana hal ini seringkali diindikasikan menggunakan subatribut kemudaan untuk dipahami, kemudahan untuk dipelajari, operabilitas.
4. Efisiensi, derajat penggunaan sumber daya sistem secara optimal dimana hal ini diindikasikan oleh subatribut perilaku waktu, perilaku sumber daya.
5. Kemudahan Pemeliharaan , kemudahan yang menentukan tentang bagaimana perbaikan – perbaikan mungkin dilakukan pada suatu perangkat lunak , dimana hal ini diindikasikan menggunakan subatribut kemampuan untuk melakukan analisis, kemampuan untuk dilakukan perubahan , hal – hal yang berkaitan dengan stabilitas , serta kemampuan untuk dilakukan pengujian.
6. Portabilitas, kemudahan bagaiman perangkat lunak dapat dipindahkan dari suatu lingkuang operasional ke lingkungan operasional lainnya yang hal ini diindikasikan menggunakan sub atribut kemempuan untuk beradaptasi, kemampuan untuk diinstal, kesesuaian, kemampuan untuk digantikan.

Untuk dapat menilai kualitas perangkat lunak , kita dapat melakukan penilaian pada antarmuka – antarmuka pengguna . berikut adalah contoh untuk mengukur kualitas perangkat lunak :

1. Intuitif , Derajat bagaimana antarmuka pengguna mengikuti pola – pola pengguna yang diharapkan sehingga , bahkan para pemula dapat menggunakan tanpa pelatihan yang berlebih.
 - a. Apakah rancangan antarmuka pengguna mudah dipahami ?
 - b. Apakah operasi – operasi yang disediakan antarmuka pengguna mudah dicari dan mudah untuk dilaksanakan ?
 - c. Apakah antar muka pengguna menggunakan metafora yang mudah dikenali ?
 - d. Apakah input dispesifikasi untuk mengurangi penekanan tombol keyboard atau mengurangi jumlah pengeklikan mouse ?
 - e. Apakah antarmuka pengguna , menempatkan pengguna sebagai pengendali ?
 - f. Apakah antarmuka pengguna , mengurangi beban komputer anda ?
 - g. Apakah antarmuka pengguna konsisten ?
 - h. Apakah estetikanya membantu pemahaman dan penggunaan ?
2. Efisiensi , Derajat bagaimana operasi – operasi dan informasi – informasi dapat ditemukan dan dilaksanakan.

- a. Apakah rancangan antarmuka pengguna dan gayanya memungkinkan pengguna untuk melokalisasi operasi – operasi dan informasi secara efisien ?
 - b. Dapatkah urutan operasi – operasi input dapat dilaksanakan dengan gerakan yang seminimal mungkin ?
 - c. Apakah data output atau isi ditampilkan sedemikian rupa sehingga mudah dipahami ?
 - d. Apakah operasi – operasi yang bersifat hirarkies diorganisasikan dengan cara yang kedalamannya minimal sehingga pengguna tidak perlu terlalu dalam melakukan penelusuran untuk mendapatkan operasi yang dikehendaki ?
3. Ketangguhan , Berkaitan dengan kemampuan perangkat lunak untuk menangani input data yang buruk atau menangani interaksi pengguna yang tidak sesuai.
 - a. Akankah perangkat lunak mengenali kesalahan jika data yang dimasukkan salah atau berbeda diluar batasan ? yang lebih penting , jika terjadi hal – hal yang tadi disebutkan , akankah perangkat lunak terus beroperasi tanpa harus mengalami kegagalan atau penurunan kinerja ?
 - b. Akankah antarmuka pengguna mampu mengenali pemahaman umum atau kesalahan – kesalahan menipulatif dan

- secara eksplisit memandu pengguna kembali ke jalur yang benar ?
- c. Apakah antarmuka pengguna menyediakan panduan dan diagnosa yang bermanfaat saat kondisi kesalahan (yang berhubungan dengan fungsionalitas perangkat lunak) tidak tersingkap ?
4. Kekayaan , derajat tentang bagaimana antarmuka pengguna menyediakan sejumlah fitur yang kaya.
- a. Dapatkah antarmuka pengguna dimodifikasi sesuai dengan kebutuhan pengguna ?
 - b. Apakah antarmuka pengguna menyediakan kemampuan makro yang memungkinkan para pengguna aplikasi untuk mengidentifikasi sejumlah operasi umum yang berurutan dengan suatu aksi atau perintah tunggal.

7.1.3. Dilema Kualitas Perangkat Lunak

Berikut adalah beberapa dilema kualitas perangkat lunak :

- I. Perangkat lunak yang cukup bagus , yaitu perangkat lunak yang memiliki fungsi – fungsi dan fitur – fitur berkualitas tinggi sesuai dengan apa yang diinginkan oleh pengguna , tetapi pada saat yang sama memiliki juga fungsi – fungsi serta fitur – fitur terkhususkan yang memuat kesalahan tertentu yang telah diketahui . kesalahan tertentu ini lah yang dapat berpotensi sangat tidak bertanggung jawab dan membuka

peluang untuk munculnya tuntutan hukum yang sangat mahal.

2. Biaya kualitas , kualitas merupakan hal yang penting , tetapi kualitas memerlukan biaya dan uang , terlalu banyak waktu dan uang yang diperlukan untuk mencapai peringkat kualitas perangkat lunak yang diinginkan. Biaya – biaya kualitas mencakup semua biaya yang diperlukan untuk mencapai peringkat kualitas tertentu atau biaya – biaya yang diakibatkan kualitas yang buruk. Biaya – biaya kualitas dapat dibagi menjadi biaya – biaya yang berkaitan dengan pencegahan , penilaian dan kegagalan .
 - a. Biaya pencegahan mencakup biaya aktivitas – aktivitas pengelolaan yang diperlukan untuk merencanakan dan mengorganisasi semua aktivitas yang berkaitan dengan kendali kualitas dan semua aktivitas yang berkaitan dengan jaminan kualitas , biaya penambahan aktivitas teknis untuk mengembangkan model – model kebutuhan serta perancangan yang lengkap, biaya perencanaan pengujian dan biaya semua pelatihan yang berkaitan dengan semua aktivitas tersebut.
 - b. Biaya penilaian mencakup biaya aktivitas – aktivitas untuk mendapatkan pandangan yang berkaitan dengan tujuan menyeluruh masing – masing proses perangkat lunak yang digunakan saat pengembangan mengembangkan perangkat

- lunak yang kelak akan digunakan oleh pera pelanggan.
- c. Biaya kegagalan merupakan biaya yang tidak diperlukan jika tidak ada kesalahan yang ditemukan sebelum atau setelah pengiriman produk ke para pelanggan. Contohnya adalah biaya yang diperlukan untuk melakukan perbaikan – perbaikan untuk menghilangkan kesalahan , biaya yang muncul saat melakukan penyelidikan dari perbaikan – perbaikan kesalahan , biaya yang berkaitan dengan langkah – langkah pengumpulan metrik kualitas yang meungkinan organisasi – organisasi perangkat lunak melakukan penilaian modus kegagalan yang muncul dari perangkat lunak yang sedang dikembangkan.
 - d. Biaya kegagalan eksternal , berhubungan dengan cacat – cacat program yang ditemukan setelah perangkat lunak dikirimkan ke pelanggan. Contohnya adalah biaya yang dikeluarkan karena penyelesaian keluhan – keluhan pelanggan.
3. Kondisi – kondisi yang berkaitan dengan hukum , misalnya terkaitan dengan tuntutan – tuntutan hukum yang dilayangkan oleh pengguna atau pihak terkait akibat dari pengembangan perangkat lunak tersebut.
4. Kualitas keamaman , berkaitan dengan masalah keamanan misalnya aktivitas yang dilakukan

oleh *hacker* dan lain – lain , sehingga mengakibatkan kegagalan perangkat lunak.

5. Imbas tindakan – tindakan manajemen , misalnya keputusan – keputusan terkait dengan perkiraan biaya , penjadwalan dan lain – lain.

7.1.4. Pencapaian Kualitas Perangkat Lunak

Kualitas perangkat lunak datang dari hasil tindakan – tindakan manajemen proyek yang baik dan dari hasil praktik – praktik rekayasa perangkat lunak yang baik. Ada empat aktivitas yang dapat membantu pencapaian kualitas perangkat lunak :

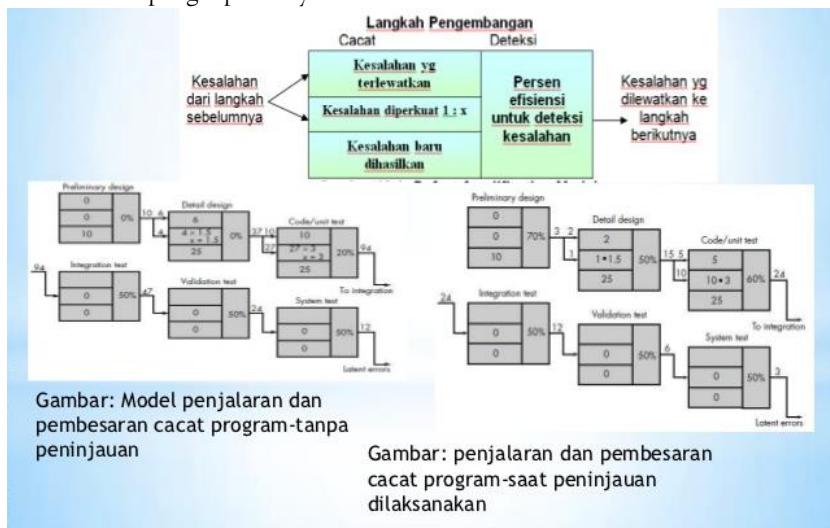
1. Metode – metode rekayasa perangkat lunak
2. Teknik – teknik manajemen proyek
3. Kendali kualitas
4. Jaminan kualitas.

7.2. Teknik – teknik Peninjauan

Tinjauan – tinjauan teknik perangkat lunak merupakan seleksi untuk proses – proses perangkat lunak yaitu tinjauan – tinjauan yang seharusnya diterapkan pada berbagai titik pada perjalanan proses rekayasa perangkat lunak dan juga sebagai penyikapan *error* dan *defect* yang selanjutnya dapat dihilangkan. Beberapa teknik yang dapat dilakukan untuk melakukan peninjauan perangkat lunak adalah sebagai berikut :

- I. Terminologi kesalahan , yaitu dengan menyatakan permasalahan kualitas perangkat lunak sebelum perangkat lunak diluncurkan . Ini dapat di evaluasi oleh rekayawan perangkat lunak atau orang lain dengan menguji perangkat lunak secara terbatas.

2. Teknik penjalaran dan pembesaran cacat program dan penghapusannya , digunakan untuk menggambarkan pembentukan dan pendekripsi kesalahan selama perancangan dan pembentukan dan dapat digunakan untuk menggambarkan pendekripsi kesalahan – kesalahan selama tindakan – tindakan kode dari suatu proses perangkat lunak di laksanakan . berikut adalah contoh model penjalaran dan pemebsaran cacat program dan penghapusannya :



Gambar 7.2 Model Penjalaran dan Pembesaran Cacat Program dan Penghapusannya
Sumber : (Roger S. Pressman, 2012)

3. Teknik informal , dilakukan dengan menyusun pertanyaan – pertanyaan kualitas untuk meningkatkan kualitas perangkat lunak.

4. Teknik formal , dilakukan dengan melakukan dengan membuat pertemuan – pertemuan peninjauan oleh pengambang dan orang lain untuk menguji perangkat lunak, kemudian hasil cacatan menjadi hasil peninjauan.

7.3. Jaminan Kualitas Perangkat Lunak

Jaminan kualitas perangkat lunak atau *Software Quality Assurance* (SQA), mencakup :

1. Suatu proses jaminan kualitas perangkat lunak
2. Pekerjaan – pekerjaan jaminan kualitas dan kendali kualitas yang bersifat spesifik
3. Praktik – praktik rekayasa perangkat lunak (metode – metode dan perkakas – perkakas) yang efektif
4. Kendali terhadap semua produk kerja perangkat lunak dan perubahan – perubahan yang dibuat atasnya.
5. Prosedur – prosedur untuk memastikan kesesuaian dengan standar – standar pengembangan perangkat lunak yang telah ada sebelumnya
6. Mekanisme – mekanisme yang berkaitan dengan pengukuran dan pelaporan.

Orang yang melaksanakan SQA melihat perangkat lunak dari sudut pandang pelanggan. Berikut adaah elemen – elemen jaminan kualitas perangkat lunak :

1. Standar , Pekerjaan SQA memastikan bahwa standar – standar kualitas seperti IEEE, ISO , dan lain – lain telah diadopsi dan diikuti dan juga memastikan bahwa semua produk kerja yang dihasilkan sesuai dengan standar – standar kualitas tersebut.
2. Peninjauan dan audit, dilakukan untuk menyikap kesalahan – kesalahan .

3. Pengujian, Pekerjaan SQA memastikan bahwa pengujian – pengujian telah dengan baik direncanakan dan telah secara efisien dilaksanakan sehingga mampu mencapai sasaran utamanya.
4. Pengumpulan kesalahan atau cacat program dan analisis, SQA mengumpulkan dan menganalisis data kesalahan – kesalahan dan data cacat program kemudian menentukan aktivitas rekayasa perangkat lunak yang sesuai untuk memperbaikinya.
5. Manajemen perubahan , SQA memastikan bahwa praktik – praktik pengelolaan perubahan – perubahan telah dilaksanakan dengan semestinya.
6. Pendidikan , SQA mendukung program – program pendidikan yang berkaitan dengan penerapan jaminan kualitas.
7. Pengelolaan vendor , SQA memastikan bahwa perangkat lunak yang berkualitas tinggi dihasilkan oleh praktik – praktik jaminan kualitas spesifik yang disarankan agar diikuti oleh vendor dan melibatkan kualitas sebagai bagian dari setiap kontrak dengan para vendor eksternal tersebut.
8. Pengelolaan keamanan, SQA memastikan agar proses – proses serta teknologi yang digunakan mencapai sasaran keamanan perangkat lunak.
9. Keamanan , SQA bertanggung jawab untuk melakukan penilaian kegagalan – kegagalan perangkat lunak dan bertanggung jawab untuk mnegawali langkah – langkah yang diperlukan untuk mengurangi risiko – risiko yang terjadi.
10. Manajemen risiko, SQA bertugas untuk memastikan bahwa aktivitas – aktivitas yang berkaitan dengan manajemen risiko telah secara baik dilaksanakan dan

perencanaan – perencanaan yang berkaitan dengan risiko telah ditetapkan dengan semestinya.

Berikut adalah pekerjaan – pekerjaan SQA :

1. Mempersiapkan suatu rencana SQA untuk proyek
2. Berpartisipasi dalam pengembangan deskripsi proses perangkat lunak yang dilaksanakan proyek.
3. Meninjau aktivitas – aktivitas rekayasa perangkat lunak untuk memverifikasi kesesuaian dengan proses perangkat lunak yang telah didefinisikan sebelumnya.
4. Melaksanakan audit terhadap produk – produk kerja perangkat lunak yang telah dirancang sebelumnya untuk memverifikasi kesesuaianya dengan proses perangkat lunak yang telah didefinisikan sebelumnya.
5. Memastikan bahwa penyimpangan pada pekerjaan perangkat lunak dan produk – produk kerja telah terdokumentasi dengan baik dan telah ditangani dengan cara yang sesuai dengan prosedur – prosedur yang telah terdokumentasi sebelumnya.
6. Mencatat setiap ketidaksesuaian dan melaporkannya kepada manajemen puncak.

Sasaran – sasaran dari SQA adalah :

- I. Kualitas kebutuhan
Kebenaran , kelengkapan dan konsistensi dari model – model kebutuhan akan memiliki imbas yang sangat kuat pada kualitas semua produk kerja yang mengikutinya. SQA harus mampu memastikan bahwa tim perangkat lunak telah secara baik meninjau model – model kebutuhan untuk mencapai peringkat kualitas yang tinggi.

2. Kualitas perancangan

Setiap elemen model perancangan semestinya dinilai oleh tim perangkat lunak untuk memastikan bahwa model perancangan sudah memperhatikan kualitas yang tinggi dan perancangannya sudah sesuai dengan spesifikasi – spesifikasi kebutuhan yang telah ditentukan sebelumnya. SQA menggunakan atribut – atribut perancangan sebagai indikator kualitas.

3. Kualitas kode

Kode – kode dalam bahasa pemograman dan produk – produk kerja terkait harus sesuai standar pengkodean lokal dan memperlihatkan karakteristik – karakteristik yang akan memfasilitasi kemudahan pemeliharaannya. SQA mengisolasi masing – masing atribut yang memungkinkan dilakukannya analisis – analisis terhadap kualitas kode dengan cara yang baik.

4. Efektivitas kendali kualitas

Tim perangkat lunak menerapkan sumber daya yang terbatas untuk dapat mencapai kualitas yang tinggi. SQA menganalisisi alokasi sumber daya untuk melakukan peninjauan dan melakukan pengujian – pengujian untuk menilai apakah sumber daya tersebut dialokasikan dengan cara efektif.

Six Sigma Rekayasa Perangkat lunak.

Metodologi Six Sigma adalah metodologi yang digunakan untuk melaksanakan jaminan kualitas secara statistik. Berikut adalah langkah Six Sigma Perangkat lunak :

1. Mendefinisikan kebutuhan – kebutuhan pelanggan dan dadaran proyek perangkat lunak menggunakan metode – metode komunikasi dengan para pelanggan yang terdefinisi dengan baik.
2. Mengukur proses – proses perangkat lunak yang ada serta outputnya untuk menentukan kinerja kualitas saat ini
3. Menganalisis metrik – metrik cacat program dan menentukan penyebabnya .
4. Memperbaiki proses perangkat lunak dengan menghilangkan akar masalah yang berkaitan dengan cacat program
5. Mengendalikan proses perangkat lunak untuk bisa memastikan bahwa pekerjaan yang dimasa akan datang tidak terpengaruh oleh cacat yang dijumpai saat ini.

Langkah diatas dapat disingkat dengan metode DMAIC (*define, measure, Analyze, improve , control*).

Salah satu elemen yang penting dalam kualitas perangkat lunak adalah keandalan perangkat lunak . keandalan ini dapat definisikan secara statistik yaitu tingkat kemungkinan operasi yang bebas kegagalan dari suatu perangkat lunak pada lingkungan operasional tertentu dalam waktu tertentu yang telah ditentukan sebelumnya.

Selain keandalan elemen lain yang penting dalam kualitas perangkat lunak adalah keamanan perangkat lunak , yang merupakan suatu aktivitas perangkat lunak yang pada dasarnya berfokus pada identifikasi dan penilaian tentang hal – hal buruk yang mungkin memiliki imbas negatif tertentu pada

perangkat lunak dan mungkin suatu waktu menyebabkan kegagalan perangkat lunak.

Standar ISO 9001 : 2000

- I. Menetapkan elemen – elemen sistem pengelolaan kualitas
 - a. Mengembangkan , mengimplementasikan dan memperbaiki sistem
 - b. Menyediakan kebijakan yang menekankan pentingnya sistem
2. Mendokumentasikan sistem kualitas
 - a. Mendeskripsikan proses
 - b. Menghasilkan sebuah manual operasional
 - c. Mengembangkan metode – metode untuk mengendalikan dokumen – dokuemen.
 - d. Menetapkan metode untuk memelihara catatan - catatan
3. Mendukung kendali dan jaminan kualitas
 - a. Mempromosikan pentingnya kualitas di antara semua stakeholder
 - b. Berfokus pada kepuasan pelanggan
 - c. Mendefinisikan sebuah rencana kualitas yang menyelesaikan sasaran – sasaran , tanggung jawab dan otoritas
 - d. Mendefinisikan mekanisme – mekanisme komunikasi di antara para stakeholder
4. Menetapkan mekanisme – mekanisma peninjauan sistem pengelolaan kualitas
 - a. Mengidentifikasi metode – metode tinjauan dan mekanisme umpan balik

- b. Mendefinisikan prosedur – prosedur tindak lanjut
5. Mengidentifikasi sumber – sumber kualitas, termasuk di dalamnya personel – personel , pelatihan – pelatihan dan elemen – elemen infrastruktur.
 - a. Menetapkan mekanisme – mekanisme kendali untuk perencanaan dan kebutuhan – kebutuhan pelanggan, untuk aktivitas – aktivitas teknis , untuk pantauan dan manajemen proyek.
6. Mendefinisikan metode – metode untuk remidasি.
 - a. Menlai kualitas dan metrik – metrik data
 - b. Mendefiniskan pendekatan untuk perbaikan proses dan kualitas berkelanjutan.

Perencanaan SQA

Standar perencanaan SQA dapat dilihat di IEEE, dimana dalam standar tersebut mengidentifikasi :

1. Tujuan dan lingkup perencanaan kualitas perangkat lunak
2. Deskripsi dari semua produk kerja rekayasa perangkat lunak (model – model, dokumen – dokumen, kode – kode dalam bahasa pemrograman tertentu) yang tergolong dalam kontek SQA
3. Semua standar serta praktik perangkat lunak yang dapat diterapkan selama proses perangkat lunak
4. Aksi – aksi dan pekerjaan – pekerjaan SQA (tinjauan – tinjauan , audit -audit) dan penempatannay di seluruh proses perangkat lunak
5. Perkakas – perkakas serta metode – metode yang mendukung aksi – aksi dan tindakan – tindakan perangkat lunak

6. Prosedur – prosedur manajemen konfigurasi perangkat lunak
7. Metode – metode untuk merakit , mengawasi keamanannya, dan memilihara catatan – catatan yang terkait dengan SQA
8. Peran – peran organisasional dan tanggung jawab yang terkait dengan kualitas perangkat lunak yang sedang dikembangkan.

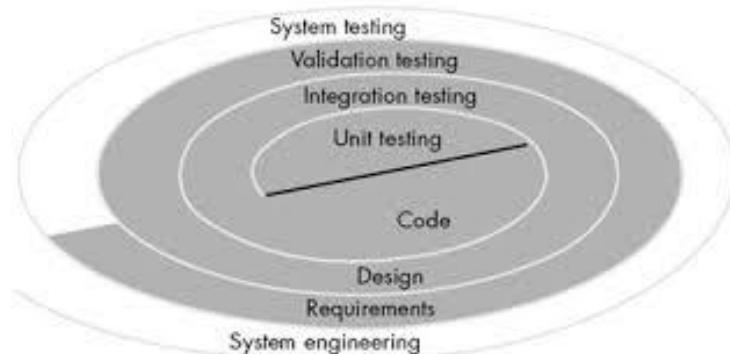
7.4. Strategi Pengujian Perangkat Lunak

Dalam pengujian perangkat lunak membutuhkan suatu strategi , dengan strategi tersebut maka pengujian perangkat lunak dapat menjadi lebih efektif dan efisien. Strategi pengujian perangkat lunak memberikan panduan langkah – langkah yang harus dilakukan , waktu pelaksanaan serta sumber daya yang diperlukan untuk pengujian tersebut.

7.4.I. Pendekatan strategi pada pengujian perangkat lunak

Pengujian perangkat lunak dapat dilakukan dengan verifikasi dan validitas. Verifikasi merujuk pada sekumpulan tugas yang memastikan bahwa perangkat lunak benar menerapkan fungsi yang ditentukan. Sedangkan Validitas merujuk ke sekumpulan tugas yang berbeda yang memastikan bahwa perangkat lunak yang telah dibangun dapat dilacak berdasarkan prasyarat pelanggan. Kegiatan verifikasi dan validasi meliputi tinjauan teknis, audit konfigurasi dan kualitas, monitoring kinerja, simulasi, studi kelayakan, kajian dokumentasi, tinjauan basis data, analisis algoritma, pengujian pengembangan , pengujian kegunaan,

pengujian kualifikasi, uji penerimaan , dan uji installasi. Pengujian perangkat lunak biasanya dilakukan oleh pengembang melalui kelompok penguji independen, karena pengujian dengan kelompok ini dapat menghilangkan konflik . kelompok ini merupakan bagian dari tim proyek pengembangan perangkat lunak. berikut adalah strategi pengujian perangkat lunak :

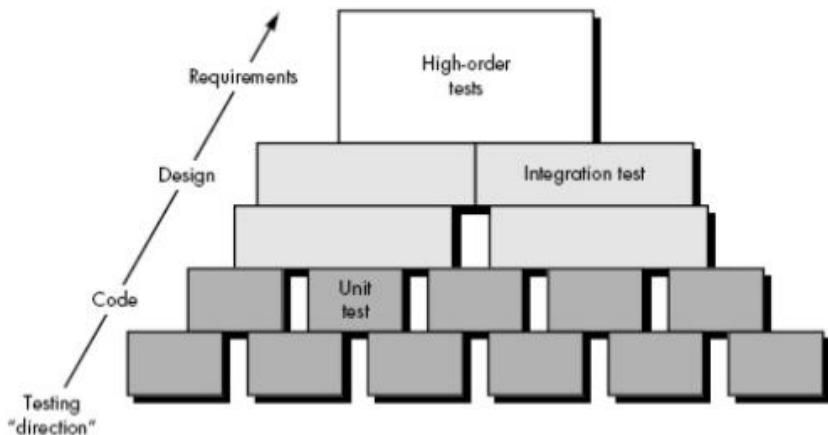


Gambar 7.3 Strategi pengujian perangkat lunak

Sumber : (Roger S. Pressman, 2012)

1. Pengujian unit , pengujian masing – masing unit komponen , kelas, atau objek isi dari perangkat lunak yang diimplementasikan.
2. Pengujian integrasi , berfokus pada perancangan dan pembangunan arsitektur perangkat lunak.
3. Pengujian validasi, yaitu proses validasi dari prasyarat perangkat lunak yang telah ditetapkan sebelumnya.
4. Pengujian sistem , yaitu pengujian seluruh elemen sistem .

Berikut adalah langkah – langkah pengujian perangkat lunak :



Gambar 7.4 Langkah – langkah pengujian perangkat lunak

Sumber : (Roger S. Pressman, 2012)

1. Pengujian pertama fokus pada komponen secara individual , kemudian komponen ini diintegrasikan untuk membentuk paket perangkat lunak.
2. Pengujian integrasi dilakukan untuk verifikasi dan pembangunan program, teknik perancangan kasus pengujinya berfokus pada input dan output. Pengujian ini berfungsi untuk memastikan keseluruhan jalur kontrol utama.
3. Pengujian selanjutnya adalah pengujian level atas dilakukan dengan pengujian validasi dan pengujian sistem . Pengujian validasi berfungsi untuk memberikan jaminan bahwa perangkat

lunak telah memenuhi prasyarat informasi, fungsional, perilaku dan prasyarat kinerja. Sedangkan pengujian sistem memverifikasi bahwa semua elemen saling bertautan dengan benar dan keseluruhan fungsi sistem dapat dicapai.

Yang pasti , dalam pengujian perangkat lunak akan berhasil jika :

1. Penguji perangkat lunak menentukan persyaratan produk kuantitas jauh sebelum pengujian dimulai.
2. Penguji perangkat lunak menyatakan tujuan pengujian secara eksplisit
3. Penguji perangkat lunak memahami pengguna perangkat lunak dan mengembangkan sebuah profil untuk setiap kategori pengguna.
4. Penguji perangkat lunak mengembangkan rencana pengujian yang menekankan pengujian siklus cepat
5. Penguji perangkat lunak membangun perangkat lunak yang tanggih yang dirancang untuk dapat menguji dirinya sendiri.
6. Penguji perangkat lunak menggunakan kajian teknis efektif sebagai filter sebelum pengujian
7. Penguji perangkat lunak melakukan kajian teknis untuk menilai strategi pengujian dan kasus pengujian itu sendiri.
8. Penguji perangkat lunak mengembangkan pendekatan perbaikan terus menerus untuk proses pengujian.

7.4.2. Strategi Pengujian untuk perangkat lunak konvensional

Pada strategi pengujian perangkat lunak konvensional ini terdapat beberapa tahapan yaitu :

I. Pengujian Unit

Fokus dari pengujian unit adalah usaha untuk memverifikasi setiap unit dari perancangan perangkat lunak. Pengujian unit diawali dengan menguji aliran data pada komponen antarmuka , jika data tidak masuk dan keluar dengan benar maka pengujian yang lain dapat dipertanyakan. Pengujian unit ini biasanya merupakan tahapan coding

2. Pengujian Integrasi

Pengujian integrasi merupakan teknik sistemik untuk membangun arsitektur perangkat lunak, sementara pada saat yang sama melakukan pengujian untuk menemukan kesalahan – kesalahan yang terkait dengan antarmuka. Tujuan dari pengujian ini adalah untuk mengambil komponen yang diuji dan membangun struktur program yang telah ditentukan oleh perancangan. Berikut adalah beberapa teknik pengujian integrasi :

- a. Integrasi atas – ke – bawah , pendekatan incremental untuk membangun arsitektur perangkat lunak
- b. Integrasi bawah – ke – atas , dilakukan dengan memulai konstruksi dan pengujian dengan modul atomik yaitu

- komponen – komponen di tingkat paling rendah dalam struktur program.
- c. Pengujian regresi, pengeksikusian kembali beberapa rangkaian pengujian yang telah dijalankan untuk memastikan bahwa perubahan tidak menimbulkan efek samping yang diharapkan.
 - d. Pengujian asap, pengujian yang memungkinkan tim perangkat lunak untuk menilai proyek sesering mungkin . Manfaat dari pengujian asap ini adalah risiko integrasi diminimalkan, kualitas produk akhir ditingkatkan, diagnosis dan koreksi kesalahan disederhanakan, kemajuan lebih mudah untuk dinilai.

7.4.3. Strategi Pengujian untuk perangkat lunak berorientasi objek

Pengujian pada perangkat lunak berbasis objek berbeda dengan konvensional karena konsep unit berubah menjadi *class* dan objek. Pada pengujian integrasi untuk perangkat lunak berorientasi objek terdapat beberapa pendekatan yaitu pengujian berbasis *thread* yang mengintegrasikan sekumpulan kelas yang dibutuhkan untuk merespon satu masukan ke dalam sistem . Pendekatan kedua menggunakan pengujian berbasis kegunaan yaitu memulai membangun sistem dengan menguji kelas – kelas secara mandiri . Pendekatan yang ketiga adalah pengujian cluster yaitu pengcusteran kelas – kelas yang berkolaborasi dijalankan dengan merancang kasus pengujian yang berusaha untuk

menemukan kesalahan dalam kolaborasi tersebut. Salah satu perangkat lunak berbasis objek adalah aplikasi web. Berikut adalah contoh strategi pendekatan untuk menguji aplikasi web :

1. Model isi untuk aplikasi web ditinjau untuk menemukan kesalahan
2. Model antarmuka ditinjau untuk memastikan bahwa semua kasus yang digunakan dapat diakomodasikan
3. Model perancangan untuk aplikasi web ditinjau untuk menemukan kesalahan navigasi
4. Antarmuka pengguna diuji untuk menemukan kesalahan dalam presentasi dan mekanik navigasi
5. Setiap komponen fungsional diterapkan pengujian unit
6. Navigasi seluruh arsitektur diuji
7. Aplikasi web diimplementasikan dalam beberapa konfigurasi lingkungan yang berbeda dan diuji kompatibilitasnya dengan setiap konfigurasi.
8. Uji keamanan dilakukan dalam upaya mengeksloitasi kelemahan – kelemahan dalam aplikasi web atau dalam lingkungannya.
9. Kinerja pengujian dikontrol.
10. Aplikasi web diuji oleh populasi yang dikendalikan dan dipantau oleh pengguna akhir . Hasil interaksi mereka dengan sistem dievaluasi , yakni dalam hal kesalahan isi dan navigasi, kegunaan, kompatibilitas, dan keandalan serta kinerja aplikasi web.

7.4.4. Pengujian Validasi

Pengujian validasi pada pengujian perangkat lunak berbasis objek adalah pengujian berfokus pada tindakan pengguna yang terlihat dan output dari sistem yang dikenali pengguna. Validasi perangkat lunak berbasis objek dicapai melalui serangkaian pengujian yang memperlihatkan kesesuaian dengan persyaratan. Rencana pengujian menguraikan kelas – kelas pengujian, prosedur pengujian , karakteristik perilaku tercapai, semua isi disajikan dengan benar dan akurat, semua prasyarat kinerja tercapai, pendokumentasian benar, dan kegunaan prasyarat lainnya dipenuhi. Elemen penting dari proses validasi adalah peninjauan konfigurasi yang bertujuan untuk memastikan bahwa semua elemen dari konfigurasi perangkat lunak telah secara benar dikembangkan, dikatalogkan, dan memiliki rincian yang diperlukan untuk mendukung aktivitas pendukung. Dalam pengujian validitas terdapat pengujian alpha dan pengujian beta. Pengujian alpa dilakukan disisi pengembang oleh sekelompok perwakilan dari pengguna akhir . Sedangkan pengujian beta dilakukan oleh pengguna akhir tanpa pengembang hadir.

7.4.5. Pengujian Sistem

Pengujian sistem adalah serangkaian pengujian yang berbeda – beda yang tujuan utamanya adalah untuk mewujudkan sistem berbasis komputer. Berikut adalah pengujian – pengujian yang terdapat dalam pengujian sistem :

1. Pengujian pemulihan
Merupakan pengujian sistem yang memaksa perangkat lunak untuk gagal dalam berbagai cara dan memverifikasi bahwa pemulihan dilakukan dengan benar.
2. Pengujian Keamanan
Merupakan pengujian yang mencoba untuk memverifikasi mekanisme perlindungan yang dibangun ke dalam sistem , yang pada kenyataanya , melindungi dari penetrasi yang tidak benar.
3. Pengujian stress
Merupakan pengujian yang menjalankan sistem dengan cara meminta sumber daya dalam jumlah , frekuensi atau volume yang abnormal.
4. Pengujian kinerja
Merupakan pengujian yang dirancang untuk menguji kinerja *run-time* dari perangkat lunak dalam konteks sistem yang terintegrasi.
5. Pengujian Deployment
Merupakan pengujian yang memeriksa semua prosedur instalasi dan perangkat lunak instalsi khusus yang akan digunakan oleh pelanggan dan semua dokumentasi yang akan digunakan untuk memperkenalkan perangkat lunak ke pengguna akhir.

7.4.6. Pelacakan Kesalahan

Untuk melakukan pelacakan kesalahan maka kita perlu memahami karakteristik kesalahan yang memberikan petunjuk :

1. Gejala dan penyebabnya mungkin secara geografis jauh , maksudnya gejala dapat muncul disalah satu bagian dari sebuah program , sementara penyebabnya terletak pada tempat yang jauh.
2. Gejala mungkin hilang saat kesalahan lain dikoreksi
3. Gejala ini sebenarnya oleh *nonerror* atau bisa disebut ketidakakuratan.
4. Gejala dapat disebabkan oleh kesalahan manusia yang tidak mudah dilacak.
5. Gejala mungkin akibat dari masalah waktu daripada akibat masalah pemrosesan.
6. Mungkin sulit untuk secara akurat mereproduksi kondisi masukan
7. Gejala dapat berselang , ini sangat umum dalam *embedded system* diaman perangkat keras dan perangkat lunak terkait erat.
8. Gejala mungkin dikarenakan penyebab yang didistribusikan ke sejumlah tugas berjalan pada prosesor yang berbeda.

Strategi – strategi yang dapat digunakan untuk pelacakan kesalahan adalah sebagai berikut :

1. *Brute force*
2. *backtracking*

3. *couse elimination*

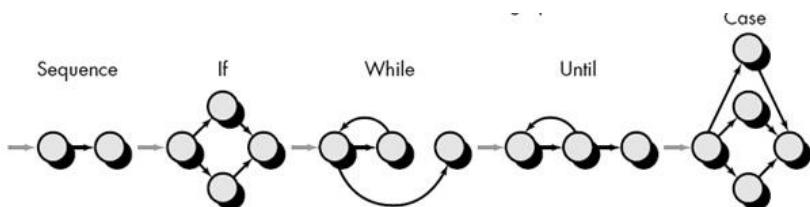
7.5. Pengujian Aplikasi Konvensional

Dasar dari pengujian perangkat lunak adalah bahwa perangkat lunak harus memiliki karakteristik – karakteristik kemampuan untuk dapat diperasikan, kemampuan untuk bisa diobservasi, kemampuan untuk dapat dikontrol, kemampuan untuk dapat disusun, kesederhanaan , stabilitas dan kemampuan untuk dapat dipahami. Sedangkan karakteristik untuk pengujian yang baik adalah memiliki profitabilitas yang tinggi untuk menemukan kesalahan , pengujian yang baik tidak berulang – ulang, pengujian yang baik harus menjadi bibit terbaik, pengujian yang baik harus tidak terlalu sederhana atau terlalu rumit. Dalam pengujian aplikasi konvensional terdapat dua pandangan yaitu pandangan internal dan pandangan external. Pendekatan pengujian dari pandangan eksternal disebut *black box testing* sedangkan pendekatan pengujian dari pandangan internal disebut *white box testing*. Pengujian *black box testing* berkaitan dengan pengujian – pengujian yang dilakukan pada antarmuka perangkat lunak. Pengujian *white box* didasarkan pada pemeriksaan yang teliti terhadap detail prosedural.

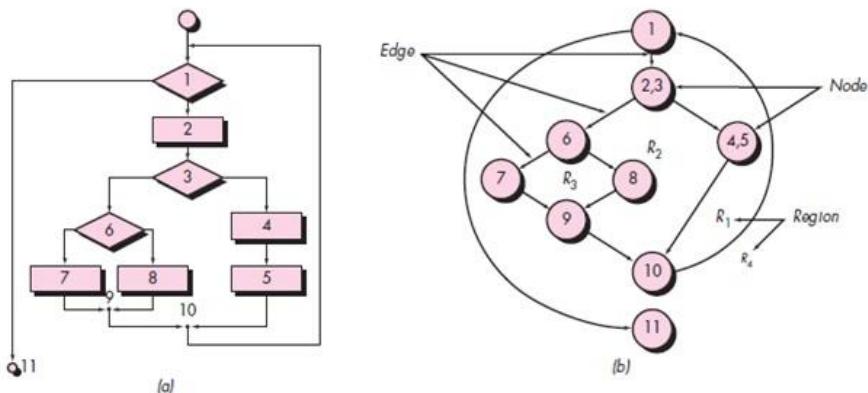
7.5.I. *White box testing*

Hasil dari *white box testing* adalah test case yang menjamin bahwa semua jalur independen di dalam modul telah dieksekusi sedikitnya satu kali , melaksanakan semua keputusan logis pada sisi benar dan yang salah , melaksanakan semua loop pada batas mereka dan dalam batas – batas operasional , melakukan struktur data internal untuk memastikan kesahihannya. Teknik pengujian *white box testing*

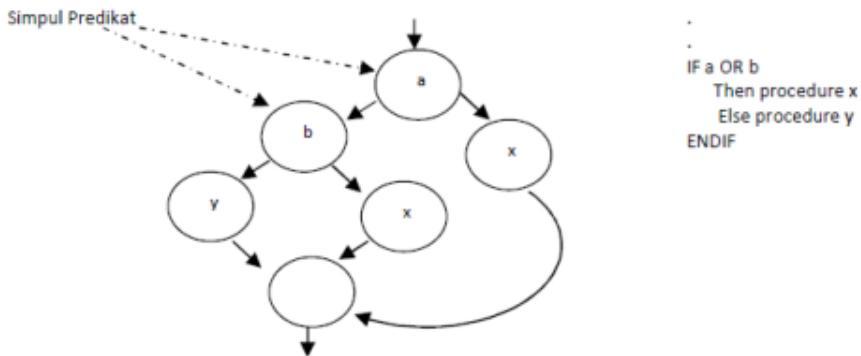
menggunakan pengujian jalur dasar (*basis path testing*). Pada teknik ini test case diturunkan untuk menguji rangkaian dasar yang dijamin untuk mengeksekusi setiap pernyataan dalam program sebanyaknya satu kali pengujian. Pengujian jalur dasar dapat menggunakan grafik alir sebagai berikut :



Gambar 7.5 Notasi grafik alir
Sumber : (Roger S. Pressman, 2012)



Gambar 7.6 (a) Digramp alir dan (b) grafik alir
Sumber : (Roger S. Pressman, 2012)



Gambar 7.7 Logika gabungan

Sumber : (Roger S. Pressman, 2012)

Dari diagram alir atau grafik alir kita dapat menterjemahkan kedalam sebuah logika , seperti terlihat pada gambar 7.7 yang memperlihatkan jika langkah dimulai dari a atau b maka akan memiliki prosedure yang berbeda. Dengan melihat pada grafik alir tersebut kita juga dapat menyusun jalur program independent yaitu setiap jalur yang melalui program yang memperkenalkan setidaknya satu kumpulan pernyataan – pernyataan pemrosesan atau kondisi baru. Seperti terlihat pada gambar 7.6 (b) kita dapat membuat satu set jalur independent dengan urututan sebagai berikut :

- | | | |
|--------|---|---------------------|
| Path 1 | : | I-II |
| Path 2 | : | I-2-3-4-5-10-I-II |
| Path 3 | : | I-2-3-6-8-9-10-I-II |
| Path 4 | : | I-2-3-6-7-9-10-I-II |

Banyaknya path atau jalur pengujian yang dapat diambil disebut dengan *kompleksitas siklomatik* yang dapat dicari dengan menggunakan rumus :

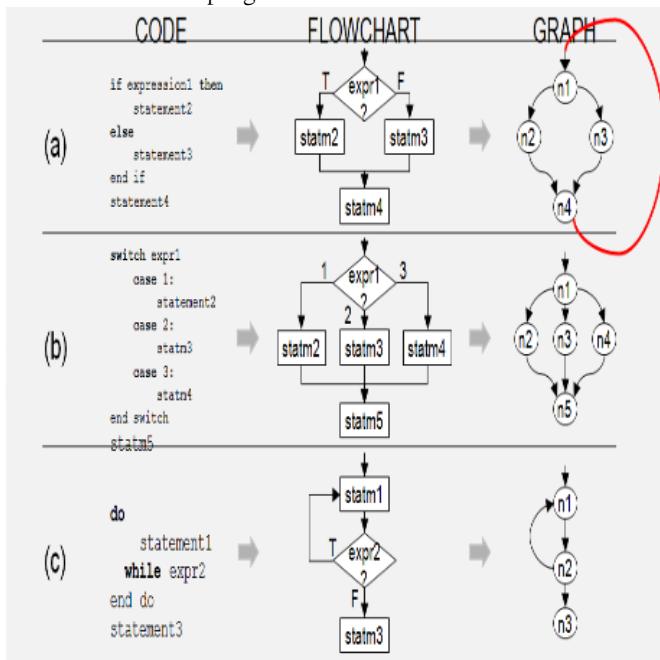
$$V(G) = E - N + 2$$

$V(G)$ = Kopleksitas siklomatik

E = Jumlah edge grafik alir

N = Jumlah node grafik alir

Metode pengujian jalur dasar ini dapat diterapkan untuk perancangan prosedural atau kode program.



Gambar 7.7 Contoh pengujian jalur dasar.

Sumber : (Roger S. Pressman, 2012)

7.5.2. Black box testing

Black box testing merupakan strategi pengujian yang memperhatikan atau memfokuskan kepada faktor fungsionalitas dan spesifikasi perangkat lunak. pada pengujian ini tidak membutuhkan pengetahuan mengenai alur internal , struktur atau implementasi dari software *under test*. Kategori – kategori kesalahan yang diuji oleh *black box testing* adalah fungsi – fungsi yang salah atau hilang, kesalahan interface, kesalahan dalam struktur data atau akses database eksterna , kesalahan perfoma, kesalahan inisialisasi dan terminasi. Berikut adalah langkah – langkah dari proses *black box testing* :

1. Menganalisis kebutuhan spesifikasi dari perangkat lunak
2. Pemilihan jenis input yang memungkinkan menghasilkan output bermar serta jenis input yang memungkinkan output salah pada perangkat lunak yang sedang diuji.
3. Menentukan output untuk suatu jenis input.
4. Pengujian dilakukan dengan input – input yang telah benar – benar diseleksi
5. Pembandingan output yang dihasilkan dengan output yang diharapkan
6. Menentukan fungsionalitas yang seharusnya pada perangkat lunak yang sedang diuji.

Pengujian *black box testing* dapat dilakukan pada setiap level pembangunan sistem yaitu mulai dari *unit*,

integration, system dan *acceptance*. Teknik – teknik dalam *black box testing* adalah sebagai berikut :

1. *Equivalence partitioning*

Teknik ini merupakan teknik pengujian software yang melibatkan pembagian nilai input ke dalam bagian nilai valid dan tidak valid dan memilih perwakilan dari masing – masing data test.

2. *Boundary value analysis*

Teknik ini merupakan teknik pengujian software yang melibatkan penentuan – penentuan nilai input dan memilih beberapa nilai dari batasan tersebut baik luar maupun dalam batasan – batasan tersebut sebagai data test.

3. *Cause Effect Graphic*

Teknik ini merupakan teknik pengujian software yang melibatkan pengidentifikasi sebab – sebab (kondisi input) dan akibat – akibat (kondisi output) menghasilkan kasus – kasus test.

7.6. Pengujian Aplikasi – aplikasi Berorientasi Objek

Untuk menguji aplikasi berorientasi objek maka ada tiga hal yang harus dilakukan yaitu definisi pengujian harus diperluas mencakup teknik penemuan kesalahan yang diterapkan untuk model analisis dan perancangan berorientasi objek , strategi untuk pengujian unit dan integrasi harus berubah secara signifikan , perancangan test case harus menjelaskan karakteristik unik dari perengkat lunak berorientasi objek.

Cara memperluas pandangan tentang pengujian pada perangkat lunak berbasis objek adalah dengan memahami bahwa pembangunan perangkat lunak berorientasi objek dimulai dengan membuat model – model kebutuhan dan peranangan maka pada setiap model – model dapat diuji dalam rangka menemukan kesalahan sebelum menyebar pada iterasi berikutnya. Semua model berorientasi objek harus diuji dalam hal kebenaran , kelengkapan, dan konsistensi dalam konteks model sintaks , semantik dan pragmatik.

1. Kebenaran Model analisis dan perancangan dikaitkan dengan notasi dan sintak yang mewakili model analisis dan perancangan. Kebenaran sintak dinilai pada pemakaian yang tepat dari simbol tersbeut, masing – masing model ditinjau untuk memastikan bahwa konvensi pemodelan yang tepat telah di pertahankan.
2. Konsistensi model berorientasi objek dapat dinilai dengan mempertimbangkan hubungan antar entitas dalam model. Untuk menilai konsistensi , maka perlu diperiksa setiap kelas dan konesinya. Untuk mengevaluasi model kelas terdapat langkah – langkah berikut :
 - a. Lihat kembali model CRC dan model hubungan objek, periksa silang untuk memastikan bahwa semua kolaborasi yang ditunjukkan oleh model kebutuhan telah direfleksikan secara tepat pada keduanya.
 - b. Periksa deskripsi dari setiap kartu indeks CRC untuk menentukan apakah tanggung jawab yang didelegasikan adalah bagian dari definisi kolaborator tersebut.

- c. Inversikan koneksi untuk memastikan bahwa setiap kolaborator yang diminta untuk layanan ini menerima permintaan dari sumber yang masuk akal.
- d. Dengan menggunakan koneksi yang diinversi yang diperiksa pada langkah C, tentukan apakah kelas – kelas lain mungkin dibutuhkan atau apakah tanggung jawab dikelompokkan diantara kelas – kelas secara tepat.
- e. Tentukan apakah tanggung jawab yang diminta secara luas dapat digabungkan menjadi tanggung jawab tunggal.

Pengujian dalam konteks perangkat lunak berorientasi objek sedikit berbeda dengan perangkat lunak konvensional . Berikut adalah perbedaannya :

- 1. Pengujian unit dalam konteks perangkat lunak berbasis objek , pada pengujian unit perangkat lunak berbasis objek melakukan pengujian kelas , yang dikendalikan oleh operasi – operasi yang dibungkus dalam kelas dan perilaku keadaan kelas tersebut.
- 2. Pengujian integrasi dalam perangkat lunak berorientasi objek , terdapat dua strategi yaitu pengujian berbasis thread yaitu yang mengintegrasikan satu set kelas yang dibutuhkan untuk merespon ke satu masukan atau peristiwa untuk sistem . Setiap thread diintegrasikan dna diuji secara individual. Pengujian yang kedua yaitu pengujian berbasis penggunaan yaitu memulai pembangunan sistem dengan menguji kelas – kelas tersebut. Pengujian lainnya adalah pengujian cluster yang dilakukan dengan memeriksa CRC dan model

hubungan objek yang dilakukan dengan merancang usecase yang berupaya untuk menemukan kesalahan – kesalahan dalam kolaborasi.

3. Pengujian validasi dalam konteks perangkat lunak berorientasi objek, berfokus pada aksi – aksi yang terlihat oleh pengguna dan keluaran – keluaran yang dikenali pengguna sistem tersebut. Untuk membantu pengujian validasi, pengujii harus menggunakan usecase yang merupakan bagian dari model kebutuhan . Usecase menyediakan skenario yang kemungkinan besar menemukan kesalahan dalam kebutuhan interaksi.

Metode – metode yang dapat digunakan untuk pengujian perangkat lunak berbasis obyek diantaranya adalah menggunakan perancangan test case , pengujian berbasis kesalahan , pengujian berbasis skenario , pengujian struktur pemukaan dan struktur dalam.

7.7. Pengujian Aplikasi – aplikasi Web

Untuk memahami tujuan pengujian dalam konteks aplikasi web , maka kita perlu memahami dimensi kualitas untuk aplikasi web diantaranya :

1. Isi dievaluasi baik tingkat sintakis maupun semantis , pada tingkat sintakis dokumen berbasis teks diuji dalam hal tata baca , bahasa . sedangkan pada tingkat semantis aspek yang dinilai adalah kebenaran informasi, konsistensi dan ambiguitas.
2. Fungsi diuji untuk menemukan kesalahan – kesalahan yang menunjukkan ketidak sesuaian dengan prasyarat pelanggan.

3. Struktur dinilai untuk memastikan bahwa aplikasi web tersebut benar – benar menyediakan isi dan fungsi aplikasi web, bahwa struktur dapat diperluas dan dapat didukung saat isi atau fungsionalitas baru ditambahkan.
4. Kegunaan diuji untuk memastikan bahwa setiap kategori pengguna didukung oleh antar muka dan dapat belajar dan menerapkan semua sintaks dan semantik navigasi yang diperlukan
5. Kemampuan untuk dapat dinavigasi diuji untuk memastikan bahwa semua sintaks dan semantik navigasi dilakukan untuk menemukan kesalahan navigasi apapun .
6. Kinerja diuji dibawah berbagai kondisi operasi, konfigurasi, dan pemuatan untuk memastikan bahwa sistem ini responsif terhadap interaksi pengguna dan dapat menangani beban ekstrem tanpa menurunkan operasional yang tidak dapat diterima.
7. Kompatibilitas diuji dengan menjalankan aplikasi web dalam berbagai konfigurasi host yang berbeda baik pada sisi klien maupun server . tujuannya adalah untuk menemukan kesalahan yang khusus pada konfigurasi host yang unik.
8. Interoperabilitas diuji untuk memastikan bahwa aplikasi web berantarmuka dengan benar dengan aplikasi lain san basis data.
9. Keamanan diuji dengan menilai kerentanan potensial dan berusaha menyingkap masing – masing kerentanan . setiap usaha penetrasi yang sukses dianggap sebagai suatu kegagalan keamanan.

Strategi – strategi untuk pengujian aplikasi web adalah sebagai berikut :

1. Model konten untuk aplikasi web ditinjau untuk menemukan kesalahan
2. Model antarmuka ditinjau untuk memastikan bahwa semua usecase dapat diakomodasikan
3. Mode perancangan untuk aplikasi web ditinjau untuk mengungkap kesalahan navigasi
4. Antarmuka pengguna diuji untuk mengungkap kesalahan dalam presentasi dan mekanik navigasi
5. Komponen fungsional diuji untuk setiap unit
6. Navigasi seluruh arsitektur diuji
7. Aplikasi web diimplementasikan dalam berbagai konfigurasi lingkungan yang berbeda dan diuji kompatibilitasnya pada masing – masing konfigurasi.
8. Pengujian keamanan dilakukan dalam upaya menyikap kelemahan – kelamahan dalam aplikasi web atau kelemahan dalam lingkungannya.
9. Pengujian kinerja dilakukan
10. Aplikasi web diuji oleh populasi pengguna akhir yang dikontrol dan dipantau, hasil interaksi mereka dengan sistem kemudian dievaluasi untuk menemukan kesalahan isi dan navigasi , kegunaan – kegunaan penting, keprihatinan terkait dengan kesesuaian , dan keamanan , keandalan dan kinerja aplikasi.

Untuk perencanaan pengujian aplikasi web dilakukan dengan menyiapkan himpunan tugas – tugas yang diterapkan ketika pengujian dimulai, produk kerja yang dihasilkan ketika pengujian dilakukan, cara dimana hasil pengujian dievaluasi, dicatat dan digunakan kembali pada saat pengujian regresi dilakukan.

Sedangkan proses pengujian aplikasi web adalah berupa tinjauan dengan menjalankan fungsionalitas isi dan antarmuka yang terlihat pada pengguna akhir. Berikut adalah proses – proses pengujian aplikasi web :

1. Pengujian Isi, dilaksanakan dengan mengungkap kesalahan isi yang dapat ditelusuri ke isi dinamis yang digerakkan oleh data.
2. Pengujian basis data
3. Pengujian antarmuka pengguna
4. Pengujian kegunaan
5. Pengujian kompatibilitas
6. Pengujian peringkat komponen
7. Pengujian navigasi
8. Pengujian konfigurasi
9. Pengujian keamanan
10. Pengujian kinerja

BAB 8 Manajemen Konfigurasi

Perangkat Lunak

8.I. Perubahan

Ketika sebuah organisasi atau perusahaan melakukan pengembangan perangkat lunak , pasti akan banyak sekali aktor – aktor yang terlibat . Aktor – aktor ini merupakan sumber daya manusia yang tidak lepas dari berfikir . Faktor sumber daya manusia ini dapat mempengaruhi proses pengembangan perangkat lunak yang diinginkan oleh perusahaan tersebut. Karena sumber daya manusia ini dapat memicu terjadinya perubahan , sehingga perubahan tidak dapat dihindarkan ketika proses pengembangan perangkat lunak. Perubahan – perubahan ini kadang dapat membuat masalah atau kebingungan pada rekayasaan perangkat lunak yang bekerja pada proyek pengembangan perangkat lunak.

Jika kebingungan muncul bila perubahan – perubahan yang terjadi tidak dianalisis sebelum perubahan dilaksanakan, dicatat sebelum diimplementasikan , dilaporkan kepada yang ingin mengetahui atau dikontrol dengan suatu cara yang akan meningkatkan kualitas dan mengurangi error. Manajemen konfigurasi perangkat lunak merupakan payung kegiatan yang dilaksanakan selama proses pengembangan perangkat lunak. Manajemen konfigurasi perangkat lunak dapat disebut juga *software configuration management (SCI)*

8.2. Tujuan Manajemen Konfigurasi Perangkat Lunak

Tujuan dari manajemen konfigurasi perangkat lunak adalah :

1. Mengidentifikasi perubahan
2. Mengontrol perubahan
3. Mengimplementasikan perubahan dengan benar
4. Melaporkan perubahan kepada pihak – pihak yang mempunyai kepentingan

8.3. Maintenance Perangkat Lunak VS Manajemen

Konfigurasi Perangkat Lunak

Penting sekali untuk membedakan dengan jelas antara maintenance perangkat lunak dan manajemen konfigurasi perangkat lunak . Maintenance perangkat lunak merupakan serangkaian aktivitas rekayasa perangkat lunak yang terjadi setelah perangkat lunak diserahkan ke pelanggan dan telah dioperasikan. Sedangkan manajemen konfigurasi perangkat lunak adalah serangkaian kegiatan peneleusuran dan kontrol yang dimulai ketika suatu proyek perangkat lunak dimulai dan berakhir ketika perangkat lunak sudah tidak beroperasi lagi. Jadi maintenance perangkat lunak merupakan bagian yang dilakukan penelusuran dan kontrol oleh manajemen perangkat lunak.

8.4. Informasi dan Perubahan

Proses pada perangkat lunak menghasilkan sebuah informasi yang dapat dikategorikasikan sebagai berikut :

- I. Perangkat lunak , program ini dapat berbentuk source code maupun executable

2. Dokumen – dokumen, dokumen yang menjelaskan program perangkat lunak tersebut yang ditujukan untuk praktisi teknik maun pengguna.
3. Data , merupakan hasil dari inputan pada perangkat lunak tersebut.

Dari proses pengembangan perangkat lunak tersebut menghasilkan sebuah beberapa informasi yang terdiri dari beberapa item yang disebut dengan item konfigurasi perangkat lunak atau *software configuration item* (SCI). SCI akan berkembang pesat selama proses rekayasa perangkat lunak, contohnya adalah *system specification* menghasilkan *software project plan* dan *software requirement specification* yang secara berurutan akan menghasilkan dokumen – dokuen untuk menciptakan hirarki informasi.

8.5. Sumber Dasar Perubahan

Dalam manajemen konfigurasi perangkat lunak terdapat beberapa sumber yang dapat menjadi sumber perubahan diantaranya adalah sebagai berikut :

1. Bisnis baru atau kondisi pasar baru yang mendiktekan perubahan – perubahan dalam produk atau aturan bisnis .
2. Keinginan pelanggan baru yang meminta modifikasi data yang dihasilkan oleh sistem informasi, fungsionalitas yang diberikan oleh produk atau layanan yang diberikan oleh suatu sistem terkomputerisasi.
3. Reorganisasi dan perampingan bisnis yang menyebabkan perubahan prioritas proyek atau struktur tim rekayasa perangkat lunak.

4. Kendala – kendala anggaran atau jadwal yang menyebabkan redefinisi sistem atau produk.

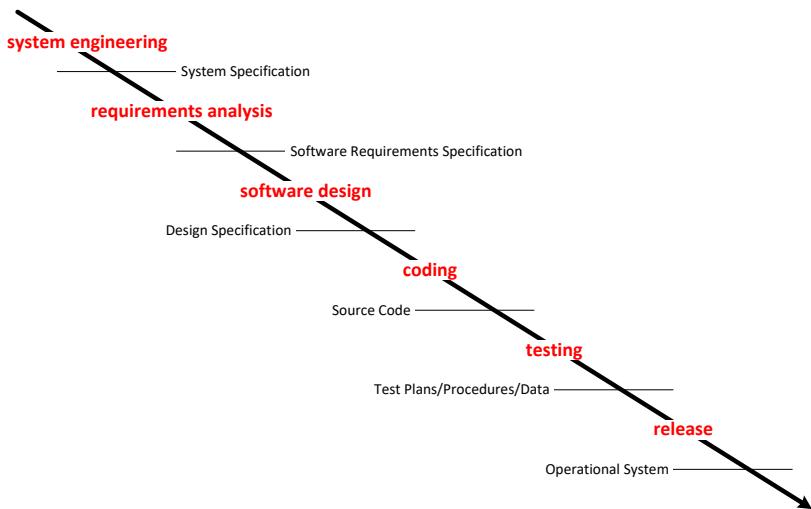
Manajemen konfigurasi perangkat lunak membantu dengan memberikan kerangka kerja untuk menangani perubahan – perubahan tersebut selama siklus hidup dari perangkat lunak. Oleh karena itu manajemen konfigurasi perangkat lunak dapat dipandang sebagai kegiatan *software quality assurance* yang dipakai selama siklus hidup perangkat lunak.

8.6. Baseline

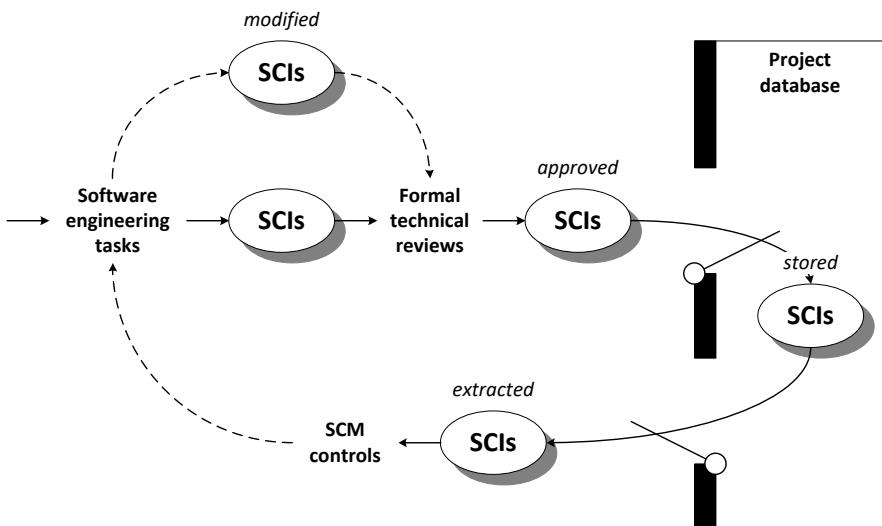
Perubahan – perubahan banyak terjadi selama siklus hidup perangkat lunak, seperti pelanggan yang ingin memodifikasi persyaratan – persyaratan , pengembang yang ingin memodifikasi metode – metode teknik , manager yang ingin memodifikasi cara peendekatan proyek , dan lain – lain. Tentunya manajemen konfigurasi perangkat lunak memiliki suatu alat yang dapat digunakan untuk mengatasi perubahan – perubahan tersebut. Dalam konsep manajemen konfigurasi perangkat lunak dikenal *baseline* yang merupakan sebuah konsep yang membantu dalam kontrol perubahan - perubahan tanpa menghalangi perubahan – perubahan yang dapat di justifikasi.

IEEE mendefinisikan *baseline* sebagai suatu spesifikasi atau produk yang telah direview secara formal dan disetujui bersama , yang selanjutnya difungsikan sebagai dasar bagi pengembangan lebih lanjut , serta dapat diubah hanya melalui prosedur – prosedur kontrol perubaha formal. Dalam konteks rekayasa perangkat lunak , *baseline* adalah *milestone* rekayasa perangkat lunak yang ditandai dengan penyampaian item

konfigurasi perangkat lunak dan persetujuan item konfigurasi perangkat lunak.



Gambar 8.1 Baseline.



Gambar 8.2 Baseline & Sofware Configuration items.

8.7. Item Konfigurasi Perangkat Lunak

Software configuration item merupakan informasi yang diciptakan sebagai bagian dari proses rekayasa perangkat lunak. *Software configuration item* menjadi target untuk manajemen konfigurasi dan membentuk sekumpulan baseline. Berikut adalah item konfigurasi perangkat lunak atau *software configuration item* :

1. *System specification*
2. *software project plan*
3. *software requirement specification*
 - a. *Graphical analysis model*
 - b. *Process specification*

- c. *Prototype*
 - d. *Mathematical specification*
- 4. *Preliminary user manual*
- 5. *Design specification*
 - a. *Data design description*
 - b. *Architectural design descriptions*
 - c. *Modul design descriptions*
 - d. *Interface design descriptions*
 - e. *Object description*
- 6. *Source code listing*
- 7. *Test specification*
 - a. *Test plan & procedure*
 - b. *Test cases & recorded result*
- 8. *Operation & installation manuals*
- 9. *Executable program*
 - a. *Module executable code*
 - b. *Linked modules*
- 10. *Database Description*
 - a. *Schema & file structure*
 - b. *Initial content*
- 11. *As-built user manual*
- 12. *Maintenance documents*
 - a. *Software problem reports*
 - b. *Maintenance requests*
 - c. *Engineering change orders*
- 13. *Standard & procedure for software engineering*

8.8. Proses Manajemen Konfigurasi Perangkat Lunak

Software configuration management merupakan salah satu elemen dari *software quality assurance*. Tanggung jawab utamanya adalah mengontrol perubahan. *Software configuration management* bertanggung jawab untuk

mengidentifikasi individual *software configuration item* dan berbagai versi perangkat lunak , mengaudit *software configuration* untuk memastikan bahwa dia telah dikembangkan dengan benar , melaporkan semua perubahan yang telah dilakukan pada konfigurasi tersebut.

Daftar Pustaka

- Ambler, S. W. (2018, 07 18). *Class Responsibility Collaborator (CRC) Models: An Agile Introduction*. Retrieved from agilemodeling.com:
<http://www.agilemodeling.com/artifacts/crcModel.htm>
- AZM Ehtesham Chowdhury, A. B. (2017). Analysis of the Veracities of Industry Used Software Development Life Cycle Methodologies. *AIUB Jurnal Sceince and Engineering Vol. 16 Issue . 01*, 1-8.
- binus. (2014, April 30). *DIAGRAM SWIMLANE*. Retrieved from sis.binus.ac.id:
<http://sis.binus.ac.id/2014/04/30/diagram-swimlane/>
- Bucanac, C. (2018, 07 05). *The V Model*. Retrieved from bucanac.com:
http://www.bucanac.com/documents/The_V-Model.pdf
- conceptdraw. (2018, 07 18). *Swim Lane Flowchart Symbols*. Retrieved from www.conceptdraw.com:
<https://www.conceptdraw.com/How-To-Guide/swim-lane-flowchart-symbols>
- Dara. (2017). *MVC dan Struktur pada Laravel*. Retrieved from [http://lea.si.fti.unand.ac.id:](http://lea.si.fti.unand.ac.id/)
<http://lea.si.fti.unand.ac.id/2017/04/mvc-pada-laravel/>
- Darmastuti. (2018, 07 19). *Data Flow Diagram*. Retrieved from gunadarma.ac.id:
<http://darmastuti.staff.gunadarma.ac.id/Downloads/files/59129/Data+Flow+Diagram.pdf>

- Desy. (2018, 07 19). *class Diagram*. Retrieved from lecturer.pens.a.id:
http://desy.lecturer.pens.ac.id/Workshop%20Pengembangan%20Perangkat%20Lunak/4_Class%20Diagram.pdf
- Fowler, M. (2018, 08 01). *Sequence Diagram Tutorial*. Retrieved from csis.pace.edu:
<http://csis.pace.edu/~marchese/CS389/L9/Sequence%20Diagram%20Tutorial.pdf>
- Gunadarma.ac.id. (2018, 07 19). *Activity Diagram & State Diagram*. Retrieved from gunadarma.ac.id:
<http://dhedee29.staff.gunadarma.ac.id/Downloads/files/36721/08-activity-and-state.pptx>
- Hamdani, D. (2014, Juni 09). *Objek Oriented Programming*. Retrieved from <http://www.tutorialkampus.com>:
<http://www.tutorialkampus.com/2014/06/perancangan-sistem-informasi-geografi.html>
- Inflectra. (2018, 03 01). *Use Cases and Scenarios*. Retrieved from inflectra.com: <https://www.inflectra.com/ideas/topic/use-cases.aspx>
- IT, M. (2014, II 30). *Mengenal The Concurrent Development Model*. Retrieved from materi-it.com: <http://www.materi-it.com/2014/II/mengenal-concurrent-development-model.html>
- Joseph S. Valacich, J. F. (2015). *Modern System Analysis and Design*. United State of Americ: Pearson Education, Inc.
- Kenneth E. Kendall, J. E. (2010). *System Analysis and Design*. United State of America: Pearson Education, Inc.

- Maxxor. (2018, 07 12). *SCRUM Software Development Process*. Retrieved from www.maxxor.com:
<https://www.maxxor.com/software-development-process>
- Receptives, T. (2018, 07 6). *Spiral Methodology*. Retrieved from techreceptives.com:
<https://techreceptives.com/development-methodology-spiral>
- Roger S. Pressman, P. (2012). *Rekayasa Perangkat Lunak , Pendekatan Praktis Edisi 7*. Yogyakarta: Andi.
- Rosa A. S, M. S. (2018). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Informatika.
- SANTOS, J. M. (2017, 07 18). *XP, FDD, DSDM, and Crystal Methods of Agile Development*. Retrieved from project-management.com: <https://project-management.com/xp-fdd-dsdm-and-crystal-methods-of-agile-development/>
- Satrio Agung W, A. K. (2011). *Database Entity Relationship Diagram*. Retrieved from power.lecture.ub.ac.id:
<http://power.lecture.ub.ac.id/files/2015/03/Modul-Basis-Data-I-3-ERD.pdf>
- Sommerville, I. (2003). *Software Engineering (Rekayasa Perangkat Lunak) Edisi 6 Jilid 1*. Jakarta: Erlangga.
- tatvasoft. (2015, 04 15). *Top 12 Software Development Methodologies & its Advantages / Disadvantages*. Retrieved from www.tatvasoft.com:
<https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/>

uml-diagrams.org. (2018, 07 20). *State Machine Diagrams*.

Retrieved from uml-diagrams.org: <https://www.uml-diagrams.org/state-machine-diagrams.html>

uml-diagrams.org. (2018, 07 16). *UML Activity Diagram Controls*.

Retrieved from uml-diagrams.org: <https://www.uml-diagrams.org/activity-diagrams-controls.html>

Victoria University of Wellington, N. Z. (2018, 07 16).

ecs.victoria.ac.nz. Retrieved from use case and uml:
https://ecs.victoria.ac.nz/Courses/ENGR110_2016T2/Assignment5