

**Mei Prabowo, M.Com.**

# **METODOLOGI PENGEMBANGAN SISTEM INFORMASI**



**Editor: Avin Wimar Budyastomo, M. Kom**



# **METODOLOGI PENGEMBANGAN SISTEM INFORMASI**

Mei Prabowo

**Editor:**

Avin Wimar Budyastomo, M.Kom

Lembaga Penelitian dan Pengabdian Kepada Masyarakat (LP2M)  
Intitut Agama Islam Negeri (IAIN) Salatiga

## **METODOLOGI PENGEMBANGAN SISTEM INFORMASI**

### **Penulis:**

Mei Prabowo, M.Kom

### **Editor:**

Avin Wimar Budyastomo, M.Kom

### **Cetakan: 2020**

15,5 x 23 cm; vi + 158 hlm.

**ISBN: 978-602-5916-88-5**

### **Penerbit:**

Lembaga Penelitian dan Pengabdian kepada Masyarakat (LP2M) IAIN Salatiga

Jl. Tentara Pelajar 02, Kode Pos 50721, Salatiga

E-mail: lp2miainsalatiga@gmail.com

### **ANGGOTA IKAPI**

*All Right reserved.* Hak cipta dilindungi undang-undang. Dilarang memperbanyak sebagian atau seluruh isi buku ini dalam bentuk apa pun tanpa ijin tertulis dari penerbit

# KATA PENGANTAR

Puji syukur kami sampaikan kepada Allah SWT atas selesainya penulisan buku ini. Buku ini kami susun berdasarkan pengalaman mengajar dari tahun 2017 di IAIN Salatiga sampai sekarang dan berbagai seminar maupun pelatihan sebagai Dosen Matakuliah Sistem Informasi Manajemen. Konsep penyusunan materi berdasarkan silabus materi perkuliahan Sistem Informasi Manajemen di IAIN Salatiga. Pembahasan setiap materi adalah dengan memahami landasan teori dasar disertai dengan studi kasus yang sangat mudah dimengerti untuk memahami teori dasar.

Adapun tujuan penulis buku ini adalah sebagai berikut :

1. Sebagai buku acuan melengkapi materi perkuliahan yang diberikan Dosen pada perkuliahan Sistem Informasi Manajemen di tingkat perguruan tinggi.
2. Membantu mahasiswa yang belum mengerti materi kuliah pada saat perkuliahan atau berhalangan hadir mengikuti perkuliahan karena sesuatu hal.
3. Membantu mahasiswa dalam materi perkuliahan yang lain yang penyelesaiannya berkaitan dengan Sistem Informasi Manajemen di Perusahaan.

Dengan menggunakan buku ini kami yakin mahasiswa dapat mengatasi kesukaran dalam menghadapi matakuliah Sistem Informasi Manajemen. Kepada Mahasiswa/i yang mengalami kesulitan kami berharap Matakuliah Sistem Informasi Manajemen dapat dikuasai dengan belajar secara tekun dan semangat yang tinggi.

Mei Prabowo

Akhirnya kami mengharapkan semoga buku ini bermanfaat dalam rangka ikut serta mencerdaskan generasi penerus bangsa Indonesia Tercinta.

Salatiga, September 2019  
Penulis

# DAFTAR ISI

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| KATA PENGANTAR..... | iii                                                     |
| DAFTAR ISI.....     | v                                                       |
| <b>BAB I</b>        | <b>KONSEP DASAR SISTEM INFORMASI..... 1</b>             |
| 1.1                 | Konsep Dasar Sistem ..... 1                             |
| 1.2                 | Konsep Dasar Informasi..... 5                           |
| 1.3                 | Sistem Informasi..... 10                                |
| <b>BAB II</b>       | <b>ANALISIS SISTEM.....20</b>                           |
| 2.1                 | Analisis Sistem.....20                                  |
| 2.2                 | Desain Sistem.....24                                    |
| 2.3                 | Pendekatan Analisis Sistem .....26                      |
| <b>BAB III</b>      | <b>PENDEKATAN PENGEMBANGAN SISTEM.....30</b>            |
| 3.1                 | Pandangan Metodologi .....30                            |
| 3.2                 | Pandangan Sasaran ( Sepotong dan Sistem ).....31        |
| 3.3                 | Penentuan Kebutuhan ( Bottom-Up dan<br>Top-Down).....32 |
| 3.4                 | Teknik Pengembangan<br>Menyeluruh dan Modular ) .....33 |
| 3.5                 | Teknologi ( Lompatan Jauh dan Berkembang ) .....34      |
| 3.6                 | Siklus Hidup Pengembangan Sistem ( SDLC ).....34        |
| <b>BAB IV</b>       | <b>WATERFALL MODEL.....37</b>                           |
| 4.1                 | Classical Waterfall .....37                             |
| 4.2                 | Lterative Waterfall.....43                              |
| 4.3                 | V-Model.....45                                          |
| 4.4                 | Prototyping.....52                                      |
| 4.5                 | Incremental Development .....56                         |

|                            |                                              |            |
|----------------------------|----------------------------------------------|------------|
| <b>BAB V</b>               | <b>AGLIE DEVELOPMENT MODEL .....</b>         | <b>62</b>  |
| 5.1                        | Adaptive Software Development (ASD) .....    | 65         |
| 5.2                        | Continuous Integration (CI).....             | 68         |
| 5.3                        | Crystal Methods .....                        | 70         |
| 5.4                        | Dynamic Systems Development Method (DSDM) .  | 73         |
| 5.5                        | Extreme Programming (XP).....                | 76         |
| 5.6                        | Feature Driven Development (FDD).....        | 81         |
| 5.7                        | Kanban.....                                  | 86         |
| 5.8                        | Lean software development .....              | 90         |
| 5.9                        | Rational Unified Process (RUP) .....         | 96         |
| 5.10                       | Scrum.....                                   | 102        |
| <b>BAB VI</b>              | <b>MODEL PENGEMBANGAN SISTEM LAINNYA ...</b> | <b>111</b> |
| 6.1                        | Rapid Application Development.....           | 111        |
| 6.2                        | Metode Spiral.....                           | 118        |
| <b>BAB VII</b>             | <b>PENGUJIAN PERANGKAT LUNAK .....</b>       | <b>124</b> |
| 7.1                        | Performance Testing .....                    | 124        |
| 7.2                        | System Testing, .....                        | 129        |
| 7.3                        | Unit Testing.....                            | 130        |
| 7.4                        | Integration Testing.....                     | 138        |
| <b>DAFTAR PUSTAKA.....</b> |                                              | <b>152</b> |

# **BAB I**

## **KONSEP DASAR SISTEM INFORMASI**

### **1.1 Konsep Dasar Sistem**

Pada umumnya setiap organisasi mempunyai sistem informasi dalam mengumpulkan, menyimpan, melihat, dan menyalurkan informasi dalam membuat perancangan sistem informasi. Konsep dasar sistem merupakan sekelompok komponen berbasis komputer yang dibuat oleh manusia dalam mengelola data, menyimpan, menghimpun kerangka kerja serta mengkoordinasikan sumber daya manusia dan komputer untuk mengubah sistem masukan menjadi sistem keluaran untuk mencapai tujuan dan sasaran yang telah ditetapkan sebelumnya.

#### **1.1.1 Pengertian Sistem**

Pembahasan tentang metode dalam pengembangan perangkat lunak, tidak lepas dari istilah sistem informasi, pemahaman tentang sistem informasi menjadi sangat penting sebagai awal dalam pengembangan perangkat lunak. Berikut ini dijelaskan tentang definisi sistem menurut para ahli secara umum :

Jogiyanto menerangkan bahwa sistem adalah kumpulan dari elemen yang berinteraksi untuk mencapai tujuan tertentu (Hartono M, 2005).

Abdul Kadir Sistem Adalah Sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai tujuan (Abdul, 2005).

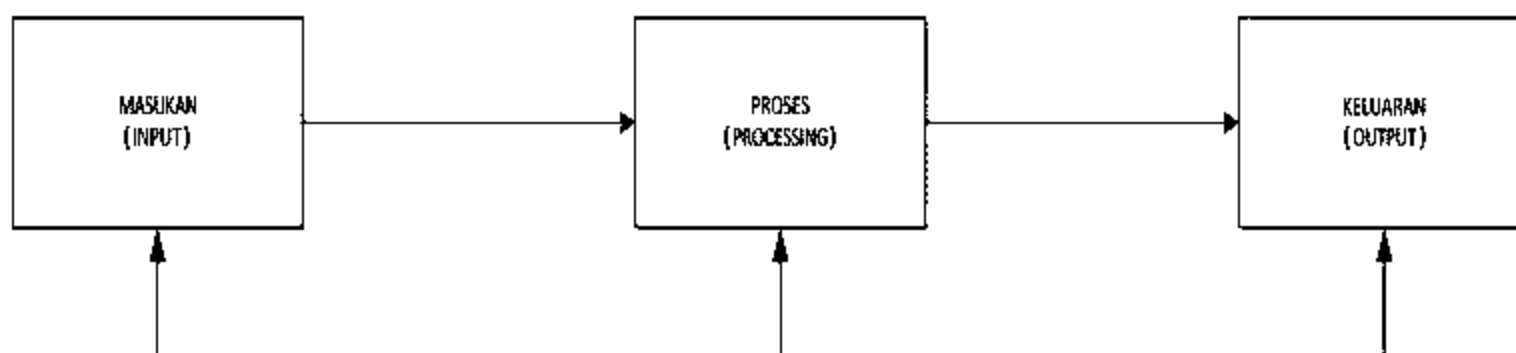
Mustakini Menjelaskan sistem merupakan sekumpulan dari prosedur-prosedur yang mempunyai tujuan tertentu “ (Mustakini, 2009).

Agus Mulyono mendefinisikan sistem secara umum yaitu kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu sebagai satu kesatuan (Mulyono, 2009).

Anastasia Diana & Lilis Setiawati mendefinisikan sistem merupakan serangkaian bagian yang sahng tergantung dan bekerja sama untuk mencapai tujuan tertentu(Diana & Setiawati, 2011)

Mulyadi mendefinisikan sistem sebagai suatu jaringan prosedur yang dibuat menurut pola yang terpadu untuk melaksanakan kegiatan pokok perusahaan (Mulyadi, 2014)

Penjelasan di atas menjelaskan bahwa sistem bekerja dalam suatu jaringan kerja dari suatu prosedur nyang saling berhubungan satu sama lain untuk menyelesaikan tujuan dan sasaran yang dimaksud. Mendefinisikan sistem dalam bidang sistem informasi sebagai “sekelompok komponen yang saling berhubungan, bekerja sama, untuk mencapai tujuan bersama dengan menerima proses input serta mengliasikan input dalam proses transformasi yang teratur”. Dari ketiga pengertian sistem diatas dapat disimpulkan bahwa sistem adalah sekumpulan unsure atau elemen-elemen yang saling terkait dan saling mempengaruhi dalam melakukan bersama untuk mencapai tujuan tertentu. Secara umum unsur – unsur yang terdapat pada sistem yaitu masukan (input), Proses (Processing) dan Keluaran (Output) hal ini digambarkan pada model sistem berikut ini :



Gambar 1. 1. Model Sistem

Pada gambar 1.1 tersebut menjelaskan bahwa suatu sistem minimal harus mempunyai tiga komponen utama tersebut yakni masukan, pengelolaan dan keluaran. selain itu satu hal yang perlu diperhatikan adalah umpan balik atau kontrol dalam sistem.

### 1.1.2 Karakteristik Sistem

Sebuah sistem terdiri dari elemen – elemen yang saling berkaitan yang berkaitan untuk mencapai beberapa saran atau maksud. Suatu sistem mempunyai karakteristik atau sifat-sifat yang tertentu, yang mempunyai komponen-komponen, data sistem, lingkungan luar sistem, masukan, keluar, pengolah, dan sasaran atau tujuan (Yulianeu & Noer, 2016). Sistem memiliki ciri-ciri karakteristik yang terdapat pada sekumpulan elemen yang harus dipahami dalam mengidentifikasi pembuatan sistem tersebut. Berikut ini merupakan karakteristik dari suatu sistem antara lain (Hutahaean, 2015) :

a. **Komponen**

Sistem meliputi kumpulan komponen yang saling berinteraksi dan bekerja sama untuk membentuk satu kesatuan. Komponen sistem dapat berupa sub sistem atau bagian-bagian dari sistem.

b. **Batasan sistem (boundary)**

Merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya, dan dapat pula menunjukkan ruang lingkup dari sistem. Batasan sistem ini memungkinkan sistem dipandang sebagai satu kesatuan dan juga menunjukkan ruang lingkup (scope) dari sistem tersebut.

c. **Lingkungan luar sistem (environment)**

Apapun yang berada di luar batas dari sistem dan mempengaruhi sistem tersebut dinamakan dengan lingkungan luar sistem. Lingkungan luar yang bersifat menguntungkan wajib dipelihara dan yang merugikan harus dikendalikan agar tidak mengganggu kelangsungan sistem.

d. Penghubung sistem (interface)

Sistem merupakan sumber-sumber daya yang mengalir diantara subsistem-subsistem. Media penghubung diperlukan untuk mengalirkan sumber-sumber daya dari sub sistem ke sub sistem lainnya dinamakan dengan penghubung sistem.

e. Masukkan sistem (input)

Energi yang dimasukkan ke dalam sistem dinamakan dengan masukan sistem (input) dapat berupa perawatan dan masukan sinyal. Perawatan ini berfungsi agar sistem dapat beroperasi dan masukan sinyal adalah energi yang diproses untuk menghasilkan keluaran (output).

f. Keluaran sistem (output)

Hasil dari energi yang telah diolah dan diklasifikasikan menjadi keluaran yang berguna dinamakan dengan keluaran sistem (output). Informasi merupakan contoh keluaran sistem.

g. Pengolah sistem

Untuk mengolah masukan menjadi keluaran diperlukan suatu pengolah yang dinamakan dengan pengolali sistem.

h. Sasaran sistem

Sistem pasti memiliki tujuan atau sasaran yang sangat menentukan input yang dibutuhkan oleh sistem dan keluaran yang dihasilkan.

### 1.1.3 Klasifikasi sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dan komponen lain karena sistem memiliki sasaran yang berbeda untuk setiap kasus yang terjadi di dalam sistem tersebut. Sistem dapat diklasifikasikan dari beberapa sudut pandang diantaranya (Yakub, 2012):

a. Sistem abstrak (abstract system)

Sistem Abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik. Sistem teologia yang berisi gagasan tentang hubungan manusia dengan Tuhan merupakan

contoh abstract system.

b. Sistem fisik (physical system)

Sistem fisik adalah sistem yang ada secara fisik, Sistem komputer, sistem akuntansi, sistem produksi, sistem sekolah, dan sistem transportasi merupakan contoh physical system.

c. Sistem tertentu (deterministic system)

Sistem tertentu adalah sistem yang beroperasi dengan tingkah laku yang dapat diprediksi, interaksi antara bagian dapat dideteksi dengan pasti sehingga keluarannya dapat diramalkan. Sistem komputer sudah diprogramkan, merupakan contoh deterministic system karena program komputer dapat diprediksi dengan pasti.

d. Sistem tak tentu (probabilistic system)

Sistem tak tentu adalah suatu sistem yang kondisi masa depannya tidak dapat diprediksikan karena mengandung unsur probabilitas. Sistem arisan merupakan contoh probabilistic system karena sistem arisan tidak dapat diprediksikan dengan pasti.

e. Sistem tertutup (close system)

Sistem tertutup merupakan sistem yang tidak bertukar materi, informasi, atau energi dengan lingkungan. Sistem ini tidak berinteraksi dan tidak dipengaruhi oleh lingkungan, misalnya reaksi kimia dalam tabung terisolasi.

f. Sistem terbuka (open system)

Sistem ini adalah sistem yang berhubungan dengan lingkungan dan dipengaruhi oleh lingkungan. Sistem perdagangan merupakan contoh open system, karena dapat dipengaruhi oleh lingkungan.

## 1.2 Konsep Dasar Informasi

### 1.2.1 Pengertian Informasi

Secara umum informasi bisa diartikan sebagai data yang memiliki nilai kebergunaan. Data tersebut merupakan nilai, keadaan, serta

mempunyai sifat berdiri sendiri lepas dari konteks apapun. Data dapat berupa catatan-catatan dalam kertas, buku, atau tersimpan sebagai file dalam basis data. Berikut ini merupakan pendapat para ahli terkait definisi dari informasi.

*Azhar Susanto “Informasi merupakan hasil dari pemrosesan data, akan tetapi tidak semua dari hasil pemrosesan data tersebut bisa menjadi informasi (Susanto, 2000)”.*

*Edhy Susanta “Informasi merupakan hasil pengolahan data sehingga menjadi bentuk yang penting bagi penerimanya dan mempunyai kegunaan sebagai dasar dalam pengambilan keputusan yang dapat dirasakan akibatnya secara langsung saat itu juga atau secara tidak langsung pada masa yang akan datang (Sutanta, 2003)”.*

*Jogiyanto “Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya (Hartono M, 2005)”.*

*Yakub “data yang diolah menjadi bentuk lebih berguna dan lebih berarti bagi yang menerimanya (Yakub, 2012)”.*

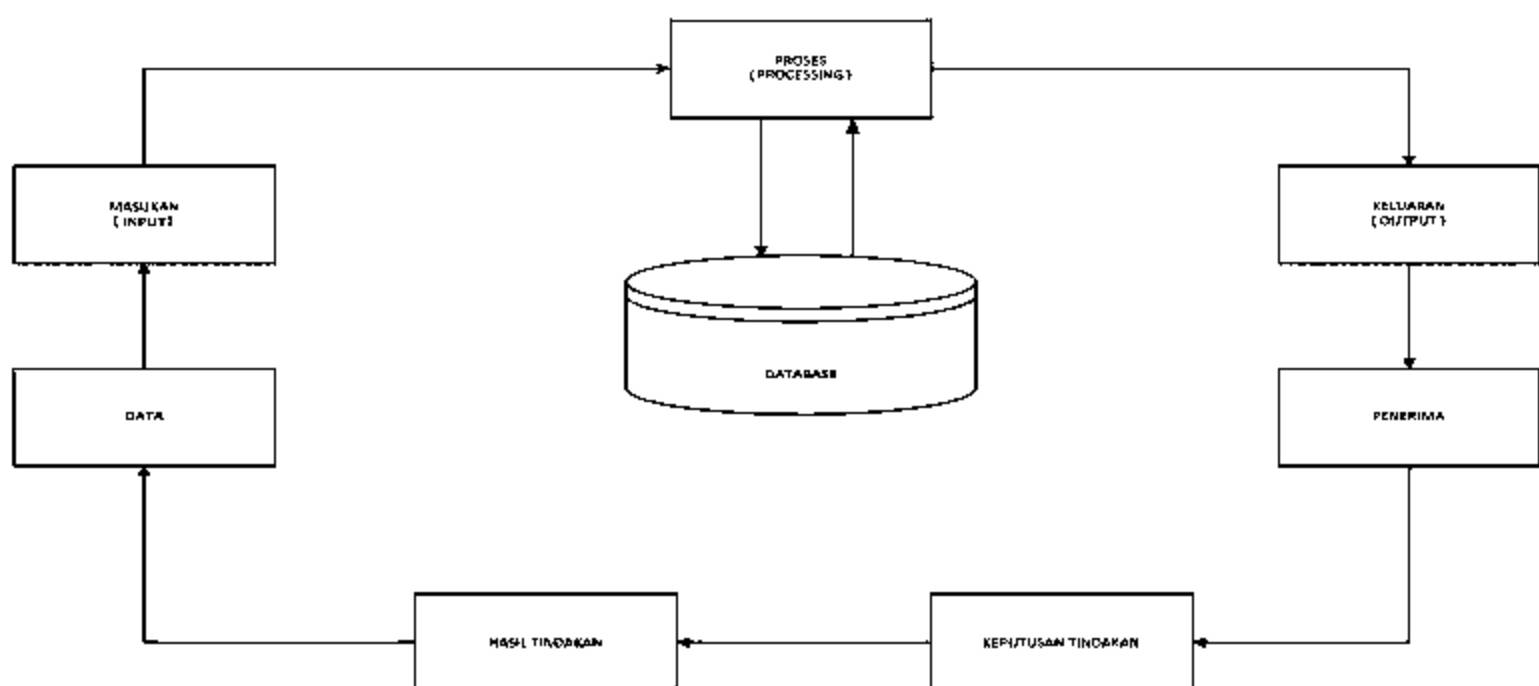
*Tata Sutabri “Data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan (Sutabri, 2012)”.*

Dari pengertian diatas dapat disimpulkan Informasi merupakan data yang telah diolah, dibentuk, ataupun dimanipulasi sesuai dengan keperluan tertentu bagi penggunaanya sehingga menghasilkan nilai yang lebih bermakna dan bermanfaat bagi penerimanya. Transformasi data menjadi informasi dapat digambarkan sebagaimana ditunjukkan oleh gambar di bawah ini. Dalam gambar tersebut, input adalah data yang akan diolah oleh unit pengolah, dan output adalah informasi sebagai hasil pengolahan data yang telah diinputkan tersebut. Suatu

unit penyimpan diperlukan sebagai alat simpanan data, pengolah, maupun informasi.

### 1.2.2 Fungsi dan Siklus Informasi

Menurut Edhy Susanta suatu informasi memiliki beberapa fungsi utama antara lain sebagai penambah pengetahuan, mengurangi ketidakpastian, mengurangi resiko kegagalan, mengurangi keanekaragaman/variasi yang tidak diperlukan, serta memberi standar, aturan-aturan, ukuran-ukuran, dan keputusankeputusan yang menentukan pencapaian sasaran dan tujuan (Susanta, 2003). Data agar menjadi lebih berarti dan berguna dalam bentuk informasi, maka perlu diolah melalui suatu model tertentu. Data yang telah diolah tersebut kemudian diterima oleh penerima, lalu penerima membuat suatu keputusan dan melakukan tindakan, yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagai input, dan diproses kembali lewat suatu model dan seterusnya sehingga membentuk suatu siklus. Berikut ini merupakan gambaran dari siklus informasi :



Gambar 1. 2 Siklus Informasi

### 1.2.3 Nilai dan Kualitas Informasi

Nilai informasi (value of information) ditentukan oleh dua hal yaitu manfaat dan biaya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya . Jogyanto berpendapat bahwa informasi dikatakan bernilai apabila informasi lebih efektif dibandingkan dengan biaya mendapatkannya (Hartono M, 2005). Informasi berguna untuk mengurangi ketidakpastian di dalam proses pengambilan keputusan tentang suatu keadaan. Nilai dari informasi juga ditentukan dari dua hal yaitu biaya dan manfaat dalam mendapatkannya. Namun, perlu juga digarisbawahi bahwa informasi yang digunakan di dalam suatu sistem, umumnya digunakan untuk beberapa kegunaan. Berikut ini disampaikan delapan kriteria yang dapat digunakan untuk menentukan nilai dari suatu informasi. Penjelasan tentang kualitas informasi tersebut dipaparkan di bawah (Hartono, 2013):

a. Relevansi

Informasi disediakan atau disajikan untuk digunakan. Oleh karena itu, informasi yang bernilai tinggi adalah yang relevan dengan kebutuhan, yaitu untuk apa informasi itu akan digunakan

b. Kelengkapan dan Keluasan.

Informasi akan bernilai semakin tinggi, jika tersaji secara lengkap dalam cakupan yang luas. Informasi yang sepotong-sepotong, apalagi tidak tersusun sistematis, tentu tidak akan banyak artinya. Demikian pun bila informasi itu hanya mencangkup area yang sempit dari suatu permasalahan.

c. Kebenaran

Kebenaran informasi ditentukan oleh validitas atau dapatnya dibuktikan. Informasi berasal dari data, dan data fakta. Informasi yang bernilai tinggi adalah informasi yang benar-benar berasal dari fakta, bukan opini atau ilus

d. Terukur

Informasi berasal dari data atau hasil pengukuran dan pencacatan terhadap fakta. Jadi, informasi yang bernilai tinggi adalah informasi yang jika dilacak kembali kepada datanya, data tersebut dapat diukur sesuai dengan faktanya.

e. Keakuratan

Informasi berasal dari data atau hasil pengukuran dan pencacatan terhadap fakta. Oleh karena itu kecermatan dalam mengukur dan mencatat fakta akan menentukan keakuratan data dan nilai dari informasi yang dihasilkan.

f. Kejelasan

Informasi dapat disajikan dalam berbagai bentuk teks, tabel, grafik, chart, dan lain-lain. Namun, apa pun bentuk yang dipilih, yang penting adalah menjadikan pemakai mudah memahami maknanya. Oleh sebab itu, selain bentuk penyajiannya harus benar, juga harus diperhatikan kemampuan pemakai dalam memahaminya.

g. Keluwesan

Informasi yang baik adalah yang mudah diubah-ubah bentuk penyajiannya sesuai dengan kebutuhan dan situasi yang dihadapi.

h. Ketepatan

Waktu Informasi yang baik adalah informasi yang disajikan tepat pada saat dibutuhkan. Informasi yang terlambat datang menjadi informasi basi yang tidak ada lagi nilainya (misalnya untuk pengambilan keputusan).

#### 1.2.4 Transformasi Informasi

Transformasi informasi adalah komponen proses dalam pengelolaan sistem informasi, yang berfungsi memproses data menjadi informasi sehingga dapat diperoleh produk informasi yang diperlukan. Transformasi informasi yang bertujuan untuk memproses

data menjadi informasi, dari proses tersebut akan menghasilkan output yang berupa produk-produk informasi yang sesuai dengan kebutuhan penggunaanya. Konsep Transformasi Informasi Pada hakikatnya transformasi informasi merupakan suatu proses perubahan wujud, sifat, ciri-ciri data menjadi sebuah informasi yang selanjutnya disajikan secara statistika atau secara visual untuk di sebarluaskan dan di dokumentasikan.

Proses transformasi informasi menjadi pengetahuan melalui empat tahapan yaitu 1) Comparison adalah membandingkan informasi pada suatu situasi tertentu dengan situasi lainnya yang telah diketahui. 2) Consequences adalah menemukan implikasi-implikasi dari informasi yang bermanfaat untuk pengambilan keputusan dan tindakan. 3) Connections adalah menemukan hubungan-hubungan bagianbagian kecil dari informasi dengan hal-hal lainnya. 4) Conversations adalah membicarakan pandangan, pendapat, serta tindakan orang lain terkait informasi tersebut (Munir, 2008).

Transformasi informasi pada hakikatnya merupakan suatu proses pengubahan wujud, sifat, ciri-ciri data sehingga menjadi informasi, yang selanjutnya disajikan secara statistika atau secara visual untuk disebarluaskan dan atau didokumentasikan. Proses transformasi ini bertitiktolak dari data yang dikumpulkan dari sumber dengan menggunakan alat atau *instrument* pengumpulan data, selanjutnya data itu diolah, dianalisis dan ditafsirkan dengan teknik tertentu. Data yang telah diproses itu membuahkan hasil yang disebut informasi.

### 1.3 Sistem Informasi

Setelah mengenal definisi dari sistem dan informasi maka selanjutnya dalam bahasan ini akan membahas tentang pengertian sistem informasi, komponen sistem informasi, tipe – tipe sistem informasi, perancangan sistem informasi, pengelolaan sistem informasi, pengendalian sisem informasi:

### 1.3.1 Pengertian Sistem Informasi

Secara umum Sistem informasi bisa di artikan suatu sistem yang sahng terintegrasi yang dapat menyediakan informasi yang bermanfaat bagi penggunaanya. Sistem Informasi memiliki komponen – komponen dasar : Perangkat keras komputer, perangkat lunak komputer, basis data, jaringan, prosedur, pengguna untuk pengelolaan operasi. Pengelola sistem Informasi memiliki tingkatan manajemen yang telah Terstruktur. Berikut ini definisi sistem informasi menurut para ahli :

*Menurut Mulyanto mendefinisikan sistem informasi adalah suatu komponen yang terdiri dari manusia, teknologi informasi, dan prosedur kerja yang memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk mencapai suatu tujuan (Mulyono, 2009)*

*Menurut Satzinger, Jackson, dan Burd, mendefinisikan Sistem informasi merupakan kumpulan dari komponen-komponen yang mengumpulkan, memproses, menyimpan, dan menyediakan output dari setiap informasi yang dibutuhkan dalam proses bisnis serta aplikasi yang digunakan melalui perangkat lunak, database dan bahkan proses manual yang terkait (John W. et al., 2012)*

*Jogiyanto mendefinisikan sistem informasi merupakan suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi yang bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan (Hartono M, 2005)*

*Menurut Hutahaeen mendefinisikan sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengelolaan transaksi harian, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi*

*dan menyediakan pihak luar tertentu dengan laporan-laporan yang dibutuhkan (Hutahaean, 2015)*

Berdasarkan pendapat para ahli tersebut dapat disimpulkan bahwa sistem informasi adalah suatu kombinasi modul yang terorganisir yang berasal dari komponen - komponen yang terkait dengan hardware, software, people dan network berdasarkan seperangkat komputer yang saling berhubungan atau berinteraksi untuk melakukan pengolahan data menjadi informasi untuk mencapai tujuan.

### **1.3.2 Komponen Sistem Informasi**

Komponen-komponen yang membangun sistem informasi dinamakan blok bangunan (building block). Blok bangunan ini mirip dengan karakteristik sistem yang telah diuraikan sebelumnya. Sistem informasi terdiri dari komponen-komponen yang disebut blok bangunan (building block), yang terdiri dari blok masukan, blok model, blok keluaran, blok teknologi, blok basis data, dan blok kendali. Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lain membentuk suatu kesatuan untuk mencapai sasaran (Sutabri, 2012):

- a. Blok masukan (masukan block)  
Masukan mewakili data yang masuk ke dalam sistem informasi. Masukan dapat berupa data seperti dokumen-dokumen dasar.
- b. Blok model (model block)  
Blok ini terdiri dari kombinasi prosedur, logika, dan model matematik yang akan memanipulasi data masukan dan data yang tersimpan di basis data dengan cara yang sudah ditentukan untuk menghasilkan keluaran yang diinginkan.
- c. Blok keluaran (keluaran block)  
Blok ini menghasilkan keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

d. Blok teknologi (technology block)

Blok ini digunakan untuk menerima masukan, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran, dan membantu pengendalian dari sistem secara keseluruhan

e. Blok basis data (basis data block)

Basis data merupakan kumpulan data yang saling berkaitan dan berhubungan satu sama lain, tersimpan di perangkat keras komputer dan menggunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan dalam basis data untuk keperluan penyediaan informasi lebih lanjut.

f. Blok kendali (control block)

Blok kendali dirancang untuk meyakinkan bahwa bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi.

Sedangkan Kusrini dan andri koniyo menyebutkan bahwa komponen/ elemen sistem informasi sebagai berikut (Kusrini & Koniyo, 2007):

- a. Perangkat Keras, mencakup berbagai piranti fisik seperti komputer dan printer.
- b. Perangkat Lunak, yaitu sekumpulan intruksi yang memungkinkan perangkat keras memproses data.
- c. Prosedur, yaitu sekumpulan aturan yang dipakai untuk mewujudkan pemrosesan data dan pembangkitan keluaran yang dikehendaki.
- d. Orang, yaitu semua pihak yang bertanggungjawab dalam pengembangan sistem informasi, pemrosesan dan penggunaan keluaran sistem informasi.
- e. Basis Data, yaitu sekumpulan tabel, hubungan dan lain-lain yang berkaitan dengan penyimpanan data.
- f. Jaringan Komputer dan Komunikasi Data, yaitu system penghubung yang memungkinkan sumber (resource) pakai bersama atau diakses sejumlah pemakai.

### 1.3.3 Tipe – tipe sistem informasi

Sistem Informasi Merupakan suatu kombinasi modul yang terorganisir yang berasal dari komponen-komponen yang terkait dengan hardware, software, people dan network berdasarkan seperangkat komputer yang saling berhubungan atau berinteraksi untuk melakukan pengolahan data menjadi informasi untuk mencapai tujuan. Sistem Informasi dapat dikembangkan sesuai kebutuhan dan sasaran yang ingin dicapai oleh suatu organisasi, baik itu terdiri dari beberapa sistem yang saling berkaitan maupun sebuah sistem informasi tunggal yang hanya berfokus untuk menyelesaikan satu tugas. Ada 6 (enam) tipe utama dari sistem informasi yaitu :

- a. Transaction processing systems (TPS)  
sistem terkomputerisasi yang menjalankan dan menyimpan transaksi rutin sehari-hari untuk menjalankan bisnis. Sistem ini bekerja pada level operasional. Input pada level ini adalah transaksi dan kejadian. Proses dalam sistem ini meliputi pengurutan data, melihat data, memperbaharui data. Sedangkan outputnya adalah laporan yang detail, daftar lengkap dan ringkasan.
- b. Knowledge work systems (KWS)  
Sistem informasi yang membuat dan mengintegrasikan pengetahuan baru ke organisasi.
- c. Office Automation Systems (OAS)  
Sistem komputer seperti pengolah kata, e-mail, dan sistem penjadwalan, yang didesain untuk meningkatkan produktifitas dari data workers di organisasi. Nomor 2 dan 3 melayani knowledge level.
- d. Management Information Systems  
Sistem informasi pada management-level sebuah organisasi yang melayani fungsifungsi perencanaan, pengendalian dan pengambilan keputusan yang dibuat dengan menyediakan ringkasan rutin dan laporan periodik.

- e. Decision-support systems (DSS)  
Sistem informasi di management-level sebuah organisasi yang mengkombinasikan data dan model analitis yang rumit untuk mendukung pengambilan keputusan yang Terstruktur dan semi Terstruktur.
- f. Executive support systems (ESS)  
Sistem informasi pada strategic-level sebuah organisasi yang dirancang untuk tujuan pengambilan keputusan yang tidak Terstruktur

#### **1.3.4 Perancangan sistem Informasi**

Pada saat membuat sistem membuat sebuah sistem yang akan digunakan pada suatu perusahaan, setiap pengembang aplikasi diharuskan membuat sebuah rancangan dari sistem yang ingin dibuat. Rancangan ini bertujuan untuk memberi gambaran umum dari sistem yang akan berjalan nantinya kepada setiap stakeholder. Perancangan sistem merupakan pelengkap dari analisa sistem ke dalam suatu sistem yang utuh dengan tujuan mendapatkan sistem yang lebih baik

Menurut Satzinger, Jackson dan Burdperancangan sistem adalah sekumpulan aktivitas yang menggambarkan secara rinci bagaimana sistem akan berjalan. Hal itu bertujuan untuk menghasilkan produk perangkat lunak yang sesuai dengan kebutuhan user (John W. et al., 2012).

Menurut Kenneth dan Jane mendefinisikan perancangan sistem adalah kegiatan merancang dan menentukan cara mengolah sistem informasi dari hasil analisa sistem sehingga sistem tersebut sesuai dengan requirement (Kenneth C. & Jane P., 2003)

Menurut O'Brien dan Marakas mendefinisikan perancangan sistem adalah sebuah kegiatan merancang dan menentukan cara mengolah sistem informasi dari hasil analisa sistem sehingga dapat memenuhi kebutuhan dari pengguna termasuk diantaranya

perancangan user interface, data dan aktivitas proses.

Dari beberapa teori-teori diatas dapat disimpulkan bahwa perancangan sistem adalah proses perancangan untuk merancang suatu sistem baru atau memperbaiki suatu sistem yang telah ada sehingga sistem tersebut menjadi lebih baik dan biasanya proses ini terdiri dari proses merancang input, output dan file.

Perancangan sistem merupakan pelengkap dari analisa sistem ke dalam suatu sistem yang utuh dengan tujuan mendapatkan sistem yang lebih baik. Terdapat 6 (enam) tahap analisis dalam sistem yaitu 1) Mengumumkan penelitian sistem, 2) Mengorganisasikan tim proyek, 3) Mendefinisikan kebutuhan informasi, 4) Mendefinisikan kriteria kinerja sistem, 5) Menyiapkan usulan rancangan, dan 6) Menyetujui atau menolak rancangan proyek.

Perancangan sistem adalah penentuan proses dan data yang diperlukan oleh sistem baru, jika sistem itu berbasis komputer, perancangan dapat menyertakan spesifikasi peralatan yang akan digunakan. Tahap perancangan sistem 1) Menyiapkan rancangan sistem yang terinci, 2) Mengidentifikasi berbagai alternatif sistem, 3) Mengevaluasi berbagai alternatif konfigurasi sistem, 4) Memilih konfigurasi terbaik, 5) Menyiapkan usulan penerapan, dan 6) Menyetujui atau menolak penerapan sistem

Dari kutipan-kutipan tersebut di atas, dapat disimpulkan bahwa perancangan sistem informasi merupakan proses penerjemahan kebutuhan pemakai informasi ke dalam suatu rancangan untuk memenuhi kebutuhan pemakai dan memberi gambaran yang lebih jelas untuk dijadikan pertimbangan.

### **1.3.5 Pengelolaan sistem Informasi**

Pengelolaan sistem informasi adalah bagian yang tak dapat dipisahkan dari studi manajemen, pengelolaan sistem informasi merupakan faktor kunci bagi keterlaksanaan dan keberhasilan

manajemen. Dengan tertatanya sistem ke arah yang lebih baik lagi tentunya mempunyai nilai positif bagi perkembangan organisasi. Pengguna informasi akan semakin percaya dan yakin terhadap produk yang dihasilkan. Dan tentunya tidak akan pindah ke lain organisasi. Manajer yang baik dan handal mempunyai tanggung jawab yang berat dalam mengelola organisasinya. Organisasi yang besar dan hebat memiliki manajer-manajer yang handal dan hebat dengan ide dan inovasi yang selalu mengikuti perubahan lingkungan usaha.

Adapun langkah-langkah dalam pengelolaan Sistem informasi sebagai berikut :

- a. Pengelolahan data, dapat berupa agregasi data sederhana menurut distrik, kalkulasi rata-rata atau analisis trend menurut waktu sampai penggunaan teknik-teknik statistik yang canggih contohnya analisis variansi.
- b. Penyajian informasi, ini dapat berbentuk tabel dan grafik atau Computer spread sheet .
- c. Menginterpretasikan informasi dengan mempertimbangkan kebenaran data atau angka, membandingkan angka sebelumnya dan untuk area geografis, dan penyebab masalah.
- d. Mengambil tindakan dengan cara mendiskusikan laporan dengan teman sekerja (kolega), menyediakan umpan balik, mengidentifikasi segala tindakan yang mungkin pada tiap interpretasi dan melakukan koreksi data.

#### **1.3.6 Pengendalian sistem informasi**

Pengendalian sistem informasi merupakan bagian yang tak dapat dipisahkan dari pengelolaan sistem informasi bahkan ia melaksanakan fungsi yang sangat penting karena mengamati setiap tahapan dalam proses pengolahan informasi. Pengelola informasi perlu memahami dan memiliki keterampilan manajerial dan melaksanakan kegiatan pengendalian sistem informasi, yakni : kemampuan mengendalikan

kegiatan perencanaan informasi, kemampuan mengendalikan proses transformasi informasi, kemampuan mengendalikan organisasi pelaksana sistem informasi, kemampuan melaksanakan kegiatan koordinasi, dengan kemampuan itu maka terjadilah kelancaran pelaksanaan pengelolaan sistem informasi guna mendukung keberhasilan program organisasi.

Pengendalian sistem informasi dilaksanakan melalui pengawasan dan pembinaan. Pengawasan dilakukan baik secara langsung yakni ditempat dimana dilaksanakannya sistem informasi tersebut, maupun secara tak langsung melalui laporan-laporan secara tertulis dan secara lisan.

Ada beberapa keterampilan untuk mengelola pengendalian sistem informasi, yaitu:

- a. Kemampuan mengendalikan kegiatan perencanaan informasi.
- b. Kemampuan mengendalikan proses transformasi informasi.
- c. Kemampuan mengendalikan organisasi pelaksana sistem informasi.
- d. Kemampuan-kemampuan kegiatan koordinasi.

Pengendalian sistem informasi adalah keseluruhan kegiatan dalam bentuk mengamati, membina, dan mengawasi pelaksanaan mekanisme pengelolaan sistem informasi (Sutabri, 2012). Organisasi pada saat ini bergantung pada teknologi informasi (TI), seperti memindahkan sebagian dari sistem informasinya ke cloud. Untuk mengatasi permasalahan pengendalian tersebut, AICPA dan CICA mengembangkan Trust Service Framework untuk menyediakan panduan penilaian keandalan sistem informasi. Trust Service Framework mengatur pengendalian TI ke dalam lima prinsip yang berkontribusi secara bersamaan terhadap keandalan sistem:

- a. Keamanan (security), dimana akses (baik fisik maupun logis) terhadap sistem dan data di dalamnya dikendalikan serta terbatas untuk pengguna yang sah.

- b. Kerahasiaan (confidentiality), dimana informasi keorganisasian yang sensitive (seperti rencana pemasaran, rahasia dagang) terlindungi dari pengungkapan tanpa izin.
- c. Privasi (privacy), dimana informasi pribadi tentang pelanggan, pegawai, pemasok, atau rekan kerja hanya dikumpulkan, digunakan, diungkapkan, dikelola sesuai dengan kepatuhan terhadap kebijakan internal dan persyaratan peraturan eksternal serta terlindungi dari pengungkapan tanpa izin.
- d. Integritas Pemrosesan (processing integrity), dimana data diproses secara akurat, lengkap, tepat waktu dan hanya dengan otorisasi yang sesuai.
- e. Ketersediaan (availability), dimana sistem dan informasinya tersedia untuk memenuhi kewajiban operasional dan kontraktual.

## **BAB II**

# **ANALISIS SISTEM**

### **2.1 Analisis Sistem**

Analisis Sistem adalah sebuah istilah yang secara umum menjelaskan tentang tahapan awal pengembangan sistem.. Analisis sistem menjadi faktor penting keberhasilan dalam pengembangan perangkat lunak. Dengan melakukan analisis sistem dapat mengetahui permasalahan yang terjadi pada sistem yang lama. Berikut ini adalah pandangan para ahli tentang sistem analisis:

Aji Supriyanto mendefinisikan Analisis sistem adalah tahapan penelitian atas sistem yang telah ada dengan tujuan untuk merancang sistem yang baru atau diperbaharui (Supriyanto, 2007)

James dan Tommie mendefinisikan Analisis sistem adalah proses intelektual yang berbaur dengan pengumpulan fakta (Hall & Singleton, 2007)

Sri mulyani menjelaskan bahwa Analisis sistem adalah sebuah penelitian yang bertujuan untuk membantu pembuat keputusan dalam membuat keputusan, sehingga tindakan ataupun keputusan yang diambil sesuai dengan kondisi sebenarnya (Mulyani, 2016)

Gunadi dan Al Fattah mendefinisikan analisis sistem merupakan teknik pemecahan masalah yang membagi komponen dengan mempelajari seberapa bagus bagian-bagian komponen tersebut bekerja dan berinteraksi untuk mencapai tujuan (Gunadi & Al Fatta, 2012)

Sehingga dapat ditarik kesimpulan bahwa analisis sistem merupakan tahapan penelitian dalam pengembangan sistem yang

bertujuan untuk mengetahui segala permasalahan yang terjadi dan memberikan kemudahan dalam menjalankan tahap selanjutnya yaitu perancangan sistem.

Menurut Aji Supriyanto tahapan – tahapan dalam melakukan analisis sistem sebagai berikut (Supriyanto, 2007) :

a. Identifikasi masalah

Cara melakukan identifikasi masalah yang paling tepat adalah dengan melakukan penelitian atau mengumumkan hasil penelitian yang dilakukan pada tahap perencanaan sistem. Rincian hasilnya berupa identifikasi penyebab masalah, identifikasi titik keputusan, dan identifikasi personel-personel yang terlibat.

b. Mengorganisasikan tim proyek

Menyusun tim proyek yang terlibat termasuk pemakai sistem yang nantinya digunakan pada kegiatannya.

c. Mendefinisikan kebutuhan informasi

Analisis sistem mempelajari kebutuhan informasi pemakai dengan terlibat dalam berbagai pengumuman informasi. Caranya yaitu melakukan wawancara, pengamatan, pencarian catatan, dan survei.

Sedangkan menurut jogiyanto hartono dalam tahap analisis sistem terdapat langkah - langkah dasar yang harus dilakukan oleh analisis sistem sebagai berikut:

a. Identify, yaitu mengidentifikasi masalah.

b. Understand, yaitu memahami kerja dari sistem yang ada.

c. Analyze, yaitu menganalisis sistem.

d. Report, yaitu membuat laporan hasil analisis.

Hasil dari analisis sistem meliputi pengertian yang jelas atas kebutuhan dan masalah - masalah yang dihadapi, jawaban pemecahan masalah dan pemenuhan kebutuhan diatas serta usul penyelesaian yang jelas (Sutabri, 2012). Adapun tujuan dari melakukan analisis terhadap sistem antara lain:

- a. Mendefinisikan masalah secara tepat.
- b. Menyusun beberapa alternatif solusi.
- c. Memilih dan mempertimbangkan satu dari beberapa alternatif solusi.
- d. Menyusun spesifikasi logis.
- e. Menyusun persyaratan fisik.
- f. Menyusun anggaran untuk dua fase penyusunan sistem selanjutnya, yaitu desain dan implementasi sistem.

Analisis sistem dipandang dari kebutuhannya dibagi menjadi 2 (dua) yaitu Analisis Kebutuhan Non-Fungsional dan Analisis kebutuhan fungsional.

### **2.1.1 Analisis kebutuhan non fungsional**

Analisis Kebutuhan non fungsional merupakan kebutuhan sistem yang menitikberatkan pada properti perilaku yang dimiliki oleh sistem, diantaranya kebutuhan perangkat lunak, perangkat keras, serta pengguna sistem (user) sebagai bahan analisis kekurangan dan kebutuhan yang harus dipenuhi dalam perancangan sistem yang akan diterapkan. Dalam hal ini analisis kebutuhan non fungsional dibagi menjadi 3 (tiga) macam yaitu :

- a. Analisis Kebutuhan Perangkat Lunak  
Analisis kebutuhan perangkat lunak adalah suatu kegiatan yang menentukan kebutuhan-kebutuhan atau kondisi yang harus dipenuhi untuk suatu produk baru atau yang akan diganti, dengan mempertimbangkan kemungkinan terjadinya konflik kebutuhan dari berbagai macam *stakeholder*. Secara umum dalam analisis perangkat lunak im dalam pengembangan dan pengimplemntasian sistem perlu diperhatikan spesifikasi minimal perangkat lunak yang digunakan.
- b. Analisis Kebutuhan Perangkat Keras  
Analisis Perangkat keras merupakan salah satu faktor yang

penting dalam pengembangan perangkat lunak, karena tanpa hardware yang memenuhi syarat, perangkat lunak yang akan dikembangkan tidak akan dapat berjalan. Analisis kebutuhan perangkat keras ini dibagi menjadi dua bagian yaitu analisis kebutuhan perangkat keras pembangunan aplikasi dan analisis kebutuhan perangkat keras pengguna.

c. Analisis Pengguna Sistem (User)

Analisis pengguna sistem dimaksudkan untuk mengetahui siapa saja aktor yang terlibat dalam menjalankan sistem. Selain itu Analisis pengguna sistem juga digunakan untuk membedakan pengguna berdasarkan hak aksesnya.

### 2.1.2 Analisis kebutuhan fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik serta sesuai dengan kebutuhan. Analisis kebutuhan fungsional ini meliputi analisis kebutuhan data dan pemodelan sistem.

a. Analisis Kebutuhan Sistem

Sebelum melakukan perancangan sistem, terlebih dahulu dilakukan analisis kebutuhan sistem yaitu dengan cara melakukan wawancara terhadap responden. Hal ini dimaksudkan agar dapat mengatasi ketidaksesuaian antara aplikasi yang dirancang dengan kebutuhan pengguna.

b. Analisis Kebutuhan Data

Analisis kebutuhan data bertujuan untuk memudahkan dalam perancangan sistem. Dalam analisis data ini peneliti mengumpulkan dan menganalisis data yang diperoleh untuk selanjutnya digunakan dalam perancangan perangkat lunak.

c. Pemodelan Sistem

Pemodelan Sistem merupakan suatu kegiatan penyederhanaan

dari sebuah elemen maupun komponen yang sangat kompleks untuk memudahkan pemahaman dari informasi yang dibutuhkan.

## **2.2 Desain Sistem**

Dalam pengembangan suatu perangkat lunak atau sistem, pastinya tidak lepas dengan tahapan desain sistem atau sering disebut perancangan. Perancangan diperlukan untuk memberikan gambaran yang jelas dan rancang bangun yang lengkap sebagai pedoman bagi pemrogram dalam mengembangkan perangkat lunak guna menghasilkan suatu rancangan sistem yang baik. Setelah membahas tentang tahapan analisis sistem selanjutnya secara prosedur kebanyakan akan mengarah ke desain sistem. Berikut ini merupakan definisi desain sistem menurut ahli :

Hanif menjelaskan Desain sistem merupakan suatu teknik pemecahan masalah yang saling melengkapi yang merangkai kembali bagian - bagian komponen sehingga menjadi sistem yang lengkap (Al Fattah, 2007).

Jogiyanto menjelaskan bahwa desain sistem merupakan penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah dari suatu kesatuan yang utuh dan berfungsi (Hartono M, 2005).

Desain sistem adalah tahapan berupa penggambaran, perencanaan dan pembuatan dengan menyatukan beberapa elemen terpisah dalam satu kesatuan yang utuh untuk memperjelas bentuk sebuah sistem (Anggraeni & Irviani, 2017).

Mengutip dari pendapat para ahli tentang definisi Desain sistem tersebut maka secara garis besar desain sistem dapat diartikan bawah suatu teknik pemecahan masalah dengan cara melakukan penggambaran, perencanaan dan pembuatan sketsa dari beberapa elemen yang terpisah dalam satu kesatuan. Adapun tujuan utama dari melakukan desain sistem yaitu Untuk memenuhi kebutuhan

kepada pemakai sistem serta untuk memberikan gambaran yang jelas dan rancang bangun yang lengkap kepada pemrogram komputer (Programmer) dan user yang terlibat. Tahapan desain sistem dibagi menjadi 2 tahap, yaitu desain logis dan desain fisik.

### **2.2.1. Desain Logis**

Logical Design adalah bagian dari fase desain dalam pengembangan sistem dimana semua fitur-fitur fungsional dari sistem dipilih dari tahapan analisis dideskripsikan terpisah dari platform atau teknologi yang akan digunakan untuk mengimplementasikan sistem. Pemodelan logis dimulai dengan mengumpulkan kebutuhan bisnis dan mengkonversi kebutuhan tersebut ke dalam model. Model logis berkisar pada kebutuhan bisnis, bukan database, meskipun kebutuhan bisnis digunakan untuk menetapkan kebutuhan database. Pemodelan logis melibatkan pengumpulan informasi tentang proses bisnis, badan usaha (kategori data), dan unit organisasi.

Hasil dari tahapan desain logis ini adalah deskripsi fungsional mengenai data dan proses yang ada dalam sistem baru serta deskripsi yang detail dari spesifikasi sistem meliputi: 1) Input (data apa saja yang menjadi input), 2) Output (informasi apa saja yang menjadi output), dan 3) Process (prosedur apa saja yang harus dieksekusi untuk mengubah input menjadi output). Tahapan desain logis biasanya menghasilkan beberapa dokumen diantaranya: dokumen model data, dokumen model proses, rancangan tabel, hirarki antar modul sampai ke desain antar muka dari sistem yang akan dibuat.

### **2.2.2. Desain Fisik**

Pada bagian ini spesifikasi logikal diubah ke dalam detail teknologi dimana pemrograman dan pengembangan sistem bisa diselesaikan. Pada tahapan inilah aktifitas coding dilakukan Adapun output dari sistem ini adalah penjelasan teknis, mengenai pilihan

teknologi software dan hardware yang digunakan dan penjelasan yang detail dari spesifikasi sistem meliputi: 1) modul-modul program, 2) file-file, 3) sistem Jaringan, 4) sistem software.

## **2.3 Pendekatan Analisis Sistem**

Secara umum analisis sistem adalah mengenai pemecahan masalah. Ada banyak cara dalam pendekatan analisis sistem, berikut ini merupakan macam-macam pendekatan analisis sistem antara lain:

### **2.3.1. Analisis Model-Driven**

Pendekatan pemecahan masalah yang memfokuskan pada pembuatan model-model sistem dengan gambar guna mendokumentasikan dan memvalidasikan sistem-sistem yang ada atau sistem yang diusulkan. biasanya dalam analisis model – driven ini menggunakan flowchart dan lain sebagainya untuk proses penggambaran. Berikut ini merupakan model pendekatan analisis driven :

- a. Analisis Terstruktur Analisis Terstruktur  
Sebuah teknik analisis yang berfokus pada proses yang digunakan untuk menganalisis sistem ataupun mendefinisikan persyaratan-persyaratan bisnis untuk pengembangan sistem baru, ataupun menggunakan kedua metode tersebut.
- b. Information engineering (IE)  
Sebuah teknik analisis yang berputar pada data akan tetapi sensitif pada proses. Teknik ini digunakan untuk merencanakan, menganalisis dan mendesain system informasi. Model ini merupakan gambar-gambar yang mengilustrasikan dan mensikronkan data dan proses-proses sistem.
- c. Analisis Berorientasi Objek  
Sebuah teknik analisis yang mewadahi data (properti) yang menjelaskan tentang orang, objek, tempat, kejadian, atau sesuatu

yang berlainan, dengan semua proses yang diizinkan untuk menggunakan atau memperbarui data dan properti-properti tersebut. Satu-satunya cara untuk mengakses atau memperbarui data objek adalah menggunakan proses-proses objek yang didefinisikan sebelumnya. Analisis berorientasi objek (OOA) adalah sebuah teknik model-driven yang mengintegrasikan data dan proses ke dalam konstruksi yang disebut objek. Model OOA adalah gambar-gambar yang mengilustrasikan objek-objek sistem dari berbagai macam perspektif, seperti struktur, kelakuan, dan interaksi objek-objek.

### **2.3.2. Analisis Sistem Terakselerasi (Accelerated Systems Analysis)**

Pendekatan yang menekankan konstruksi model sistem yang berfungsi/bekerja dalam usaha untuk mengakselerasi analisis sistem. Pendekatan ini menekankan pada pembangunan prototype untuk lebih mempercepat pengidentifikasian kebutuhan bisnis dan pengguna (user) dari sebuah sistem baru. Berikut ini merupakan macam-macam pendekatan analisis sistem terakselerasi:

- a. **Discovery Prototyping** Discovery prototyping  
Sebuah teknik yang digunakan untuk mengidentifikasikan persyaratan-persyaratan bisnis pengguna dengan membuat para pengguna bereaksi pada implementasi quick-and-dirty (bijaksana dan efektif tapi tanpa cacat atau efek samping yang tidak diinginkan) persyaratan-persyaratan tersebut.
- b. **Rapid Architected Development**  
Sebuah pendekatan yang mencoba memperoleh model-model sistem dari sistem-sistem yang ada atau prototipe-prototipe. Discovery Reverse Engineering adalah penggunaan teknologi yang membaca kode-kode program untuk database, program aplikasi, dan/atau antarmuka pengguna yang ada dan secara otomatis menghasilkan model sistem yang ekuivalen.

### 2.3.3. Metode Penemuan Persyaratan

Penemuan persyaratan/requirements discovery adalah proses digunakan oleh para analisis sistem, identifikasi atau ekstraksi masalah-masalah sistem dan persyaratan-persyaratan solusi dari komunitas pengguna. Berikut ini merupakan teknik – teknik analisis dalam metode penemuan persyaratan :

- a. Teknik Penemuan Fakta/Fact Finding Fact Finding.  
Proses pengumpulan informasi mengenai masalah, kesempatan, persyaratan solusi, dan prioritas sistem. Istilah ini disebut juga pengumpulan informasi (information gathering).
- b. Perencanaan Persyaratan Gabungan/Joint Requirement Planning (JRP).

Penggunaan seminar-seminar yang terfasilitasi untuk mengumpulkan para pemilik, pengguna, analis, beberapa desainer dan pembangun sistem untuk bersama-sama melakukan analisis sistem. JRP umumnya dianggap sebagai bagian dari metode lebih besar yang disebut joint application development (JAD), pengembangan aplikasi gabungan, sebuah penerapan yang lebih komprehensif dan teknikteknik JRP pada keseluruhan proses pengembangan sistem.

### 2.3.4. Metode Desain Ulang Proses Bisnis

Teknik analisa dan desain ulang alur kerja (proses bisnis) baik itu didalam atau antara perusahaan untuk mengoptimalkan proses *end-to-end* dan mengotomatisasi tugas-tugas yang tidak memiliki nilai tambah (*non value added tasks*). Desain ulang proses bisnis adalah penerapan metode-metode analisis sistem pada tujuan secara dramatis mengubah dan memperbaiki proses-proses bisnis mendasar sebuah organisasi terpisah dari teknologi informasi.

### **2.3.5. Strategi Analisis Sistem Fast**

Agile method/metode cerdas adalah integrasi berbagai macam pendekatan analisis dan desain sistem untuk aplikasi yang dianggap tepat untuk masalah yang sedang dipecahkan dan sistem yang sedang dikembangkan. FAST (Framework for the Application of System Thinking) adalah kerangka cerdas fleksibel untuk menyediakan tipe-tipe berbeda tipe proyek dan strategi. FAST tidak seperti metodologi lainnya karena tidak bersifat preskriptif atau tidak bergantung pada ketentuan resmi yang berlaku. Dalam FAST juga hal pertama yang harus dilakukan adalah melakukan pendekatan pemecahan masalah seperti memahami dan mengidentifikasi dahulu masalah seperti apa yang dihadapi dan menentukan solusinya. Terdapat beberapa fase dalam metode FAST, yaitu: 1. Definisi Lingkup (Scope Definition Phase), 2. Analisis Masalah (Problem Analysis Phase), 3. Analisis Persyaratan (Requirements Analysis Phase), 4. Desain Logis (Logical Design Phase), 5. Analisis Keputusan (Decision Analysis Phase), 6. Desain Fisik dan Integrasi (Physical Design & Integration), 7. Konstruksi dan Pengujian (Construction & Testing), 8. Instalasi dan Pengiriman (Installation & Delivery)

# **BAB III**

## **PENDEKATAN**

### **PENGEMBANGAN SISTEM**

Pendekatan pengembangan sistem dapat diartikan Suatu proses yang dengan kebutuhan diidentifikasi, problem dipilih, syarat –syarat pemecahan problem diidentifikasi, pemecahan dipilih dari beberapa alternatif, metode dan alat dicari dan diterapkan, hasil dievaluasi, dan revisi yang diperlukan terhadap seluruh bagian dari sistem tersebut dilaksanakan sedemikian rupa sehingga kebutuhan tersebut dapat tercapai. Berikut ini beberapa macam pendekatan pengembangan sistem berdasarkan sudut pandang:

#### **3.1 Pandangan Metodologi**

Ditinjau dari segi metodologi, pendekatan pengembangan sistem dibagi menjadi 2 (dua) yaitu pendekatan klasik atau konvensional dan pendekatan terstruktur. Dimana pendekatan klasik dalam pengembangan suatu sistem dengan mengikuti tahapan – tahapan pada metode yang digunakan, sedangkan pendekatan terstruktur menyediakan sistem tambahan berupa alat-alat dan teknik - teknik untuk mengembangkan sistem disamping tetap mengikuti ide dari metode yang digunakan. Berikut ini penjelasan lebih lanjut tentang pendekatan klasik dan pendekatan terstruktur.

##### **3.1.1 Pendekatan Klasik**

Pendekatan klasik disebut juga pendekatan tradisional atau pendekatan konvensional adalah pendekatan dalam pengembangan

sistem yang mengikuti tahapan-tahapan pengembangan sistem tanpa dibekali dengan alat-alat dan teknik-teknik yang memadai.

Pendekatan klasik tidak cukup digunakan untuk mengembangkan sistem informasi yang kini semakin kompleks, dan dapat menimbulkan permasalahan, seperti: 1) Pengembangan perangkat lunak menjadi sulit, 2) Biaya perawatan atau pemeliharaan sistem menjadi lebih mahal, 3) Kemungkinan kesalahan sistem besar, 4) Keberhasilan sistem kurang terjamin, 5) Masalah dalam penerapan sistem.

### **3.1.2 Pendekatan Terstruktur**

Karena terjadi banyak permasalahan pada pendekatan klasik, maka dibutuhkan pendekatan pengembangan sistem yang lebih baik yang tidak hanya mengikuti tahapan pengembangan sistem namun juga dilengkapi dengan beberapa alat dan teknik. Pendekatan Terstruktur dilengkapi dengan alat-alat (tools) dan teknik - teknik (techniques) yang dibutuhkan dalam pengembangan sistem sehingga didapatkan hasil akhir berupa sistem yang strukturnya didefinisikan dengan baik dan jelas. Alat tersebut meliputi : diagram arus data (data flow diagram), kamus data (data dictionary), tabel keputusan (decision table), diagram HIPO (HIPO diagram), dan bagan Terstruktur (structured chart). Permasalahan yang kompleks dipecah menjadi modul-modul terstruktur dan terarah, fleksibel, dokumentasi yang baik, tepat waktu, sesuai rencana dan biaya, produktifitas, kualitas sistem baik, dan melibatkan pemakai sistem.

### **3.2 Pandangan Sasaran ( Sepotong dan Sistem )**

Berdasarkan sudut pandang sasaran dalam pendekatan pengembangan sistem dibagi menjadi Pendekatan sepotong (*Piecemeal Approach*) dan Pendekatan Sistem (*System Approach*).

### **3.2.1 Pendekatan Sepotong (*Piecemeal Approach*)**

Pendekatan pengembangan sistem yang menekankan pada suatu kegiatan atau aplikasi saja. Kegiatan atau aplikasi yang dipilih tersebut kemudian dikembangkan tanpa memperhatikan posisinya di sistem informasi atau tanpa memperhatikan sasaran organisasi secara keseluruhan.

### **3.2.2 Pendekatan Sistem (*System Approach*)**

Pendekatan Pengembangan sistem yang memperhatikan sistem informasi sebagai satu kesatuan terintegrasi dari masing-masing kegiatan atau aplikasinya dan menekankan pada pencapaian sasaran keseluruhan.

## **3.3 Penentuan Kebutuhan ( Bottom-Up dan Top-Down )**

Pendekatan pengembangan sistem dari sudut pandang kebutuhan di bagi menjadi pendekatan bottom – up dan pendekatan top – bottom. Berikut ini penjelasan dari kedua pendekatan tersebut.

### **3.3.1 Pendekatan Bawah ke Atas (*bottom-up approach*)**

Pendekatan bawah ke atas (*bottom-up approach*) dimulai dari level bawah organisasi, yaitu level operasional dimana transaksi dilakukan. Pendekatan ini dimulai dari perumusan kebutuhan-kebutuhan untuk menangani transaksi dan naik ke level atas dengan merumuskan kebutuhan informasi berdasarkan transaksi tersebut.

Pendekatan ini juga merupakan ciri-ciri dari pendekatan klasik. Pendekatan bawah-naik bila digunakan pada tahap analisis sistem disebut juga dengan istilah data analysis, karena yang menjadi tekanan adalah data yang akan diolah terlebih dahulu, informasi yang akan dihasilkan menyusul mengikuti datanya.

### **3.3.2 Pendekatan Atas ke Bawah (*top-bottom approach*)**

Sedangkan pendekatan atas ke bawah (*top – down*) sebaliknya dimulai dari level atas organisasi yaitu level perencanaan strategis. Pendekatan ini dimulai dengan mendefinisikan sasaran dan kebijakan organisasi. Selanjutnya, dilakukan analisis kebutuhan informasi kemudian ke penentuan input, output, basis data, prosedur-prosedur operasi, dan kontrol. Pendekatan ini merupakan ciri-ciri dari pendekatan terstruktur. Jika pendekatan ini digunakan pada tahap analisis, disebut dengan *decision analysis*, karena yang menjadi fokus adalah informasi yang dibutuhkan untuk pengambilan keputusan oleh manajemen terlebih dahulu.

## **3.4 Teknik Pengembangan ( Menyeluruh dan Moduler )**

Berdasarkan teknik pengembangannya, pendekatan pengembangan sistem dibagi menjadi pendekatan menyeluruh dan pendekatan moduler.

### **3.4.1 Pendekatan Menyeluruh (*Total-system approach*)**

pendekatan pengembangan sistem serentak secara menyeluruh. Pendekatan ini sulit dilakukan untuk sistem yang kompleks, karena menjadi sulit untuk dikembangkan.

### **3.4.2 Pendekatan Moduler (*Moduler Approach*)**

Pendekatan Pengembangan sistem dengan cara memecah sistem yang rumit menjadi beberapa bagian atau modul yang sederhana sehingga akan lebih mudah dipahami dan dikembangkan. Sistem juga akan dapat dikembangkan sesuai dengan waktu yang direncanakan, mudah dipahami oleh pemakai dan mudah untuk dipelihara.

### **3.5 Teknologi ( Lompatan Jauh dan Berkembang )**

#### **3.5.1 Pendekatan Lompatan Jauh (*Great Loop Approach*)**

Menerapkan perubahan menyeluruh secara serentak menggunakan teknologi canggih. Hal ini mengandung resiko karena teknologi komputer begitu cepat berkembang dan tahun-tahun mendatang sudah menjadi usang investasinya juga mahal dan terlalu kompleks.

#### **3.5.2 Pendekatan Berkembang (*Evolutionary Approach*)**

Menerapkan teknologi canggih hanya untuk aplikasi yang memerlukan saja saat itu dan akan terus dikembangkan untuk masamasa selanjutnya mengikuti kebutuhan dan sesuai dengan perkembangan teknologi yang ada.

### **3.6 Siklus Hidup Pengembangan Sistem ( SDLC )**

Dalam pengembangan perangkat lunak pastinya melewati beberapa tahapan pengembangan, mulai dari perencanaan sampai dengan implementasi. Bila perangkat lunak atau sistem tersebut sudah diimplementasikan pada perkembangannya timbul permasalahan terkait sistem tersebut, maka perlu dikembangkan untuk dapat mengatasi permasalahan baru tersebut. Siklus ini disebut dengan siklus hidup pengembangan sistem (system development life cycle/SDLC). System Development Life Cycle merupakan metodologi umum yang digunakan untuk mengembangkan sistem informasi (R. Susanto & Andriana, 2016).

Siklus hidup pengembangan sistem informasi merupakan suatu bentuk yang digunakan untuk menggambarkan tahapan utama dan langkah-langkah pada tahapan tersebut dalam proses pengembangan sistem (Hartono M, 2005).

Siklus hidup pengembangan sistem dimulai dari tahap perencanaan, tahap pengembangan (mulai dari investigasi, analisis, desain sampai dengan implementasi) dan tahap evaluasi. Pada tahap evaluasi ini akan dilakukan secara terus menerus untuk memastikan bahwa sistem yang diimplementasikan berjalan dengan lancar dan tidak terjadi permasalahan. Jika terjadi suatu permasalahan maka sistem informasi akan dikembangkan kembali atau diganti dengan sistem yang baru. Pengembangan sistem yang baru ataupun pengembangannya ini akan dimulai dari tahap perencanaan kembali. Berikut ini merupakan penjelasan dari siklus hidup pengembangan sistem :

a. Tahap Perencanaan

Perencanaan pengembangan sistem pada dasarnya bertujuan untuk mengetahui dan mengedepankan sistem yang akan dikembangkan. Target apa saja yang di dicapai, lama pengerjaan, serta mempertimbangkan kebutuhan dana yang digunakan dalam pengembangan sistem. Pada tahap perencanaan ini juga mendefinisikan tentang masalah bisnis yang dihadapi oleh organisasi, perihal apa saja yang kiranya menghambat laju kembangnya bisnis.

b. Tahap Analisis

Pada tahapan analisis ini mengumpulkan kebutuhan-kebutuhan bisnis apa saja dari berbagai stakeholders yang berkepentingan. Mendefinisikan kebutuhan-kebutuhan bisnis yang akan ditindaklanjuti guna diakomodir oleh sistem yang akan dibangun. Membuat prototype sederhana untuk memenuhi kebutuhan-kebutuhan tersebut. Melakukan prioritasasi terhadap kebutuhan-kebutuhan bisnis yang telah didefinisikan. Membuat dan mengevaluasi alternatif-alternatif solusi untuk masing-masing kebutuhan tersebut. Mengulas rekomendasi-rekomendasi solusi dengan pihak manajemen.

c. Design

Melakukan perancangan dengan mengintegrasikannya dengan jaringan. Merancang arsitektur aplikasi yang akan digunakan. Merancang tampilan layar untuk pengguna. Merancang system interfaces. Merancang dan mengintegrasikan sistem dengan database. Membuat prototypes untuk detail-detail perancangan. Membuat dan mengintegrasikan sistem dengan system control.

d. Implementation

Membangun komponen-komponen perangkat lunak. Melakukan verifikasi dan uji coba terhadap sistem yang telah selesai dibangun. Melakukan konversi data. Melatih para pengguna untuk berinteraksi serta menyelesaikan tugas kerjanya dengan menggunakan sistem. Membuat dokumentasi terhadap sistem yang telah selesai dibangun, yang dapat berupa manual book, etc. Meng-install sistem di terminal-terminal PC yang membutuhkan.

e. Activities

Melakukan pemeliharaan sistem dengan pengecekan secara berkala/periodik. Memperkaya atau mengembangkan sistem dengan penambahan fitur-fitur baru yang dapat meningkatkan kinerja kerja user guna mendukung kinerja bisnis. Memberikan pelayanan kepada para users, seperti dalam bentuk call center ataupun IT support.

## **BAB IV**

# **WATERFALL MODEL**

SDLC (Software Development Life Cycle, Siklus Hidup Pengembangan Sistem) atau Systems Life Cycle (Siklus Hidup Sistem), dalam rekayasa sistem dan rekayasa perangkat lunak, adalah proses pembuatan dan perubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. Konsep ini umumnya merujuk pada sistem komputer atau informasi. Salah satu metode dalam SDCL yang sering dijumpai adalah Waterfall. Secara konsep SDLC ini berkembang secara sistematis dari satu tahap ke tahap lain layaknya air terjun. Metode ini merupakan suatu metode dalam pengembangan software dimana pengerjaannya harus dilakukan secara berurutan yang dimulai dari tahap perencanaan konsep, pemodelan (design), implementasi, pengujian dan pemeliharaan

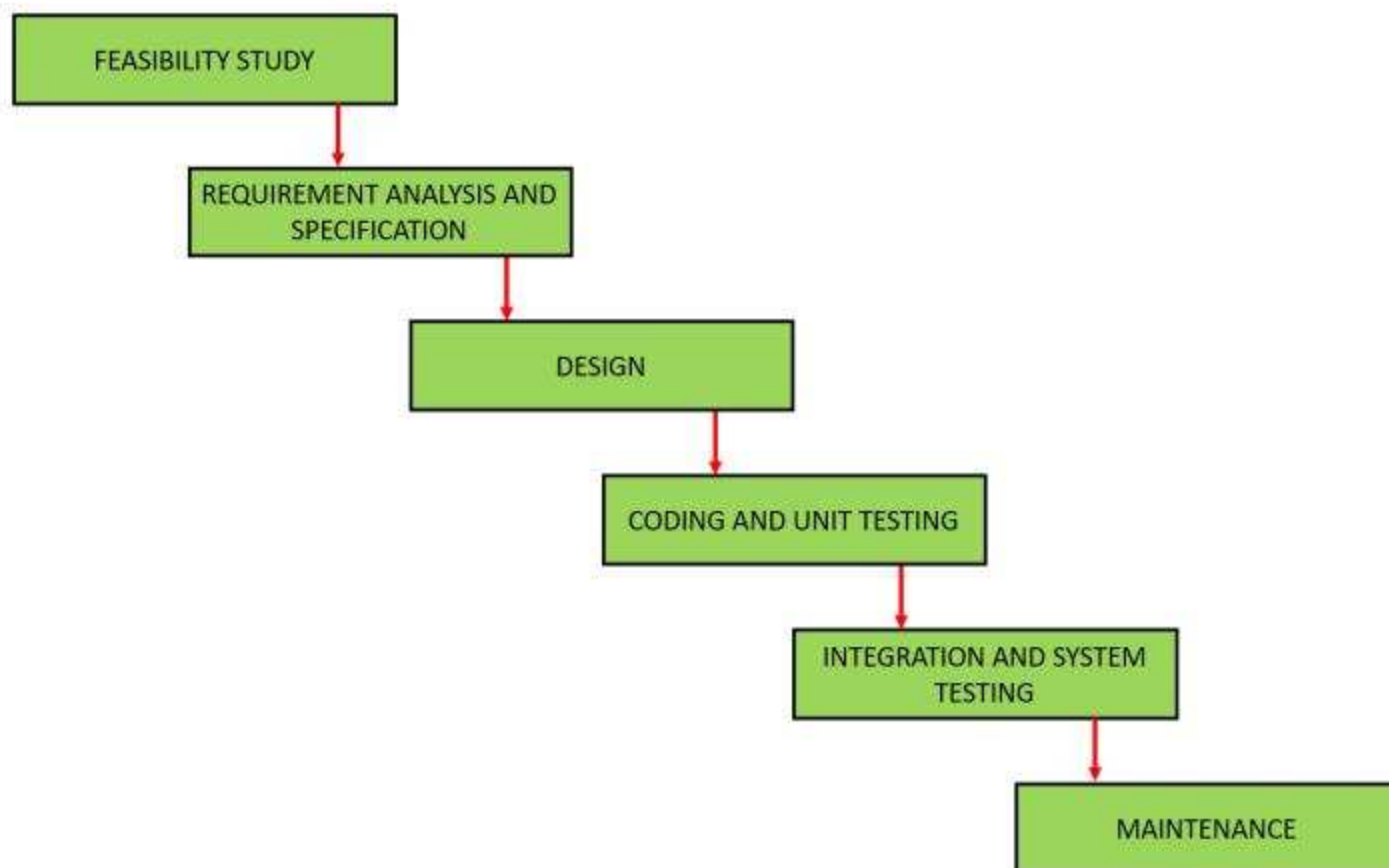
### **4.1 Classical Waterfall**

Classical waterfall adalah metode pengembangan perangkat lunak tertua yang bersifat sistematis dan berurutan. Nama model ini sebenarnya adalah “Linear Sequential Model”. Model ini sering disebut juga dengan “classic life cycle” atau metode waterfall . Model ini dapat digunakan pada saat kebutuhan untuk sebuah masalah telah dipahami dengan baik, dan pekerjaan dapat mengalir secara linear dari proses komunikasi hingga penyebaran (deployment).

Situasi ini ditemui saat adaptasi atau perpanjangan dari sistem yang ada sudah terdefinisi dengan baik. Adapun model ini juga dapat digunakan pada situasi dimana dibutuhkan usaha yang terbatas untuk pengembangan perangkat lunak, namun kebutuhan perangkat lunak

sudah terdefinisi dengan baik dan cenderung stabil. Namun, dalam pengembangan perangkat lunak, model ini cenderung menjadi salah satu pendekatan yang kurang iteratif dan fleksibel, karena proses mengalir satu arah (“ke bawah” seperti waterfall).

Pada model ini satu fase baru dapat dimulai setelah menyelesaikan fase sebelumnya. Artinya output satu fase akan menjadi masukan ke fase berikutnya. Dengan demikian proses pembangunan dapat dianggap sebagai aliran berurutan seperti air terjun (Waterfall). Dengan demikian, metode ini dianggap lebih cocok digunakan untuk proyek pembuatan sistem baru dan juga pengembangan software dengan tingkat resiko yang kecil serta waktu pengembangan yang cukup lama. Berikut ini Fase sekuensial yang berbeda dari model waterfall ini ditunjukkan pada gambar di bawah ini :



Gambar 4. 1 Classical Waterfall

a. Studi kelayakan (*Feasibility Study*) :

Tujuan utama dari tahap ini adalah menentukan apakah layak secara finansial dan teknis dalam pengembangan perangkat

lunak yang akan dikembangkan. Studi kelayakan ini melibatkan pemahaman terhadap permasalahan dan kemudian merancang berbagai strategi dalam menyelesaikan masalah.

b. Analisis dan spesifikasi kebutuhan (*Requirement Analysis and Specification*)

Tujuan dari tahap analisis dan spesifikasi kebutuhan adalah untuk memahami kebutuhan / permintaan yang tepat dari pelanggan dan mendokumentasikannya dengan benar. Pada tahapan ini biasanya pengumpulan data dengan mewawancarai langsung pengguna, pelanggan, dan pemangku kepentingan lainnya.

- 1) Pengumpulan dan analisis kebutuhan : Semua persyaratan terkait pengembangan perangkat lunak dikumpulkan dari pelanggan dan kemudian persyaratan yang dikumpulkan untuk dilakukan analisis. Tujuan dari bagian analisis adalah untuk menghilangkan ketidaklengkapan dan ketidakkonsistenan kebutuhan dalam pengembangan perangkat lunak.
- 2) Spesifikasi kebutuhan: Persyaratan yang telah dilakukan analisis ini didokumentasikan dalam dokumen spesifikasi kebutuhan pengembangan perangkat lunak (*software requirement specification*). Dokumen ini berfungsi sebagai kontrak antara tim pengembangan dan pelanggan.

c. Desain (*Design*)

Tujuan pada tahapan ini adalah untuk mengubah persyaratan yang ditentukan dalam dokumen SRS menjadi struktur yang sesuai untuk diimplementasikan dalam beberapa bahasa pemrograman. Pada tahap ini masalah dijabarkan lebih lanjut dengan menggambar sketsa dan diagram. ada dua pendekatan desain yang digunakan sering digunakan saat ini: pendekatan desain tradisional dan pendekatan desain berorientasi objek.

d. Coding and Unit Testing

Tujuan tahap Pengkodean dalam pengembangan perangkat lunak ini adalah untuk menerjemahkan desain perangkat lunak ke dalam bahasa pemrograman. Tahapan ini terkadang juga disebut tahap implementasi karena desain diimplementasikan menjadi solusi yang bisa diterapkan pada tahap ini. Setiap komponen yang desain diimplementasikan sebagai modul program. Produk akhir dari tahap ini adalah sekumpulan modul program yang telah diuji secara individu. Setelah selesai melakukan pengkodean, selanjutnya setiap modul diuji untuk menentukan apakah kinerja setia modul berjalan dengan lancar atau tidak.

e. Integration and System Testing

Integrasi berbagai modul dilakukan setelah proses penerjemahan ke dalam bahasa pemrograman (pengkodean) dan setiap unit modul diuji secara individu selesai dilakukan. Integrasi modul dilakukan secara bertahap melalui sejumlah langkah. Setelah semua modul berhasil diintegrasikan dan diuji selanjutnya dilakukan pengujian sistem. Pengujian sistem terdiri dari tiga jenis aktivitas pengujian, seperti yang dijelaskan di bawah ini:

- 1) Alpha testing : Pengujian ini lakukan oleh oleh tim pengembangan
- 2) Beta testing : Pengujian sistem dilakukan oleh sekumpulan pelanggan / pemakai.
- 3) Acceptance testing: Setelah perangkat lunak selesai diuji maka selanjutnya sistem akan diserahkan ke pemohon (pelanggan) untuk melakukan pengujian dalam pengujian ini pemohon (pelanggan) bisa melakukan apakah pemohon menerima perangkat lunak atau menolaknya.

f. Perawatan ( *Maintenance* )

Perawatan dilakukan ketika ditemukan kerusakan pada perangkat lunak atau perubahan atau peningkatan perangkat lunak diminta

oleh pelanggan. Pada tahapan ini merupakan tahapan terpenting dari siklus hidup perangkat lunak. Upaya yang dihabiskan untuk pemeliharaan adalah 60% dari total upaya yang dihabiskan dalam pengembangan perangkat lunak lengkap. Pada dasarnya ada tiga jenis perawatan:

- 1) **Corrective Maintenance** : Jenis pemeliharaan ini dilakukan untuk memperbaiki kesalahan yang tidak ditemukan selama fase pengembangan produk.
- 2) **Perfective Maintenance** : Jenis pemeliharaan ini dilakukan untuk meningkatkan fungsionalitas sistem berdasarkan permintaan pelanggan.
- 3) **Adaptive Maintenance** : Pemeliharaan adaptif biasanya diperlukan untuk mem-porting perangkat lunak agar dapat bekerja di lingkungan baru seperti bekerja pada platform komputer baru atau dengan sistem operasi baru.

#### **4.1.1 Keuntungan / Keunggulan Classical Waterfall**

Model Classical waterfall ini merupakan model yang idealis untuk pengembangan perangkat lunak. Ini sangat sederhana, sehingga dapat dianggap sebagai dasar untuk model siklus hidup pengembangan perangkat lunak lainnya. Berikut adalah beberapa keuntungan utama dari model ini :

- a. Model ini sangat sederhana dan mudah dimengerti dan Setiap tahap dalam model didefinisikan dengan jelas.
- b. Prosesnya lebih Terstruktur, hal ini membuat kualitas software baik dan tetap terjaga. Dari sisi user juga lebih menguntungkan, karena dapat merencanakan dan menyiapkan kebutuhan data dan proses yang diperlukan sejak awal.
- c. Penjadwalan pada model ini juga menjadi lebih menentu, karena jadwal setiap proses dapat ditentukan secara pasti. Sehingga dapat dilihat jelas target penyelesaian pengembangan program. Dengan

adanya urutan yang pasti, dapat dilihat pula perkembangan untuk setiap tahap secara pasti. Dari sisi lain,

- d. Model ini merupakan jenis model yang bersifat dokumen lengkap sehingga proses pemeliharaan dapat dilakukan dengan mudah
- e. Model ini bekerja dengan baik untuk proyek kecil dan proyek dengan persyaratan yang baik mengerti.

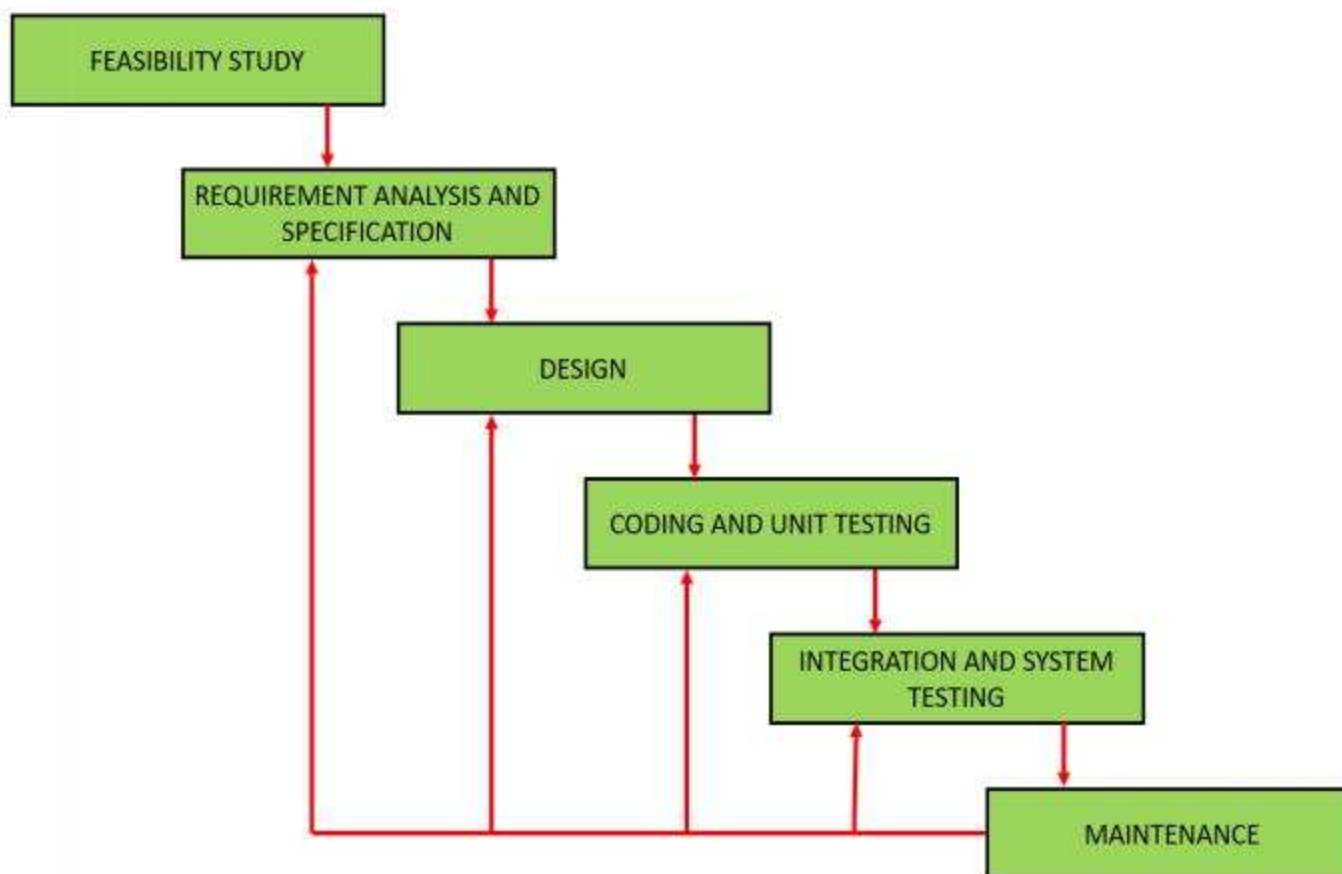
#### **4.1.2 Kekurangan / Kelemahan Classical Waterfall**

Model classical waterfall ini memiliki berbagai kekurangan dalam proses pengembangan suatu perangkat lunak. Berikut adalah beberapa kelemahan utama model ini:

- a. Bersifat kaku  
Jika terdapat kekurangan proses/prosedur dari tahap sebelumnya, maka tahapan pengembangan harus dilakukan mulai dari awal lagi. Hal ini akan memakan waktu yang lebih lama.
- b. Tidak ada jalur umpan balik  
Dalam pengembangan dari model classical waterfall ini mengasumsikan bahwa tidak ada kesalahan yang pernah dilakukan oleh pengembang selama fase apa pun. Oleh karena itu, ini tidak memasukkan mekanisme apa pun untuk koreksi kesalahan.
- c. Sulit untuk mengakomodasi permintaan perubahan  
Model ini mengasumsikan bahwa setiap tahapan berjalan dengan benar, tetapi sebenarnya persyaratan pelanggan terus berubah seiring waktu. Sulit untuk mengakomodasi permintaan perubahan apa pun setelah fase spesifikasi persyaratan selesai.
- d. Tidak ada fase yang tumpang tindih  
Model ini merekomendasikan bahwa fase baru dapat dimulai hanya setelah penyelesaian fase sebelumnya. Tetapi dalam proyek nyata, ini tidak dapat dipertahankan. Untuk meningkatkan efisiensi dan mengurangi biaya, ada beberapa tahapan yang harus dilakukan secara bersamaan.

## 4.2 Iterative Waterfall

Dalam proyek pengembangan perangkat lunak praktis, model Classical Waterfall ini sulit untuk diimplementasikan. Jadi, model Iterative waterfall dapat dianggap menggabungkan perubahan yang diperlukan pada model classical waterfall agar dapat digunakan dalam proyek pengembangan perangkat lunak. Pada dasarnya model Ini secara pengembangan hampir sama dengan model classical waterfall kecuali terdapat beberapa perubahan yang dilakukan untuk meningkatkan efisiensi dalam pengembangan perangkat lunak. Model Iterative waterfall memberikan jalur umpan balik / perubahan pada setiap fase ke fase sebelumnya, hal ini merupakan perbedaan utama dari model air terjun klasik. Berikut ini fase sekuensial yang berbeda dari model Iterative waterfall ini ditunjukkan pada gambar di bawah ini :



Gambar 4. 2 Model Iterative waterfall (PAL, 2018)

Pada model ini ketika terdapat kesalahan pada beberapa fase selanjutnya, jalur umpan balik dimungkinkan untuk melakukan

perbaikan pada fase tersebut. Dengan menggunakan model ini maka jalur umpan balik memungkinkan untuk dikerjakan secara ulang di mana kesalahan dilakukan. dan setelah dilakukan perubahan maka dapat berdampak pada fase selanjutnya. Namun, tidak ada jalur umpan balik ke tahap studi kelayakan karena sekali proyek telah diambil, maka proyek tidak diperbolehkan untuk dikembalikan. Mendeteksi kesalahan pada suatu fase yang sama saat terjadi adalah hal yang baik. hal Ini mengurangi usaha dan waktu yang dibutuhkan untuk memperbaiki kesalahan.

#### **4.2.1 Keunggulan Model Iterative waterfall**

Umpan balik: dalam model classical waterfall, tidak ada jalur umpan balik, jadi tidak ada mekanisme untuk koreksi kesalahan. Namun dalam jalur umpan balik model ini dari satu fase ke fase sebelumnya memungkinkan untuk melakukan koreksi kesalahan yang telah dilakukan.

Sederhana: Model Iterative waterfall sangat mudah dipahami dan digunakan. Itulah mengapa ini adalah salah satu model pengembangan perangkat lunak yang paling banyak digunakan.

#### **4.2.2 Kekurangan Model Iterative waterfall**

Sulit untuk memasukkan permintaan perubahan: kelemahan utama dari model Iterative waterfall adalah bahwa semua persyaratan harus dinyatakan dengan jelas sebelum memulai fase pengembangan. Pelanggan dapat mengubah persyaratan setelah beberapa waktu, tetapi model waterfall berulang tidak meninggalkan ruang lingkup apa pun untuk memasukkan permintaan perubahan yang dibuat setelah fase pengembangan dimulai.

Pengiriman inkremental tidak didukung: dalam model Iterative waterfall, perangkat lunak lengkap dikembangkan dan diuji sepenuhnya sebelum dikirimkan ke pelanggan. Tidak ada ruang

untuk pengiriman perantara. Sehingga pelanggan harus menunggu lama untuk mendapatkan software tersebut.

Tumpang tindih fase tidak didukung: model Iterative waterfall mengasumsikan bahwa satu fase dapat dimulai setelah menyelesaikan fase sebelumnya. Namun dalam proyek nyata, fase mungkin dilakukan secara bersamaan untuk mengurangi tenaga dan waktu yang dibutuhkan dalam penyelesaian suatu s.

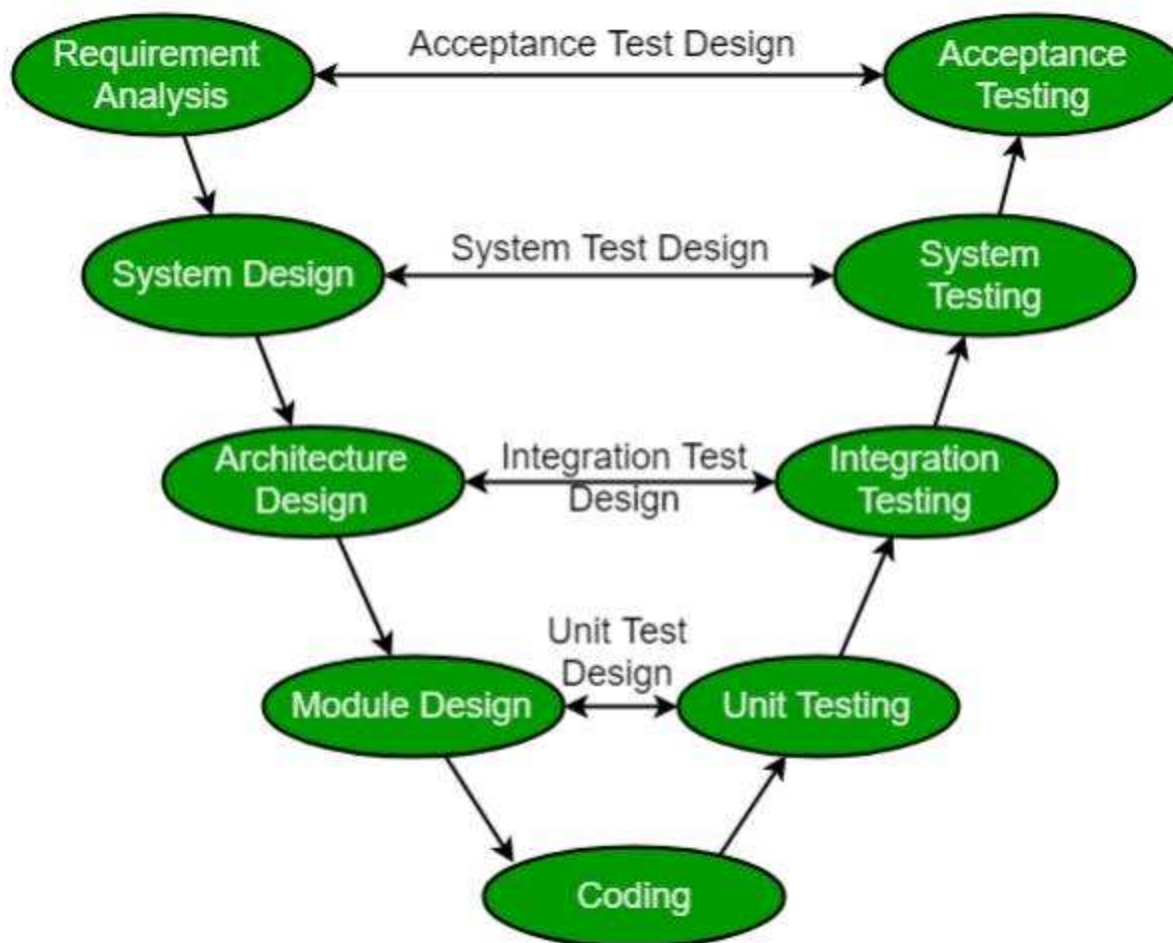
Interaksi pelanggan terbatas: interaksi pelanggan terjadi pada awal proyek pada saat pengumpulan persyaratan dan pada penyelesaian proyek pada saat pengiriman perangkat lunak. Interaksi yang lebih sedikit dengan pelanggan ini dapat menyebabkan banyak masalah karena perangkat lunak yang akhirnya dikembangkan mungkin berbeda dari kebutuhan pelanggan yang sebenarnya.

### 4.3 V-Model

Model V adalah salah satu model proses pengembangan lunak (juga berlaku untuk perangkat keras) yang merupakan variasi representasi dari model waterfall (Yadav, 2012). Model ini pada tahap-tahapnya mirip dengan yang terdapat dalam model waterfall. Jika dalam model waterfall proses dijalaukan secara linear, maka dalam V-model proses dilakukan bercabang. Pada model ini tahapan pengujian dirinci untuk masing-masing tahapan (Eka Y. R et al., 2013). Model V menggambarkan hubungan dari aksi jaminan kualitas (quality assurance) ke aksi yang berhubungan dengan komunikasi, pemodelan, dan aktivitas pembangunan awal. Ketika tim pengembang perangkat lunak bergerak ke sisi kiri V, kebutuhan dari masalah dasar disempurnakan menjadi representasi yang lebih rinci dan teknis dari masalah dan solusinya (Bucanac, 1999). Setelah kode dihasilkan tim bergerak ke sisi kanan V, yang pada dasarnya melakukan serangkaian tes (tindakan penjaminan kualitas) yang memvalidasi masing-masing model yang dibuat saat tim bergerak ke sisi kiri. Pada kenyataannya,

tidak ada perbedaan mendasar antara siklus hidup klasik (classic life cycle) dan model V.

Model V menyediakan cara memvisualisasikan bagaimana tindakan verifikasi dan validasi diterapkan pada pekerjaan teknik sebelumnya (Pressman, 2010). Model Ini juga dikenal sebagai model verifikasi dan validasi. Ini didasarkan pada asosiasi fase pengujian untuk setiap tahap pengembangan yang sesuai. Pengembangan setiap langkah terkait langsung dengan tahap pengujian. Fase berikutnya dimulai hanya setelah menyelesaikan fase sebelumnya yaitu untuk setiap aktivitas pengembangan, ada aktivitas pengujian yang sesuai dengannya. Model V berisi fase verifikasi di satu sisi fase validasi di sisi lain. Fase verifikasi dan validasi digabungkan dengan fase pengkodean dalam bentuk V. Berikut ini gambaran dari tahapan model V :



Gambar 4. 3 V-Model (Kumar, 2019)

#### 4.3.1 Fase Verifikasi pada V Model

Verifikasi disini melibatkan teknik analisis statis (tinjauan) yang dilakukan tanpa menjalankan kode. Hal Ini merupakan proses

evaluasi tahap pengembangan produk untuk menemukan apakah telah memenuhi persyaratan yang dibutuhkan.

a. **Requirement Analysis**

Dalam rekayasa sistem dan rekayasa perangkat lunak, analisis kebutuhan mencakup tugas-tugas yang digunakan untuk menentukan kebutuhan atau kondisi yang harus dipenuhi untuk produk atau proyek baru atau yang diubah, dengan mempertimbangkan kebutuhan yang mungkin bertentangan dari berbagai pemangku kepentingan, menganalisis, mendokumentasikan, memvalidasi, dan mengelola kebutuhan perangkat lunak atau sistem (Kotonya & Sommerville, 1998). Fase ini menghasilkan elisitasi kebutuhan sistem untuk mendapatkan kebutuhan sistem melalui komunikasi dengan customer, sistem user, dan pihak lain yang berhubungan pada sistem yang akan dikembangkan (Saputra, 2012).

b. **System Design**

Fase ini bertujuan untuk memahami dan merancang arsitektur sistem (Indah et al., 2018). Fase ini berisi desain sistem dan perangkat keras lengkap dan pengaturan komunikasi untuk mengembangkan perangkat lunak.

c. **Architectural Design**

Fase desain arsitektur komputer dan arsitektur perangkat lunak juga dapat disebut sebagai desain tingkat tinggi. Dasar dalam memilih arsitektur adalah bahwa ia harus menyadari semua yang terdiri dari daftar modul, fungsionalitas singkat dari masing-masing modul, hubungan antarmuka, dependensi, tabel basis data, diagram arsitektur, detail teknologi dll. Desain integration testing dilakukan dalam fase ini

d. **Module Design**

Fase ini adalah tempat komponen perangkat lunak yang sebenarnya dirancang . Pada fase ini juga dilakukan pembagian desain sistem

menjadi modul-modul yang lebih kecil (Indah et al., 2018). Tahap ini mendefinisikan logika aktual untuk masing-masing dan setiap komponen sistem. Diagram kelas dengan semua metode dan hubungan antar kelas berada di bawah LLD (*Low level design*). Desain *unit testing* atau *component testing* dibuat dalam fase ini. Desain atau perancangan sistem yang digambarkan melalui sequence diagram digunakan untuk menunjukkan komunikasi dinamis antar objek selama sebuah task dieksekusi. Proses tersebut akan menunjukkan bagaimana pesan yang dikirimkan antar objek untuk menyelesaikan sebuah task (Pressman, 2010), class diagram memberikan gambaran statis maupun struktural dari sebuah sistem yang didalamnya terdapat atribut, operasi dan hubungan antar class (Pressman, 2010).

#### 4.3.2 Fase Validasi pada V Model

Validasi Ini melibatkan teknik analisis dinamis (fungsional, non-fungsional), pengujian dilakukan dengan mengeksekusi kode. Validasi adalah proses untuk mengevaluasi perangkat lunak setelah selesainya tahap pengembangan untuk menentukan apakah perangkat lunak memenuhi harapan dan persyaratan pelanggan.

a. Unit Testing

Pada fase ini dilakukan pengujian unit yang dikembangkan selama fase desain modul. Pengujian Unit ini dilakukan untuk menghilangkan bug pada proses pengkodean. Pengujian ini melibatkan pemeriksaan bahwa setiap fitur yang ditentukan dalam desain komponen telah diimplementasikan dalam komponen. Pengecekan ini berfokus pada masing-masing komponen secara terpisah, memastikan bahwa komponen tersebut berfungsi dengan baik sebagai sebuah unit (Yadav, 2012).

b. Integration Testing

Setelah menyelesaikan pengujian unit Pengujian integrasi

dilakukan. Dalam pengujian integrasi, modul diintegrasikan dan sistem diuji. Pengujian integrasi dilakukan pada tahap desain Arsitektur. Tes ini memverifikasi komunikasi modul di antara mereka sendiri. Pengujian Ini membahas perakitan dan integrasi komponen untuk membentuk paket perangkat lunak lengkap. Pengujian ini menggunakan teknik *black box testing* untuk mengatasi masalah yang terkait dengan masalah verifikasi dan pembangunan program (Yadav, 2012).

c. System Testing

Pengujian sistem menguji aplikasi lengkap dengan fungsionalitas, ketergantungan, dan komunikasinya, menguji persyaratan fungsional dan non-fungsional dari aplikasi yang dikembangkan. *System testing* masih berfokus pada pengembang, meskipun pengembang spesialis yang dikenal sebagai penguji sistem (*tester*) biasanya dipekerjakan untuk melakukannya. Intinya *system testing* bukan tentang memeriksa bagian-bagian individu dari desain, tetapi tentang memeriksa sistem secara keseluruhan.

d. User Acceptance Testing (UAT)

Acceptance testing memeriksa sistem terhadap kebutuhan pengguna. Hal ini mirip dengan *system testing* bahwa seluruh sistem diperiksa tetapi perbedaan penting adalah perubahan fokusnya. *System testing* memeriksa bahwa sistem yang ditentukan telah diberikan, sedangkan *acceptance testing* memeriksa bahwa sistem memberikan apa yang diminta. Pelanggan dan bukan pengembang harus selalu melakukan *acceptance testing*. UAT memverifikasi bahwa sistem yang dikirim memenuhi persyaratan pengguna dan sistem siap digunakan di dunia nyata.

Menurut Kumar pengimplementasian model V dalam pengembangan perangkat lunak akan lebih baik ketika dimana persyaratan didefinisikan dengan jelas dan ditetapkan dan V-Model digunakan ketika sumber daya teknis yang cukup tersedia dengan

keahlian teknis (Kumar, 2019). Prinsip dasar dalam pengembangan perangkat lunak dengan menggunakan model V sebagai berikut :

a. Large to Small

Dalam V-Model, pengujian dilakukan dalam perspektif hierarki, Misalnya, persyaratan yang diidentifikasi oleh tim proyek, membuat desain tingkat tinggi, dan fase desain detail proyek. Saat masing-masing fase ini memenuhi persyaratan, mereka mendefinisikan menjadi lebih dan lebih halus dan rinci.

b. Data/Process Integrity

Prinsip ini menyatakan bahwa desain yang berhasil dari setiap proyek memerlukan penggabungan dan kohesi data dan proses. Elemen proses harus diidentifikasi di setiap persyaratan.

c. Scalability

Prinsip ini menyatakan bahwa konsep V-Model memiliki fleksibilitas untuk mengakomodasi setiap proyek TI terlepas dari ukuran, kompleksitas, atau durasinya.

d. Cross Referencing

Korelasi langsung antara persyaratan dan aktivitas pengujian yang sesuai dikenal sebagai referensi silang.

e. Tangible Documentation

Prinsip ini menyatakan bahwa setiap proyek perlu membuat dokumen. Dokumentasi ini diperlukan dan diterapkan oleh tim pengembangan proyek dan tim pendukung. Dokumentasi digunakan untuk memelihara aplikasi setelah tersedia di lingkungan produksi.

#### **4.3.3 Keuntungan dari V-Model**

Keuntungan besar penggunaan model V dalam pengembangan perangkat lunak yaitu sangat fleksibel. Ini mendukung penyatuan proyek serta penambahan dan penghapusan metode dan alat secara dinamis. Berikut ini merupakan keuntungan penggunaan model V

dalam pengembangan sistem (Mohammed et al., 2010) :

- a. Sederhana dan mudah digunakan.
- b. Setiap fase memiliki kiriman tertentu.
- c. Peluang keberhasilan yang lebih tinggi dibandingkan model air terjun karena pengembangan awal rencana pengujian selama siklus hidup.
- d. Bekerja dengan baik untuk proyek kecil dimana persyaratannya mudah dipahami.

#### **4.3.4 Kelemahan dari V-Model**

Kelemahan-kelemahan besar dari model V adalah model V berorientasi pada proyek siklus hidup dan hanya digunakan sekali selama proyek. Hal itu tidak mencakup seluruh organisasi. Berikut ini merupakan kelemahan model V antara lain :

- a. Aktifitas V-Model hanya difokuskan pada projectnya saja, bukan pada keseluruhan organisasi. V-Model adalah proses model yang hanya dikerjakan sekali selama project saja, bukan keseluruhan organisasi.
- b. Prosesnya hanya secara sementara. Ketika project selesai, jalannya proses model dihentikan. Tidak berlangsung untuk keseluruhan organisasi.
- c. Metode yang ditawarkan terbatas. Sehingga kita tidak memiliki cara pandang dari metode yang lain. Kita tidak memiliki kesempatan untuk mempertimbangkan jika ada tools lain yang lebih baik.
- d. Toolnya tidak selengkap yang dibicarakan. SDE (Software Development Environment). Tidak ada tools untuk hardware di V-Model. Tool yang dimaksud adalah software yang mendukung pengembangan atau pemeliharaan / modifikasi dari sistem IT.
- e. V-Model adalah model yang project oriented sehingga hanya bisa digunakan sekali dalam suatu proyek.

- f. V-Model terlalu fleksibel dalam arti ada beberapa activity dalam V-Model yang digambarkan terlalu abstrak sehingga tidak bisa diketahui dengan jelas apa yang termasuk dalam activity tersebut dan apa yang tidak.

#### **4.4 Prototyping**

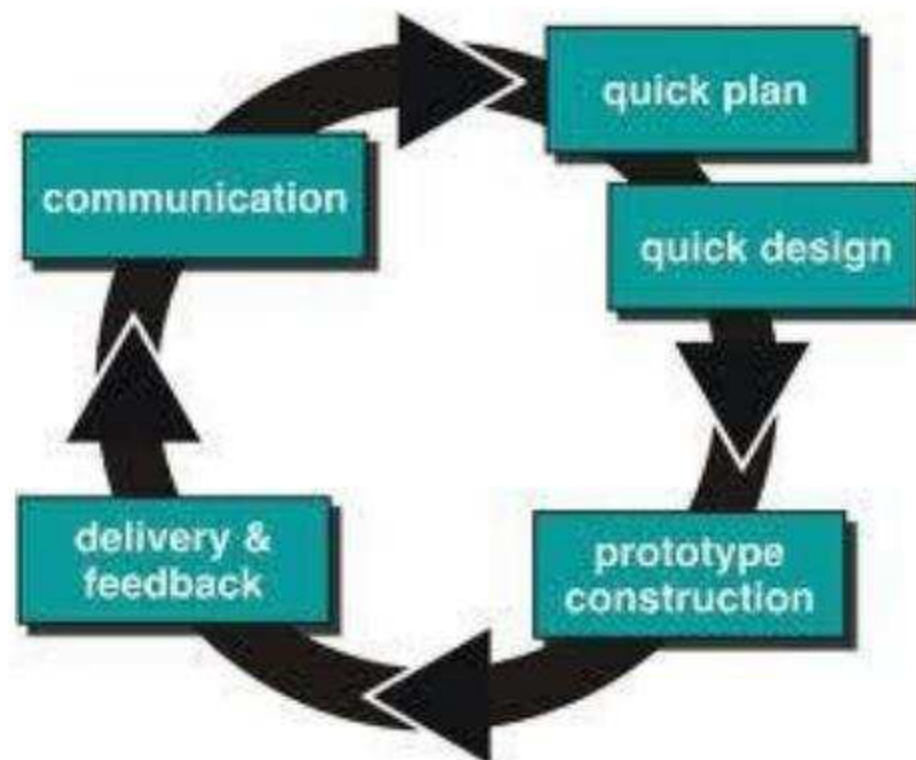
Prototyping merupakan suatu metode pengembangan sistem yang menggunakan pendekatan untuk membuat sesuatu program dengan cepat dan bertahap sehingga segera dapat dievaluasi oleh pemakai (Noor Santi, 2018). Metode ini adalah evolusi dalam dunia pengembangan atau pembuatan perangkat lunak, metode ini juga merevolusi metode pengembangan atau pembuatan perangkat lunak yang lama, yaitu sistem sekuensial yang biasa dikenal dengan nama Metode Waterfall. Proses pembuatan prototype ini disebut prototyping. Dasar pemikirannya adalah membuat prototype secepat mungkin bahkan dalam waktu semalam, lalu memperoleh umpan balik dari pengguna yang akan memungkinkan prototype tersebut diperbaiki kembali dengan sangat cepat.

Prototipe mewakili model produk yang akan dibangun atau mensimulasikan struktur, fungsionalitas dan operasi sistem. Dihuatnya sebuah Prototyping bagi pengembang sistem bertujuan untuk mengumpulkan informasi dari pengguna sehingga pengguna dapat berinteraksi dengan model prototype yang dikembangkan, sebab prototype menggambarkan versi awal dari sistem untuk kelanjutan sistem sesungguhnya yang lebih besar (Purnomo, 2017).

Ogedebe menegaskan: Telah ditemukan bahwa dalam analisis dan desain sistem, terutama untuk proses transaksi, di mana dialog yang ditampilkan lebih mudah difahami. Semakin besar interaksi antara komputer dan pengguna, besar pula manfaat yang diperoleh ketika proses pengembangan sistem informasi akan lebih cepat dan membuat pengguna akan lebih interaktif dalam proses pengembangannya (Ogedebe & Jacob, 2012).

Prototyping dapat diterapkan pada pengembangan sistem kecil maupun besar dengan harapan agar proses pengembangan dapat berjalan dengan baik, tertata serta dapat selesai tepat waktu. Keterlibatan pengguna secara penuh ketika prototype terbentuk akan menguntungkan seluruh pihak yang terlibat bagi pimpinan, pengguna sendiri serta pengembang sistem. Model ini menghasilkan prototype dari suatu perangkat lunak yang dapat digunakan sebagai perantara pengembang dengan pengguna untuk berinteraksi dalam pengembangan sistem informasi. Prototype adalah sebuah versi awal dari perangkat lunak yang digunakan untuk mendemonstrasikan konsep, mencoba berbagai pilihan desain, dan menggali lebih banyak permasalahan serta solusinya (Syarifudin & Ani, 2019). Berikut ini gambar tahapan – tahapan dari model prototyping :

Contoh”Gambar tahapan-tahapan dari model prototyping dapat dilihat pada Gambar ....”



Gambar 4. 4 Model Prototyping (Pressman & Maxim, 2014)

a. Communication

Proses pembuatan prototype ini disebut prototyping. Dasar pemikirannya adalah membuat prototype secepat mungkin,

bahkan dalam waktu semalam, lalu memperoleh umpan balik dari pengguna yang akan memungkinkan prototype tersebut diperbaiki kembali dengan sangat cepat. Tahap ini merupakan tahap awal sebelum melakukan pekerjaan yang bersifat teknis. Tahap ini penting bagi developer untuk berkomunikasi dan berkolaborasi dengan pelanggan atau perusahaan. Pada modul API dan Plugin, tahap communication dilakukan dengan menanyakan kepada tim yang mengerjakan proyek akhir yang sama dengan modul yang berbeda. Hal yang ditanyakan adalah service atau layanan apa saja yang dibutuhkan untuk mendukung aplikasi yang dibuat setiap modulnya.

b. Quick Plan

Tahap ini merupakan tahap perencanaan yang dilakukan terhadap aplikasi yang akan dibuat. Perencanaan ini dilakukan dengan mencari garis besar dari aplikasi. Sehingga proses pada tahap ini bisa dibilang sangat cepat. Perencanaan cepat ini akan berfokus pada penyajian dari aspek-aspek software yang akan terlihat oleh client.

c. Modelling Quick Design

Tahapan selanjutnya dari metode prototype adalah modeling quick plan, dari tahapan-tahapan sebelumnya menjelaskan permasalahan dan peralatan yang harus dibutuhkan, tahapan ini menjelaskan rancangan tentang perangkat lunak yang akan dibangun. Tahap ini bisa disebut dengan tahap pembuatan sketsa. Dimana semua diawali dengan pembuatan yang belum terlihat jelas atau spesifik seperti sketsa. Jika telah sesuai dengan karakteristik lainnya, sketsa mulai dibuat lebih detail. Seperti itu Rekayasa Perangkat Lunak. Pertama harus membuat suatu model agar dapat memahami kebutuhan perangkat lunak tersebut. Kemudian desain yang dibuat harus sesuai agar mencapai kebutuhan yang diminta.

d. Construction

Pada tahapan ini dilakukan pengkodean berdasarkan rancangan-rancangan yang telah dibuat. Pada tahap ini developer membuat coding (pembuatan code) baik manual atau otomatis. Jika telah selesai, maka pengujian harus langsung dilakukan untuk meminimalisir kesalahan-kesalahan dalam coding

e. Deployment, delivery & feedback

Pada tahapan ini, program yang telah dibuat dilakukan pengujian untuk menguji fungsionalitas dari sistem yang dibuat. Software atau aplikasi sudah dapat dikirimkan kepada pengguna. Selanjutnya, pengguna akan memberikan umpan balik atau feedback kepada aplikasi dalam melakukan evaluasi jika diperlukan.

#### 4.4.1 Kelebihan Metode Prototyping

Kelebihan metode prototyping yang paling utama adalah merupakan salah satu jenis metode pengembangan sistem yang sifatnya sangat cepat dan dapat menghemat waktu. Berbeda dengan pengembangan sistem menggunakan metode waterfall yang membutuhkan banyak biaya dan memakan waktu. Maka bagi user yang membutuhkan sebuah sistem dalam jangka waktu yang sangat singkat, bisa mengandalkan metode pengembangan sistem prototyping ini. Selain itu, metode prototyping juga memiliki beberapa kelebihan lainnya, seperti (Mubarok & Hadijah, 2015):

- a. Dapat menjalin komunikasi yang baik antar user dan pengembang sistem.
- b. Setiap perbaikan yang dilakukan pada prototype merupakan hasil masukan dari user yang akan menggunakan sistem tersebut, sehingga lebih reliabel.
- c. User akan memberikan masukan terhadap sistem sesuai dengan kemauannya.
- d. Menghemat waktu dalam mengembangkan sebuah system.

- e. Menghemat biaya, terutama pada bagian analisa, karena hanya mencatat poin – point penting saja.
- f. Cocok digunakan pada sebuah sistem kecil, yang digunakan pada ruang lingkup tertentu, seperti sistem di dalam sebuah kantor.
- g. Penerapan dari sistem yang menjadi lebih mudah untuk dilakukan.
- h. Prototype melibatkan user dalam analisa dan desain.
- i. Punya kemampuan menangkap requirement secara konkret daripada secara abstrak.
- j. Untuk digunakan secara standalone .
- k. Digunakan untuk memperluas SDLC.
- l. Mempersingkat waktu pengembangan sistem informasi.

#### **4.4.2 Kelemahan dari Metode Prototyping**

Beberapa kelemahan dan juga kekurangan dari metode prototyping antara lain:

- a. Untuk menghemat waktu, biasanya pengembangannya menggunakan bahasa pemrograman sederhana, yang mungkin rentan dari segi keamanannya tidak cocok untuk diimplementasikan pada sebuah sistem yang sangat besar dan global, seperti sistem operasi komputer.
- b. Kualitas aplikasi belum teruji dan belum mencantumkan pemeliharaan jangka panjang
- c. Algoritma dan bahasa yang digunakan sederhana karena ingin menyelesaikan proyek secara cepat padahal program merupakan blue print sistem.
- d. Teknik perancangan tidak baik dilihat dari hubungan pelanggan dengan komputer

#### **4.5 Incremental Development**

Incremental model adalah model pengembangan sistem pada software development berdasarkan requirement software yang dipecah

menjadi beberapa fungsi atau bagian sehingga model pengembangannya secara bertahap. Model ini menggabungkan elemen-elemen model waterfall dengan filosofi iteratif dari prototyping (Pressman, 2010).

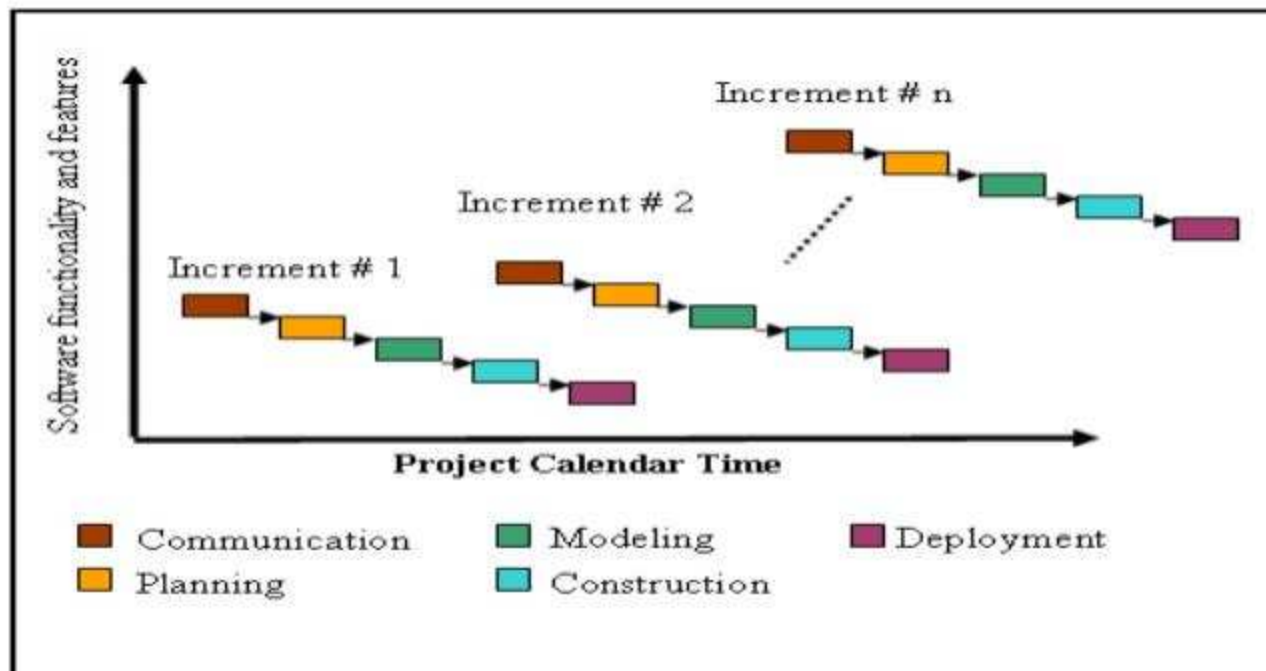
Ada banyak situasi dimana kebutuhan perangkat lunak awal didefinisikan dengan cukup baik, tetapi ruang lingkup keseluruhan dari upaya pengembangan menghalangi proses yang murni linier. Dengan menggunakan incremental model dapat membantu kita untuk mengurangi biaya sebelum mencapai level dari initial productivity dan mengakselerasi proses dari pembuatan suatu fungsi sistem.

Model Incremental Process memakai urutan-urutan linear yang berulang dalam membangun suatu perangkat lunak. Seiring berjalan waktu pengerjaan, setiap urutan linear akan menghasilkan perkembangan dalam pengerjaan perangkat lunak yang kemudian dapat digunakan oleh pengguna (Pressman, 2010).

Pada model incremental yang pertama sering disebut sebagai core product. Core product adalah dasar kebutuhan yang diperlukan oleh pengguna, terkadang banyaknya tambahan fitur yang diperlukan dapat menyebabkan tidak semuanya dapat tersampaikan. Oleh karena itu, hasil evaluasi dari core product dapat dijadikan sebagai rencana perkembangan untuk incremental selanjutnya dengan cara memodifikasi core product agar menjadi lebih baik untuk memenuhi kebutuhan pengguna (fitur dan fungsi). Proses ini dilakukan berulang hingga menghasilkan produk yang lengkap.

Sebagai contoh, pembentukan perangkat lunak word processing dengan menggunakan model incremental process. Pada incremental pertama hanya memberikan fungsi inti (basic file management, editing, dan document production function), pada incremental kedua memberikan tambahan agar menjadi lebih baik (advance editing dan document production capabilities) dan pada incremental ketiga memberikan tambahan selanjutnya (spelling dan grammar checking) proses ini dilakukan berulang hingga menghasilkan produk yang

lengkap. Berikut ini merupakan tahapan – tahapan dari model incremental (Larman & Basili, 2003) :



Gambar 4. 5 Tahapan Model Incremental

a. Communication

Sebelum kebutuhan dapat dianalisis, dimodelkan, atau dispesifikasikan, kebutuhan tersebut harus dikumpulkan melalui aktivitas communication (disebut juga pemancing kebutuhan).

b. Planning

Aktivitas communication membantu mendefinisikan gol keseluruhan dan sasaran. Pada aktivitas planning mencakup himpunan dari management dan technical practices yang memungkinkan untuk mendefinisikan road map sebagaimana perjalanannya terhadap gol strategi dan sasaran taktikal. Planning memiliki beberapa filosofi diantaranya “minimalis”, berargumen bahwa perubahan selalu menyingkirkan kebutuhan rencana yang detail. Yang lainnya adalah “tradisional”, berargumen bahwa rencana menyediakan sebuah peta jalan yang efektif, dan lebih detail. Yang lainnya lagi adalah “agilist,” berargumen bahwa sebuah “planning game” cepat mungkin perlu, tetapi suatu peta

jalan akan digabungkan sebagai “real work” pada permulaan perangkat lunak.

c. Modeling

Dalam rekayasa perangkat lunak, ada dua kelas dari model dibuat, yakni: model analisis dan model desain. Model analisis merepresentasikan kebutuhan customer dengan menggambarkan suatu perangkat lunak kedalam tiga domain berbeda (domain informasi, domain fungsional, dan domain behavioral). Model desain merepresentasikan karakteristik dari perangkat lunak yang membantu pelaksana untuk mengkonstruksi secara efektif, seperti: arsitektur, user interface, dan component level detail.

d. Construction

Aktifitas construction merangkum kumpulan dari contoh coding dan testing yang menuntun perangkat lunak secara operasional, yaitu perangkat lunak yang sudah siap untuk diantar ke customer atau end-user. Dalam dunia rekayasa perangkat lunak modern, coding bisa berupa: (1) kreasi langsung dari source code bahasa pemrograman, (2) generasi otomatis source code seperti menggunakan representasi desain intermediate dari komponen untuk dibangun, (3) generasi otomatis dari executable code menggunakan bahasa pemrograman generasi ke-4 (misalnya, Visual C++). Fokus awal dari testing adalah pada level komponen, sering disebut juga unit testing. Level lainnya yaitu: (1) integration testing (diadakan sebagai sistem yang telah dikonstruksi), (2) validation testing yang mengkaji kebutuhan apakah telah cocok untuk sistem keseluruhan (atau software incrementnya), dan (3) acceptance testing yang diadakan oleh pengguna dalam sebuah usaha untuk mengetes semua fitur dan fungsi yang dibutuhkan.

e. Deployment

Aktifitas deployment merangkum tiga action: delivery, support, and feedback. Karena model proses perangkat lunak modern

adalah evolusioner secara alamiah, deployment tidak terjadi sekali, tetapi sejumlah waktu pada pergerakan perangkat lunak menuju penyelesaian. Setiap delivery cycle memperkenalkan customer dan end-user dengan sebuah operational software increment yang menyediakan fungsi dan fitur bermanfaat setiap support cycle menyediakan dokumentasi dan bantuan orang untuk semua fungsi dan fitur yang diperkenalkan selama semua siklus deployment berlangsung. Setiap feedback cycle menyediakan tim perangkat lunak dengan suatu petunjuk penting yang menghasilkan modifikasi fungsi, fitur, dan pendekatan yang diambil untuk increment berikutnya. Penyampaian dari software increment merepresentasikan milestone penting untuk beberapa software project.

Menurut (Saxena, 2019) penerapan metode incremental ini digunakan ketika: 1) jadwal pendanaan, risiko, kompleksitas program, atau kebutuhan untuk realisasi manfaat lebih awal, 2) saat persyaratan diketahui di awal, 3) ketika proyek memiliki jadwal pengembangan yang panjang dan 4) proyek dengan teknologi baru.

#### **4.5.1 Kelebihan Model Incremental**

Beberapa Kelebihan Dari Model Incremental antara lain (Saxena, 2019):

- a. Merupakan model dengan manajemen yang sederhana.
- b. Pengguna tidak perlu menunggu sampai seluruh sistem dikirim untuk mengambil keuntungan dari sistem tersebut. Increment yang pertama sudah memenuhi persyaratan mereka yang paling kritis, sehingga perangkat lunak dapat segera digunakan.
- c. Risiko untuk kegagalan proyek secara keseluruhan lebih rendah. Walaupun masalah masih dapat ditemukan pada beberapa increment. Karena layanan dengan prioritas tertinggi diserahkan pertama dan increment berikutnya diintegrasikan dengannya, sangatlah penting bahwa layanan sistem yang paling penting

mengalami pengujian yang ketat. Ini berarti bahwa pengguna akan memiliki kemungkinan kecil untuk memenuhi kegagalan perangkat lunak pada increment sistem yang paling bawah.

- d. Nilai penggunaan dapat ditentukan pada setiap increment sehingga fungsionalitas sistem disediakan lebih awal.
- e. Memiliki risiko lebih rendah terhadap keseluruhan pengembangan sistem.
- f. Prioritas tertinggi pada pelayanan sistem adalah yang paling diuji.
- g. Penerapan Sumber Daya Tambahan.

#### **4.5.2 Kelemahan Model Incremental**

Berikut ini kelemahan dari model incremental antara lain (Tedja, 2019):

- a. Kemungkinan tiap bagian tidak dapat diintegrasikan.
- b. Dapat menjadi build and fix model, karena kemampuannya untuk selalu mendapat perubahan selama proses rekayasa berlangsung.
- c. Harus Open Architecture.
- d. Mungkin terjadi kesulitan untuk memetakan kebutuhan pengguna ke dalam rencana spesifikasi masing-masing hasil increment.
- e. Membutuhkan waktu yang relatif lama untuk menghasilkan produk yang lengkap.

## **BAB V**

# **AGILE DEVELOPMENT MODEL**

Teknologi di era saat ini berkembang lebih cepat dari sebelumnya, memaksa perusahaan perangkat lunak global untuk bekerja dalam lingkungan yang berubah dengan cepat. Karena bisnis ini beroperasi dalam lingkungan yang selalu berubah, mustahil untuk mengumpulkan persyaratan perangkat lunak yang lengkap dan lengkap.

Tanpa persyaratan ini, praktis menjadi sulit bagi model perangkat lunak konvensional untuk bekerja. Model perangkat lunak konvensional seperti Model Waterfall yang bergantung pada spesifikasi persyaratan, perancangan, dan pengujian sistem tidak diarahkan untuk pengembangan perangkat lunak yang cepat. Akibatnya, model pengembangan perangkat lunak konvensional gagal memberikan produk yang dibutuhkan.

Model waterfall sangat populer untuk menyelesaikan sebuah proyek. Namun saat ini para pengembang menghadapi berbagai kendala saat menggunakannya dalam pengembangan perangkat lunak. Kesulitan utama termasuk menangani permintaan perubahan dari pelanggan selama pengembangan proyek dan biaya tinggi serta waktu yang dibutuhkan untuk memasukkan perubahan ini. Di sinilah pengembangan perangkat lunak tangkas datang untuk menyelamatkan. Ini dirancang khusus untuk mengatur kebutuhan lingkungan yang berubah dengan cepat dengan merangkul gagasan pengembangan bertahap dan mengembangkan produk akhir yang sebenarnya. Dalam mengatasi kelemahan model Waterfall ini, pada pertengahan 1990-an model Agile Software Development diusulkan.

Model Agile terutama dirancang untuk membantu proyek beradaptasi dengan permintaan perubahan dengan cepat. Jadi, tujuan

utama model Agile adalah untuk memfasilitasi penyelesaian proyek yang cepat. Untuk menyelesaikan tugas ini diperlukan ketangkasan.

Agility dicapai dengan menyesuaikan proses dengan proyek, menghilangkan aktivitas yang mungkin tidak penting untuk proyek tertentu. Juga, apa pun yang membuang-buang waktu dan tenaga dihindari. Sebenarnya model Agile mengacu pada sekelompok proses pengembangan. Proses ini memiliki beberapa karakteristik dasar tetapi memiliki perbedaan halus tertentu di antara mereka sendiri.

Dalam model Agile, persyaratan diuraikan menjadi banyak bagian kecil yang dapat dikembangkan secara bertahap. Model Agile mengadopsi pengembangan Iteratif. Setiap bagian inkremental dikembangkan melalui iterasi. Setiap iterasi dimaksudkan agar kecil dan mudah dikelola dan dapat diselesaikan hanya dalam beberapa minggu. Pada satu waktu satu iterasi direncanakan, dikembangkan, dan disebarkan ke pelanggan. Rencana jangka panjang tidak dibuat.

Model tangkas adalah kombinasi dari model proses iteratif dan inkremental. Langkah-langkah yang terlibat dalam model SDLC tangkas adalah: 1) requirement gathering, 2) requirement analysis, 3) design, 4) coding, 5) unit testing. Time-box mengacu pada jumlah waktu maksimum yang dibutuhkan untuk mengirimkan iterasi kepada pelanggan. Jadi, tanggal akhir untuk iterasi tidak berubah. Meskipun tim pengembangan dapat memutuskan untuk mengurangi fungsionalitas yang dikirimkan selama time-box jika perlu untuk mengirimkannya tepat waktu. Kumar menjelaskan tentang prinsip dasar dalam model agile sebagai berikut :

- a. Untuk menjalin kontak dekat dengan pelanggan selama pengembangan dan untuk mendapatkan pemahaman yang jelas tentang berbagai persyaratan, setiap proyek Agile biasanya menyertakan perwakilan pelanggan di tim. Di akhir setiap pemangku kepentingan iterasi dan tinjauan perwakilan pelanggan, kemajuan dibuat dan evaluasi ulang persyaratan.

- b. Model tangkas bergantung pada penerapan perangkat lunak yang berfungsi daripada dokumentasi yang komprehensif.
- c. Pengiriman versi tambahan perangkat lunak yang sering ke perwakilan pelanggan dalam interval beberapa minggu.
- d. Permintaan perubahan persyaratan dari pelanggan didorong dan dimasukkan secara efisien.
- e. Ini menekankan pada memiliki anggota tim yang efisien dan meningkatkan komunikasi di antara mereka menjadi lebih penting. Disadari bahwa peningkatan komunikasi di antara anggota tim pengembangan dapat dicapai melalui komunikasi tatap muka daripada melalui pertukaran dokumen formal.
- f. Direkomendasikan bahwa ukuran tim pengembangan harus dibuat kecil (5 hingga 9 orang) untuk membantu anggota tim terlibat secara bermakna dalam komunikasi tatap muka dan memiliki lingkungan kerja kolaboratif.
- g. Proses pada model agile biasanya menerapkan Pemrograman Berpasangan. Dalam Pemrograman berpasangan, dua pemrogram bekerja bersama di satu stasiun kerja. Yang satu melakukan pengkodean sementara yang lain meninjau kode seperti yang diketikkan. Kedua pemrogram mengganti peran mereka setiap sekitar satu jam.

Dengan prinsip-prinsip tersebut agile berusaha untuk menyiasati tiga masalah yang biasanya dihadapi saat proses pembuatan perangkat lunak, yaitu:

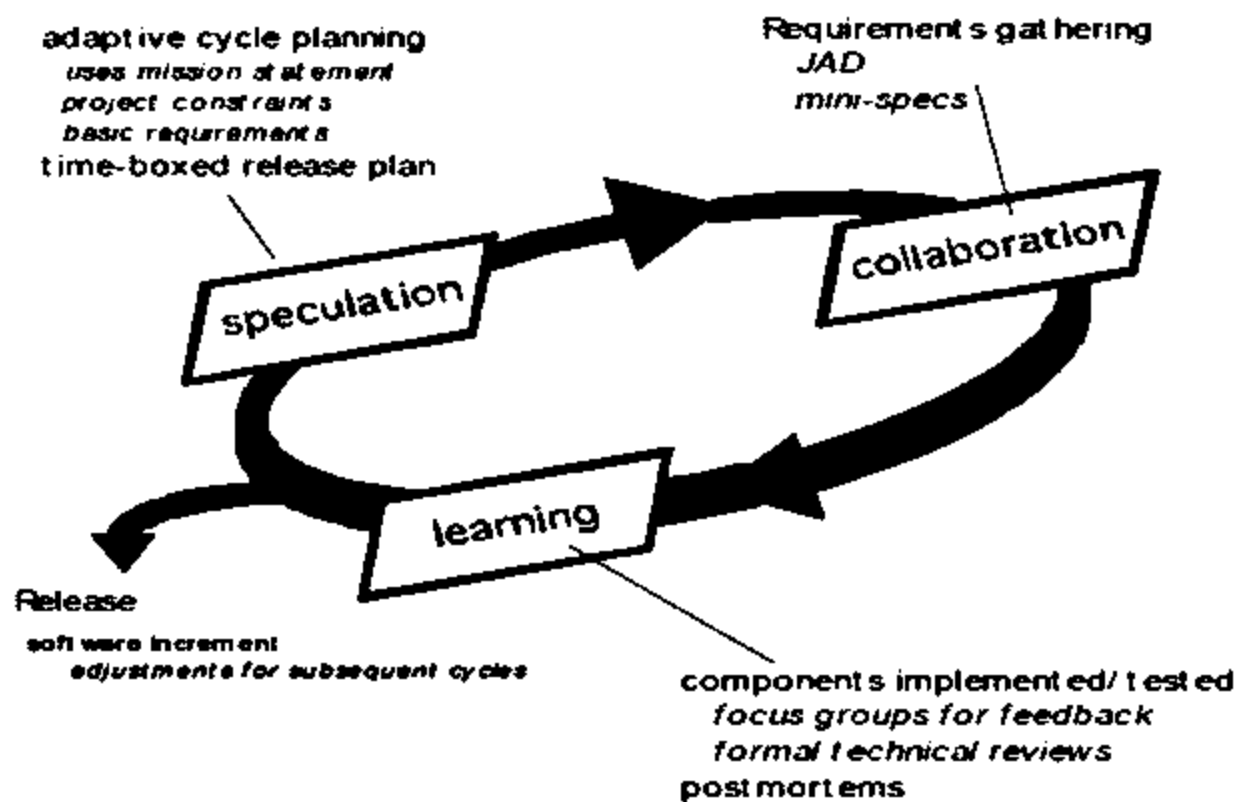
- a. Kebutuhan perangkat lunak sulit diprediksi dari awal dan selalu akan berubah. Selain itu, prioritas klien juga sering berubah seiring berjalannya proyek.
- b. Desain dan pembangunan sering tumpang tindih Sulit diperkirakan seberapa jauh desain yang diperlukan sebelum pembangunan.
- c. Analisis, desain, pembangunan dan testing tidak dapat diperkirakan seperti yang diinginkan

Model pengembangan agile mendukung berbagai metode SDLC. Beberapa metode agile berfokus pada praktik seperti XP, pragmatic programming, agile modeling, sementara beberapa model berfokus pada pengelolaan aliran pekerjaan seperti Scrum dan Kanban. Ada Beberapa metode dalam agile yang mendukung untuk spesifikasi dan pengembangan persyaratan dalam pengembangan perangkat lunak seperti FDD, ada juga metode agile yang berusaha untuk mencakup siklus hidup pengembangan penuh seperti DSDM dan RUP.

### **5.1 Adaptive Software Development (ASD)**

Adaptive Software Development adalah pendekatan extreme programming yang dimodifikasi, yang merupakan agile model yang paling banyak digunakan. Adaptive Software Development (ASD) merupakan salah satu model dari beberapa model, yang merupakan pengembangan dari agile methods. Oleh karena itu, ada sebagian orang menyebutnya sebagai agile software development (Sagala, 2014). ASD menekankan pada pengorganisasian tim secara mandiri, kolaborasi antar perseorangan, dan terus belajar, baik secara individu maupun secara tim.

ASD menggunakan tools yang disebut dengan “time-boxing” yaitu berupa aktifitas yang menentukan jangka waktu tertentu yang dialokasikan untuk menyelesaikan berbagai macam tugas. Apabila waktu yang ditentukan tersebut selesai, maka pembangunan sistem akan pindah ke tugas berikutnya, dengan harapan bahwa sebagian besar dari critical work telah berhasil diselesaikan sebelum waktu keseluruhan tugas berakhir. Berikut ini merupakan tahapan dari model Adaptive Software Development:



Gambar 5 1 Tahapan Adaptive Software Development  
(sumber: Pressman, 2010)

Dari gambar 5.1 dapat dijelaskan bahwa:

a. Speculation

Pada Tahap Speculation ini, proyek dimulai dan adaptive cycle planning diselenggarakan. Adaptive cycle planning yaitu menggunakan informasi awal seperti tujuan dari client, batas project yang akan dikerjakan, kebutuhan dasar pengembangan aplikasi. Pada tahap speculation ini juga, didefinisikan visi dan misi user (pengguna) terhadap sistem yang akan user buat. Lalu selanjutnya mendefinisikan project constraints. Contohnya waktu deliver (mengirim) . Dan setelah itu mendefinisikan satu set dari requirement (persyaratan) yang akan dikerjakan dalam suatu proses.

b. Collaboration

Pada tahap *collaboration* ini diorganisasikan tim kerja untuk membangun sebuah sistem dan direkomendasikan menggunakan model *Join Application Development (JAD)*. Model JAD merupakan tahapan atau langkah - langkah dan

merupakan salah satu prinsip bagaimana agar pengembangan sistem informasi sukses. Untuk proyek yang melibatkan tim terdistribusi dari berbagai mitra aliansi dan pengetahuan yang luas serta bagaimana orang berinteraksi dan bagaimana mereka mengelola ketergantungan adalah masalah penting. Untuk proyek yang lebih kecil dimana anggota tim bekerja dalam kedekatan fisik, kolaborasi dapat terdiri dari obrolan informal dan papan tulis. Namun, proyek yang lebih besar membutuhkan praktik tambahan, alat kolaborasi, dan interaksi manajer proyek. Kolaborasi, suatu tindakan penciptaan bersama dipupuk oleh kepercayaan dan rasa hormat. Penciptaan bersama mencakup tim pengembangan, pelanggan, konsultan luar, dan vendor. Tim harus berkolaborasi dalam masalah teknis, kebutuhan bisnis, dan pengambilan keputusan yang cepat.

c. Learning

Pada tahap Learning juga, tim pembangun sering sudah merasa tahu semua hal tentang proyek, padahal tidak selamanya begitu. Oleh karena itu, proses ini membuat mereka belajar lebih tentang proyek melalui 3 cara, yakni: 1) *focus group* yaitu klien dan pengguna memberi masukan terhadap software, 2) *formal Technique Reviews* yaitu Tim ASD lengkap melakukan review dan *postmortems* yaitu Tim ASD lakukan introspeksi pada kinerja dan proses. Belajar dari kesalahan dan eksperimen mengharuskan anggota tim membagikan kode dan artefak yang diselesaikan sebagian lebih awal, untuk menemukan kesalahan, belajar dari mereka dan mengurangi jumlah total pengerjaan ulang dengan menemukan masalah kecil sebelum menjadi masalah besar. Tim harus belajar membedakan antara pekerjaan jelek dan pekerjaan setengah jadi

Praktek dari Adaptive Software Development (ASD) digerakkan oleh kepercayaan pada adaptasi yang terus-menerus disesuaikan

untuk menerima perubahan yang berkesinambungan sebagai norma. Dalam ASD, siklus hidup plan design-build yang statis diganti dengan siklus hidup speculate collaborate learn yang dinamis. Siklus hidup ini didedikasikan untuk pembelajaran yang terus-menerus, re-evaluasi, mengamati ketidakpastian masa depan, and kolaborasi yang kuat diantara developer, manajemen, dan customer.

### **5.1.1 Kelebihan Adaptive Software Development**

Berikut ini merupakan kelebihan model adaptive software development antara lain :

- a. Meningkatkan kepuasan kepada client.
- b. dapat meakukan review pelanggan mengenai perangkat lunak yang dibuat lebih awal.
- c. pembangan sistem dibuat lebih cepat.
- d. mengurangi resiko kegagalan implementasi perangkat lunak dari segi non teknis.
- e. jika pada saat pembangunan sistem terjadi kegagalan kerudian dari segi material relatif kecil.

### **5.1.2 Kelemahan Adaptive Software Development**

- a. Pengembang harus selalu siap dengan perubahan karena akan selalu diterima.
- b. Proses pengembangan tidak akan berjalan dengan baik apabila kurangnya komitmen dari tim.
- c. Kurang cocok dengan tim sekala besar (>20 orang).
- d. Perkiraan waktu release dan harga perangkat lunak sulit untuk ditentukan.

## **5.2 Continuous Integration (CI)**

Continuous Integration merupakan metode pengembangan sistem yang menawarkan kemampuan integrasi sistem, uji coba fungsionalitas

serta penanaman aplikasi pada lingkungan produksi secara otomatis. Dengan menggunakan metode ini, memungkinkan kode sumber dari pengembang yang berbeda bisa diintegrasikan, Setelah diintegrasikan sistem akan dilakukan pengujian fungsionalitas untuk memastikan kelayakan dari sistem tersebut. Metode ini mendukung pengembangan aplikasi dapat dilakukan secara cepat dan memastikan aplikasi secara fungsionalitasnya dengan berjalan dengan baik.

Secara praktik pengembangan perangkat lunak dengan menggunakan metode ini yang dilakukan secara suatu tim. Anggota dalam tim tersebut melakukan proses integrasi secara berkala terhadap bagian tugas yang telah diberikan. Proses integrasi tersebut berlangsung pada suatu direktori pusat yang menyimpan aplikasi secara utuh dan nantinya akan menjadi hasil akhir dari proses pengembangan. Tiap anggota tim melakukan pengembangan modul yang sudah menjadi tugasnya pada direktori yang sudah ditentukan, sehingga anggota tim yang lain bisa mendapatkan perubahan yang terjadi pada perangkat lunak yang sedang dikembangkan. Setiap perubahan yang terjadi pada direktori pusat akan dilakukan uji coba fungsionalitas, sehingga bisa dipastikan bahwa direktori pusat menyimpan aplikasi yang benar.

Metode Continuous Integration harus didukung dengan alat kakas yang mempermudah pengembang dalam menyelesaikan project yang dilakukan dengan tim. Alat kakas tersebut disebut dengan Version Control System (VCS) adalah suatu sistem repositori berkas yang biasanya digunakan dalam pengembangan perangkat lunak. VCS memungkinkan repositori tersebut dipantau perkembangannya, mulai dari perubahan yang terjadi, pengguna yang melakukan perubahan, kapan perubahan terjadi, dan lain sebagainya. Oleh karena kemampuan yang dimilikinya, VCS cocok digunakan pada proses pengembangan perangkat lunak dengan metode kolaborasi. Beberapa contoh VCS yang sering digunakan, antara lain: Git, SVN, CVS, Mercurial, Bazaar, LibreSource, dan Monotone.

### **5.2.1 Kelebihan Continuous Integration**

Kelebihan yang bisa didapatkan dengan menggunakan metode Continuous Integration antara lain sebagai berikut :

- a. Mengurangi proses manual yang berulang.
- b. Mengurangi resiko karena mendeteksi & memperbaiki masalah integrasi yang terus menerus.
- c. Membuat proyek lebih baik dan jelas.
- d. Menghasilkan perangkat lunak yang dapat di-deploy kapan saja dan dimana saja.
- e. Menghemat waktu dan biaya selama proyek berlangsung.

### **5.2.2 Kelemahan Continuous Integration**

Berikut ini kelemahan dari metode Continuous Integration antara lain sebagai berikut :

- a. Memerlukan pengaturan awal terlebih dahulu tahap demi tahap.
- b. Memerlukan test kode untuk mencapai pengujian secara otomatis.
- c. Refactoring (melakukan perubahan pada kode program dari perangkat lunak dengan tujuan meningkatkan kualitas dari struktur program tersebut tanpa mengubah cara program tersebut bekerja) dalam skala besar dapat mengganggu karena dapat merubah basis kode.

## **5.3 Crystal Methods**

Ketika pada tahun 1991, IBM meminta Alistair Cockburn untuk mengembangkan metodologi untuk proyek berorientasi objek, dia tahu itu akan menjadi tantangan nyata karena dia tidak tahu banyak tentang metodologi proyek pada saat itu. Jadi, dia memutuskan untuk mewawancarai tim proyek dan mencari tahu pandangan mereka tentang proyek tersebut. Setelah melakukan penelitian ekstensif, dia sampai pada kesimpulan bahwa tim tinggi yang sukses berbagi pola

dan teknik yang sama bahkan tanpa menggunakan metodologi proyek tertentu. Dengan kata lain, mereka menambahkan nilai pada aspek-aspek seperti komunikasi yang erat, moral, akses ke pengguna, dan lainnya yang tidak dapat Anda temukan dalam metodologi tertentu. Akhirnya, dia menggunakan temuannya untuk membangun keluarga metodologi dan menamakannya Crystal. Metode kristal (Crystal Methods) adalah salah satu metode agile yang dikembangkan oleh Alistair Cockburn.

Metode ini memprioritaskan tim yang akan mengembangkan perangkat lunak tersebut daripada proses yang dilakukan untuk proyek tersebut.

Metode kristal adalah pendekatan pengembangan perangkat lunak tangkas yang berfokus terutama pada orang dan interaksi mereka saat mengerjakan proyek daripada pada proses dan alat. Alistair percaya bahwa keterampilan dan bakat orang serta cara mereka berkomunikasi memiliki pengaruh terbesar pada hasil proyek.

Metode Kristal didasarkan pada dua asumsi mendasar: Tim dapat menyederhanakan proses mereka sebagai pekerjaan mereka dan menjadi tim yang lebih optimal proyek itu unik dan dinamis dan membutuhkan metode khusus.

Menurut Cockburn, pengembangan produk harus dilihat sebagai permainan yang merangsang setiap orang untuk berinteraksi, berkreasi, dan menghasilkan ide-ide cemerlang. Dia mengatakan bahwa alih-alih berfokus pada pertanyaan seperti “apakah model kami akurat?” kita harus mencari jawaban untuk pertanyaan seperti “Apakah produk kita memenuhi kebutuhan pelanggan? atau “Apakah tujuan kita selaras sebagai sebuah tim?”

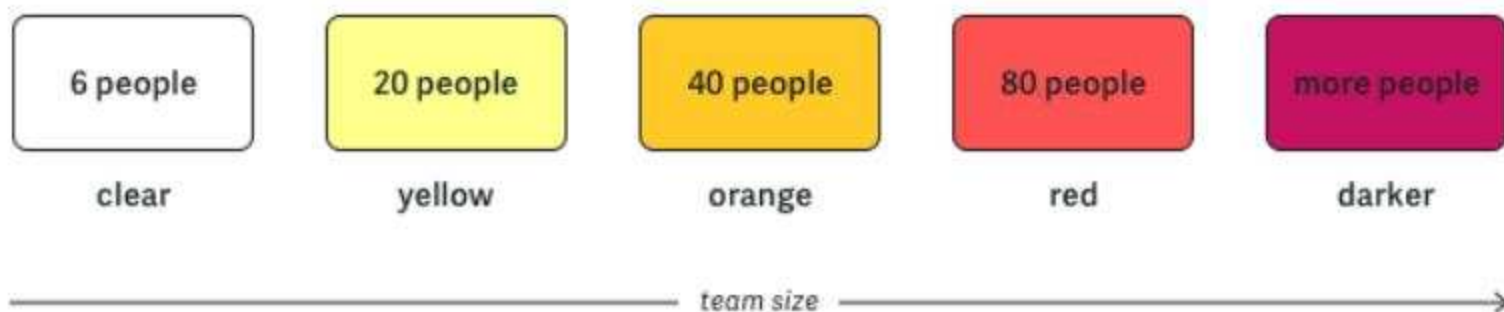
### **5.3.1 Anggota Keluarga Metode Kristal**

Salah satu hal yang ditemukan cockburn adalah bahwa properti proyek berubah tergantung pada jumlah orang yang terlibat dalam

proyek dan tingkat kekritisannya proyek yang sedang ditangani. Sementara tim yang lebih kecil dapat menangani dan membangun produk tanpa banyak pelaporan status dan dokumen, jumlah “artefak komunikasi” meningkat dengan tim yang lebih besar yang mengerjakan proyek skala besar. Dengan kata lain, semakin banyak orang yang anda miliki di tim, semakin penting proyek tersebut dan semakin kompleks pendekatan yang dibutuhkan. Oleh karena itu, tidak ada satu metode crystal; ada metodologi crystal yang berbeda untuk berbagai jenis proyek. Untuk membuat kategorisasi ini mudah dipahami, cockburn menamai metodologi crystal dan mengkategorikannya dalam dua dimensi ukuran dan kekritisannya yang cocok dengan mineral warna dan kekerasan.

Pada dasarnya, cockburn mengembangkan keluarga-keluarga ini untuk menunjukkan bahwa setiap proyek mungkin memerlukan serangkaian kebijakan, praktik, dan proses tertentu untuk memenuhi karakteristik unik proyek. cockburn mencoba menjelaskan hal ini dengan menyebut crystal “sekumpulan sampel yang anda sesuaikan dengan keadaan anda”. Pendekatan mana yang paling cocok untuk proyek anda bergantung pada tiga dimensi, yakni: team size, criticality, dan what the priority of the project is. Umumnya, mereka dicirikan oleh warna, sesuai dengan jumlah orang yang terlibat dalam proyek:

- a. Clear - untuk tim yang terdiri dari 8 orang atau kurang.
- b. Yellow - untuk tim yang terdiri dari 10-20 orang.
- c. Orange - untuk tim yang terdiri dari 20-50 orang.
- d. Red - untuk tim yang terdiri dari 50-100 orang.



### 5.3.2 Karakteristik metode kristal

Tidak seperti metodologi agile lainnya, crystal berfokus pada penyesuaian teknik yang digunakan dalam sebuah proyek dengan tujuan untuk memperkuat proses komunikasi tim. Berikut ini karakteristik dari metode kristal :

a. Human-powered

Ini berarti bahwa orang-orang yang terlibat dalam proyek sangat penting dan prosesnya harus disesuaikan untuk memenuhi kebutuhan masyarakat. Ini juga menekankan bahwa orang mampu mengatur diri mereka sendiri dan bahwa mereka dapat menjadi lebih terorganisir dan kompeten saat proses berkembang.

b. Adaptif

Crystal adalah metodologi peregangannya agar sesuai yang berarti bahwa proses dan alat tidak tetap, tetapi harus disesuaikan untuk memenuhi persyaratan tim dan proyek yang dihadapi.

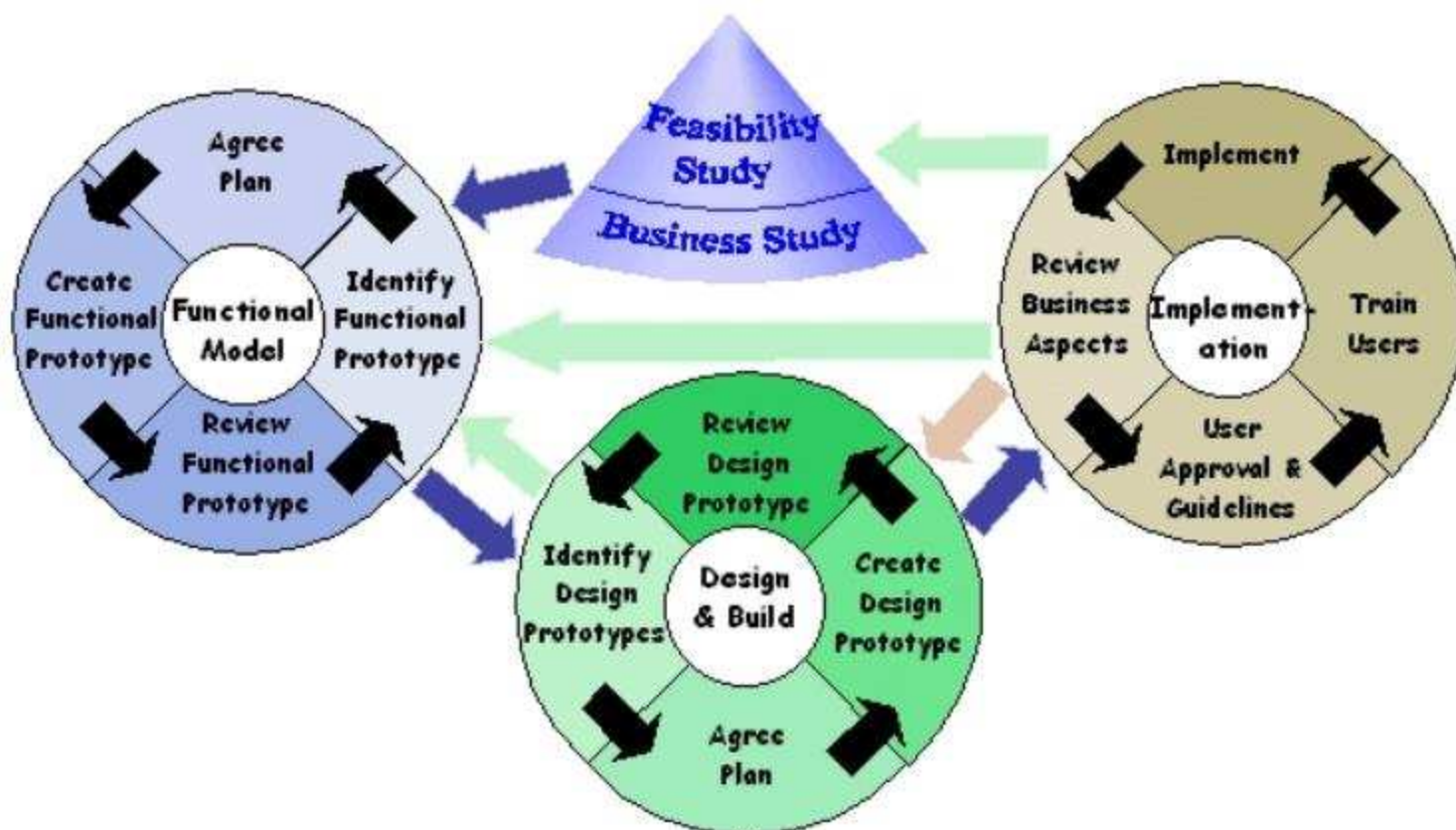
c. Ultra-light

Crystal tidak melibatkan terlalu banyak dokumentasi, manajemen dan pelaporan. Itu membuat segala sesuatunya ringan dengan berfokus pada alur kerja yang transparan antara tim dan klien dan dengan mempraktikkan komunikasi terbuka antara anggota tim.

### 5.4 Dynamic Systems Development Method (DSDM)

Dynamic Systems Development Method (DSDM) merupakan salah satu metode agile yang menyediakan kerangka kerja dalam pengembangan maupun pemeliharaan sistem yang menghadapi kendala waktu yang ketat. Metode ini awalnya didasarkan pada Rapid Application Development (RAD) dan mengutamakan keterlibatan pengguna secara berkesinambungan dengan pendekatan pengembangan secara berulang dan bertambah, tanggap terhadap perubahan, untuk membangun sistem perangkat lunak yang

memenuhi kebutuhan bisnis tepat waktu dan juga tepat anggaran. DSDM adalah pendekatan secara iteratif dan incremental yang menekankan pengguna secara kontinyu keterlibatan pelanggan. Tujuannya adalah untuk memberikan proyek-proyek tepat waktu dan menyesuaikan anggaran sementara untuk perubahan kebutuhan sepanjang jalan. DSDM adalah salah satu dari sejumlah metode Agile untuk mengembangkan perangkat lunak dan non IT solusi (Sugianto & Tjandra, 2016). Tahapan-tahapan dari Dynamic Systems Development Method dapat dilihat pada Gambar ....



a. Feasibility Study

Kesesuaian proyek awal dinilai dalam fase ini. Fase ini membantu untuk mengidentifikasi kelayakan proyek. Ruang lingkup dari studi kelayakan adalah untuk mengumpulkan rincian yang diperlukan tentang apakah solusi yang layak ada atau tidak.

b. Business Study

Setelah melakukan analisis kelayakan, langkah selanjutnya adalah menganalisis karakteristik bisnis dan teknologi. Studi Bisnis memberikan dasar untuk semua karya karya berikutnya. Fase

ini mengarah pada garis rinci proses bisnis yang terkena dampak dan informasi yang dibutuhkan.

c. **Functional model Iteration**

Persyaratan yang telah diidentifikasi pada tahap sebelumnya dikonversi ke model fungsional. Model ini terdiri dari prototype model dan fungsional. Prototyping adalah salah satu teknik proyek kunci dalam tahap yang membantu untuk mewujudkan keterlibatan pengguna yang baik di seluruh proyek.

d. **Design and Build Iteration**

Fokus utama dari iterasi DSDM adalah untuk mengintegrasikan komponen fungsional dari tahap sebelumnya ke dalam satu sistem yang memenuhi kebutuhan pengguna. Tahap ini juga membahas kebutuhan non-fungsional yang telah ditetapkan dalam pengembangan proyek.

e. **Implementasi(Implementation)**

Pada tahap implementasi, sistem diuji termasuk dokumentasi pengguna dikirim ke pengguna dan pelatihan calon pengguna direalisasikan. Sistem akan dikirimkan telah ditinjau untuk menyertakan persyaratan yang telah ditetapkan pada tahap awal proyek.

#### **5.4.1 Kelebihan metode DSDM**

Berikut ini merupakan Kelebihan - kelebihan dari metode DSDM di antaranya adalah:

- a. Pengguna secara aktif terlibat dalam pengembangan sistem sehingga dapat mempengaruhi arah proyek.
- b. Fungsionalitas disampaikan dengan cepat dan berkala.
- c. Pengerjaan sistem dilakukan tepat waktu dan tepat anggaran.

#### **5.4.2 Kelemahan Metode DSDM**

Kelemahan dari metode Dynamic Software Development Method (DSDM) antara lain :

- a. Setiap iterasi bergantung pada prototype sebelumnya.
- b. Menentukan scope dari suatu prototype proyek tidak pernah selesai.
- c. Dokumentasi sering kali tidak lengkap fokus pada pembuatan prototype Isu-isu mengenai system backup and recovery, system performance dan system security kurang/tidak diperhatikan dan sering terlupakan.

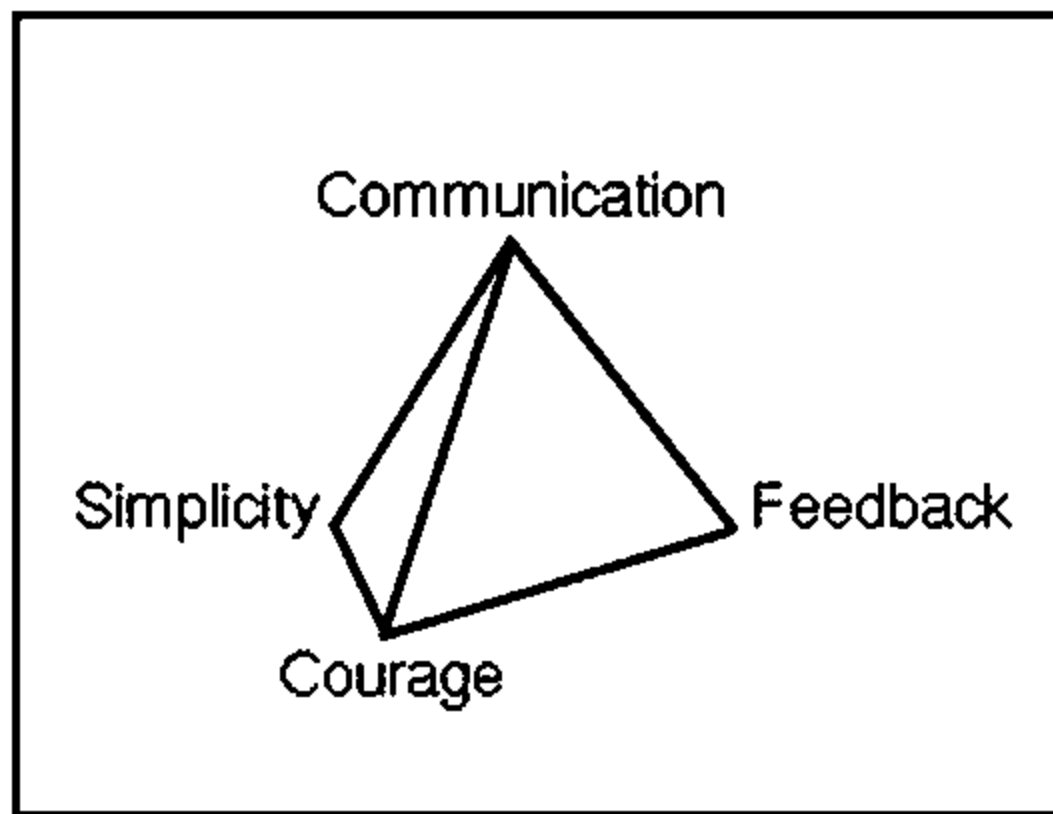
## **5.5 Extreme Programming (XP)**

Extreme Programming merupakan salah satu dari pendekatan agile software development yang paling sering digunakan. Extreme Programming (XP) merupakan sebuah proses rekayasa perangkat lunak yang cenderung menggunakan pendekatan berorientasi objek dan sasaran dari metode ini adalah tim yang dibentuk dalam skala kecil sampai medium serta metode ini juga sesuai jika tim dihadapkan dengan requirement yang tidak jelas maupun terjadi perubahan-perubahan requirement yang sangat cepat (Supriyatna, 2018).

Extreme Programming (XP) dikenal dengan metode atau “technical how to” bagaimana suatu tim teknis mengembangkan perangkat lunak secara efisien melalui berbagai prinsip dan teknik praktis pengembangan perangkat lunak. XP menjadi dasar bagaimana tim bekerja sehari-hari. Metode XP adalah melakukan penyederhanaan berbagai tahapan pengembangan sistem informasi menjadi lebih efisien, adaptif dan fleksibel. Nilai dasar extreme programming antara lain communication, Courage, Simplicity, Feedback dan Quality Work.

### **5.5.1 Nilai Utama Dalam Extreme Programming**

Terdapat nilai utama pada XP yang mendasar pada setiap tahapan proses pengembangan sistem informasi yaitu :



a. Komunikasi

XP memfokuskan pada hubungan komunikasi yang baik antar anggota tim. Para anggota tim harus membangun saling pengertian, mereka juga wajib berbagi pengetahuan dan keterampilan dalam mengembangkan perangkat lunak. Ego dari para programmer yang biasanya cukup tinggi harus ditekan dan mereka harus membuka diri untuk bekerja sama dengan programmer lain dalam menuliskan kode program.

b. Courage

Para anggota tim dan penanggung jawab pengembang perangkat lunak harus selalu memiliki keyakinan dan integritas dalam melakukan tugasnya. Integritas ini harus selalu dijaga bahkan dalam kondisi adanya tekanan dari situasi sekitar (misalnya oleh klien atau pemilik perusahaan), untuk dapat melakukan sesuatu dengan penuh integritas para anggota tim harus terlebih dahulu memiliki rasa saling percaya.

c. Simplicity

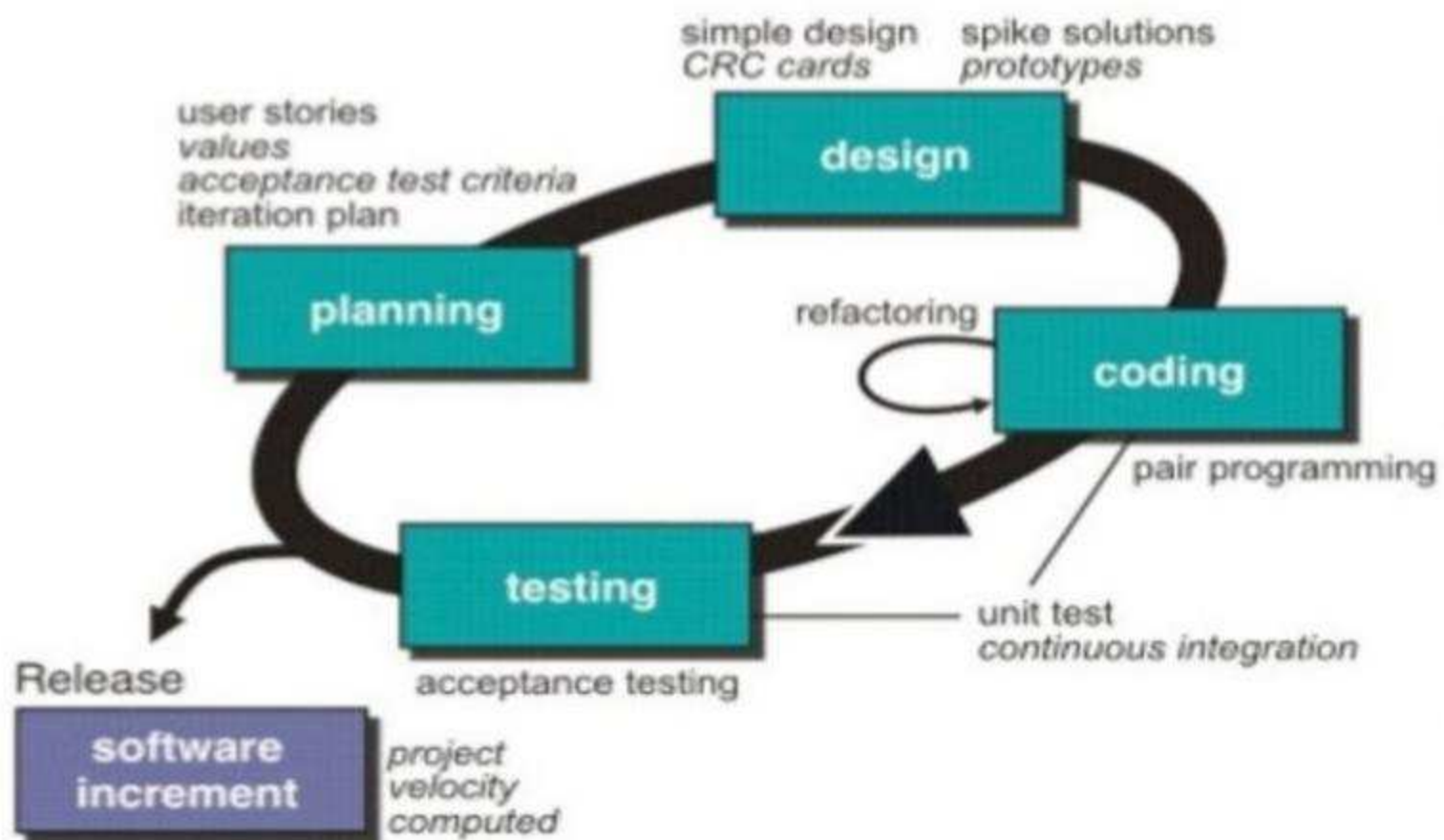
Lakukan semua dengan sederhana. Hal tersebut adalah salah satu nilai dasar dari XP. Gunakan method yang pendek dan simple, jangan terlalu rumit dalam membuat desain, hilangkan fitur yang

tidak ada gunanya, dan berbagai proses penyederhanaan lain akan selalu menjadi nilai utama dari setiap aspek XP.

d. Umpan balik (feedback)

Berikan selalu feedback kepada sesama anggota tim maupun pihak - pihak lain yang terlibat dalam pengembagnan perangkat lunak. Utarkan selalu pikiran anda dan diskusikan kesalahan-kesalahan yang muncul selama proses proses pengembagnan. Dengarkan selalu pendapat rekan yang lain, dengan adanya feedback inilah seringkali kita menyadari bagian mana yang salah atau bisa dtingkatkan lagi dari perangkat lunak yang dikembangkan.

### 5.5.2 Tahapan Dalam Extreme Programming



XP menggunakan konsep pendekatan yang berorientasi objek dan memiliki empat kerangka kegiatan, yaitu :

a. Planning

Kegiatan Perencanaan dimulai dengan mengumpulkan requirement yang memungkinkan para anggota teknis tim

XP memahami konteks bisnis untuk software dan untuk mendapatkan pandangan luas untuk output yang diperlukan dan fitur utama fungsinya. Hal ini akan mengarah ke penciptaan satu set “cerita” (biasa disebut juga cerita user) yang mendeskripsikan output yang dibutuhkan, fitur, dan fungsi dari software yang akan dibuat. Setiap cerita ditulis oleh customer dan ditempatkan pada kartu indeks dan memberikan skala prioritas berdasarkan pada keseluruhan nilai bisnis dari fungsi fitur tersebut. Jika cerita tersebut dianggap membutuhkan waktu lebih dari tiga minggu, customer diminta untuk membagi cerita tersebut menjadi cerita yang lebih kecil lagi dan perhitungan nilai serta biaya akan terjadi lagi. Penting untuk diingat bahwa cerita baru dapat ditulis setiap saat.

b. Design

Desain pada XP mengikuti prinsip KIS (“Keep It Simple”). Desain yang sederhana selalu lebih dipilih dibandingkan dengan desain yang kompleks. Desain untuk fungsi tambahan (karena developer merasa akan diminta nanti) tidak disarankan. XP menerapkan penggunaan CRC (Class Responsibility Card) sebagai mekanisme yang efektif untuk memikirkan mengenai software dalam konteks berorientasi objek. CRC akan mengidentifikasi dan mengorganisasikan class berorientasi objek yang sesuai dengan peningkatan software. Jika terjadi masalah dalam pembuatan desain cerita, XP merekomendasikan suatu solusi yang disebut “Spike Solution”. Spike solution adalah pembuatan segera suatu prototype operasional dari sebagian desain yang mengalami masalah.

c. Coding

Setelah cerita selesai dikembangkan dan desain kerja awal selesai, tim XP tidak segera melanjutkan ke tahap coding, melainkan mengembangkan serangkaian tes unit yang akan dijalankan pada

setiap cerita yang akan dibuat. Saat unit tes telah dibuat, developer akan lebih fokus kepada apa yang harus dibuat untuk melewati tes tersebut. Tidak ada sesuatu yang berlebihan yang ditambahkan (prinsip KIS). Saat kode selesai dibuat, kode dapat diuji segera, sehingga dapat memberikan umpan balik segera kepada developer. XP merekomendasikan konsep “Pair Programming” yaitu: dua orang bekerja bersama dalam pembuatan kode untuk cerita. Hal ini memberikan kesempatan pada pemecahan masalah secara langsung dan membuat developer lebih fokus kepada masalah yang sedang ditanganinya. Pada dasarnya pada pair programming, kedua developer memiliki peran yang berbeda yaitu untuk memikirkan mengenai detail coding pada desain, dan untuk memastikan bahwa coding yang dibuat sudah sesuai standar dan melewati tes unit.

d. Testing

Setelah tahapan pengkodean selesai, kemudian dilakukan tahapan pengujian sistem untuk mengetahui kesalahan apa saja yang timbul saat aplikasi sedang berjalan serta mengetahui apakah sistem yang dibangun sudah sesuai dengan kebutuhan pengguna.

### 5.5.3 Kelebihan Metode Extreme Programming

Metode XP merupakan metode yang sering digunakan oleh para pengembang perangkat lunak. Berikut ini merupakan kelebihan dari metode ini antara lain :

- a. Komunikasi dalam XP dibangun dengan melakukan pemrograman berpasangan (pair programming). Developer didampingi oleh pihak klien dalam melakukan coding dan unit testing sehingga klien bisa terlibat langsung dalam pemrograman sambil berkomunikasi dengan developer. Selain itu perkiraan beban tugas juga diperhitungkan.

- b. Menekankan pada kesederhanaan dalam pengkodean: “What is the simplest thing that could possibly work?” Lebih baik melakukan hal yang sederhana dan mengembangkannya besok jika diperlukan. Komunikasi yang lebih banyak mempermudah, dan rancangan yang sederhana mengurangi penjelasan.
- c. Setiap feed back ditanggapi dengan melakukan tes, unit test atau system integration dan jangan menunda karena biaya akan membengkak (uang, tenaga, waktu).
- d. Banyak ide baru dan berani mencobanya, berani mengerjakan kembali dan setiap kali kesalahan ditemukan, langsung diperbaiki.

#### **5.5.4 Kelemahan Metode Extreme Programming**

Selain memiliki banyak kelebihan metode XP ini juga memiliki kelemahan dalam prakteknya. Berikut ini kelemahan dari metode XP antara lain :

- a. Developer harus selalu siap dengan perubahan karena perubahan akan selalu diterima.
- b. Tidak bisa membuat kode yang detail di awal (prinsip simplicity dan juga anjuran untuk melakukan apa yang diperlukan hari itu juga).

#### **5.6 Feature Driven Development (FDD)**

Feature Driven Development (FDD) merupakan salah satu metode Agile yang pertama kali diperkenalkan pada tahun 1999 oleh Peter Coad dkk dalam bukunya yang berjudul “java modeling in color with UML: enterprise components and process”. Awal metode ini dirancang adalah sebagai model praktis dalam pengembangan perangkat lunak berorientasi objek. FDD merupakan proses pengembangan perangkat lunak yang dirancang dan diimplementasikan untuk memberikan hasil kerja yang terbaik. Dalam proses pengembangannya dilakukan secara berulang dalam waktu yang telah ditentukan dan dapat diukur

(Palmer & Felshing, 2002).

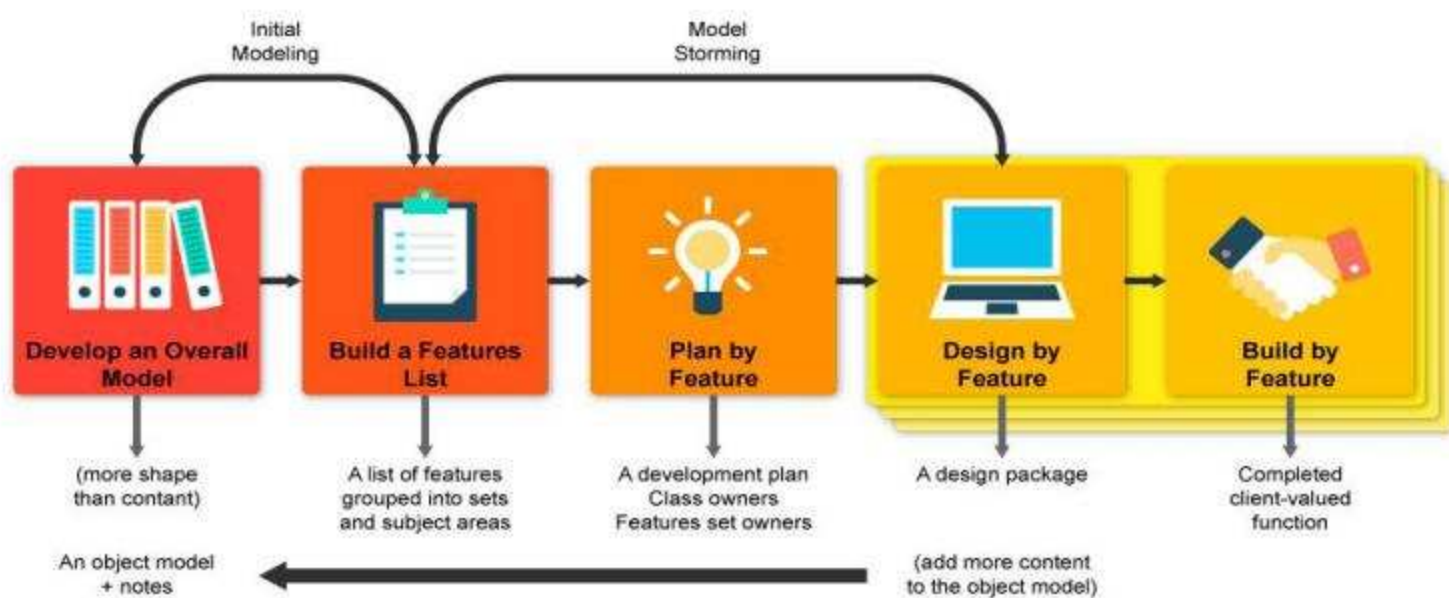
FDD Memecah kumpulan fitur menjadi potongan-potongan yang lebih kecil dan rilis secara berulang dengan teratur, yang membuatnya lebih mudah untuk melacak dan memperbaiki kesalahan pengkodean, mengurangi risiko, dan memungkinkan untuk membuat perputaran cepat dalam hal memenuhi kebutuhan klien. Dengan menggunakan teknik pengembangan seperti itu diharapkan proses pengembangan aplikasi lebih cepat dan adaptif, terlebih dalam pendekatan metode FDD ini tidak membahas pengembangan software secara keseluruhan tetapi lebih berfokus pada tahap perancangan serta pembuatan program.

Pada dasarnya pendekatan menggunakan FDD ini, mengacu pada pembuatan perangkat lunak yang mudah dimengerti dan diimplementasikan, teknik problem solving, dan pelaporan yang mudah dimengerti dan dikontrol oleh stakeholders. Seperti metode agile lainnya, metode FDD ini menekankan pada 1) kerja sama tim, 2) pengelolaan masalah dan kompleksitas proyek menggunakan dekomposisi berbasis fitur diikuti oleh integrasi software increment, serta 3) komunikasi detail teknis menggunakan cara verbal, grafis, dan berbasis teks.

Dalam metode ini, pengembang diberikan informasi yang cukup sesuai dengan kebutuhan dalam pengembangan perangkat lunak serta alat bantu dalam menyelesaikan proyek. Pengguna sistem yang dikembangkan dapat secara langsung melihat bagian mereka sebagai hasil progress pengembangan dan memberikan saran dalam tahap pengembangan yang bertujuan untuk menciptakan sistem sesuai keinginan pengguna. FDD memberikan kegiatan jaminan kualitas perangkat lunak (software quality assurance) dengan mendorong strategi pengembangan tambahan, penggunaan desain dan inspeksi kode, penerapan audit jaminan kualitas perangkat lunak, pengumpulan metrik, dan penggunaan pola (Pressman, 2010).

### 5.6.1 Fase Pengembangan Feature Driven Development

FDD terdiri dari 5 tahap secara berurut dalam pengembangan perangkat lunak. Proses FDD yang interaktif dalam mendesain dan membangun (design and build) mendukung metode Agile dengan adaptasi yang cepat terhadap perubahan requirement dan kebutuhan bisnis, berikut ini merupakan tahapan pengembangan perangkat lunak menggunakan FDD :



(Mirzoyan, 2020)

a. Develop an overall model

Fase ini dimulai dengan Pembuatan dokumen kebutuhan seperti use case atau spesifikasi fungsional. Namun FDD tidak secara eksplisit menggali, mencari dan mengatur requirement ini. Domain expert menyajikan apa yang disebut “walkthrough” yang mana anggota tim dan Chief Architect diinformasikan dengan deskripsi level tinggi dari sistem. Domain keseluruhan (overall domain) lebih lanjut dibagi kedalam area domain yang berbeda sedangkan walkthrough yang lebih detail diberikan oleh anggota domain. Kemudian anggota tim developer bekerja dalam grup-grup kecil untuk mengerjakan project model dari domain area yang telah diterima.

b. Build a features list

Dalam daftar (list), tim menyajikan masing-masing client valued functions ke dalam sistem. Fungsi-fungsi tersebut dibagikan kepada masing-masing domain area dan masing-masing grup dari fungsi tersebut disebut sebagai major feature set. Sebagai tambahan, major feature sets kemudian dibagi lagi menjadi feature sets. Ini merepresentasikan aktifitas yang berbeda di setiap domain area. Feature list adalah yang dilihat oleh user atau sponsor untuk validitas dan kelengkapan mereka. Feature dalam hal ini adalah langkah-langkah aktifitas bisnis, berbasis customer bukan teknologi. Bahasa yang digunakan mencakup bahasa yang dimengerti oleh customer.

c. Plan by Features

Mencakup perencanaan pada level yang lebih tinggi, dimana feature set diatur sedemikian rupa sesuai dengan prioritas dan hubungannya. Prioritas ditentukan sesuai dengan kebutuhan customer yang disetujui oleh Chief Programmer. Dalam fase ini, Project Manager, Development Manager dan Chief Programmer merencanakan urutan feature-feature yang akan dikerjakan dengan demikian class ownership teliti dilengkapi.

d. Design by Features dan build By Features

Sekelompok kecil fitur diambil dari feature set dan diperlukan feature team untuk membangun fitur terpilih yang disebut sebagai class owner. Proses design by feature dan build by feature bersifat iteratif selama fitur yang dipilih tersebut diproduksi. Satu kali iterasi memerlukan waktu beberapa hari sampai 2 minggu. Proses iteratif ini mencakup beberapa tugas seperti inspeksi rancangan, pengkodean, pengujian unit, integrasi dan inspeksi kode

e. Karakteristik

Menurut Calberg, penggunaan FDD sebaiknya digunakan jika mempekerjakan 10 – 250 developer yang memiliki kemampuan

teknis lebih dari rata-rata dan jangan digunakan jika jumlah tim kurang dari 10, tim sedang belajar menguasai pekerjaan dan jika kurang dukungan dari sistem. FDD lebih terhirarki daripada Extreme Programming, memiliki class ownership yang terpisah-pisah, sukses jika dalam rentang jumlah developer diatas rata-rata, klien tidak dilibatkan dalam.

#### **5.6.2 Kelebihan Metode Feature Driven Development**

Berikut ini merupakan kelebihan metode Feature Driven Development dalam pengembangan perangkat lunak:

- a. Memberi pemahaman yang sangat baik terhadap tim tentang ruang lingkup dan konteks proyek.
- b. Membutuhkan lebih sedikit pertemuan. Salah satu keluhan yang sering dikeluhkan tentang agile adalah terlalu banyak meeting. FDD menggunakan dokumentasi untuk berkomunikasi.
- c. Menggunakan pendekatan yang berpusat pada pengguna. FDD, klien adalah pengguna akhir.
- d. Bekerja dengan baik dengan proyek skala besar, jangka panjang, atau berkelanjutan. Metodologi ini sangat skalabel dan dapat berkembang seiring pertumbuhan perusahaan dan proyek Anda. Kelima langkah yang didefinisikan dengan baik sehingga memudahkan anggota tim yang baru atau karyawan baru dalam beradaptasi serta mempercepat proyek penyelesaian proyek.

#### **5.6.3 Kelemahan Metode Feature Driven Development**

Selain memiliki banyak kelebihan ternyata metode Feature Driven Development juga memiliki kelemahan. Berikut ini merupakan kelemahan metode FDD :

- a. Tidak ideal untuk proyek skala kecil dan tidak berfungsi apabila dalam pengembangan perangkat lunak yang hanya ada satu pengembang saja.

- b. Lebih menekankan pengembang dengan keterampilan tinggi dan sulit untuk mencari pengembang dengan kriteria tersebut.
- c. Dalam FDD terdapat class owner yang menangani setiap fitur. Masalahnya adalah ketika fitur A memiliki dependensi terhadap fitur B dan fitur B mengalami perubahan, maka fitur A harus menunggu kepastian dari fitur B yang menyebabkan mundurnya jadwal proyek.
- d. Menempatkan ketergantungan tinggi pada kepala tim yang harus mampu bertindak sebagai koordinator, perancang utama, dan mentor bagi anggota tim baru.
- e. Tidak memberikan dokumentasi tertulis kepada klien, meskipun ada banyak komunikasi yang terdokumentasi diantara anggota tim selama siklus pengembangan proyek.

## 5.7 Kanban

Kanban adalah metode untuk mendefinisikan, mengelola, dan meningkatkan layanan yang memberikan pekerjaan pengetahuan, seperti layanan profesional, upaya kreatif, dan desain produk fisik serta perangkat lunak (Kniberg & Skarin, 2010). Kanban merupakan metode untuk secara bertahap meningkatkan apapun yang dikerjakan, baik perangkat lunak pengembangan, IT / Operasi, Penetapan Staf, Perekrutan, Pemasaran dan Penjualan atau Pengadaan. Kanban muncul menggunakan rasa “kartu visual”, “papan nama”, “Billboard”, atau “sistem pensinyalan” untuk menunjukkan alur kerja yang membatasi Work In Progress (WIP).

Tugas dan cerita direpresentasikan pada kartu tersebut. Status saat ini tiap tugas dikenal dengan menampilkan kartu pada kolom-kolom terpisah yang terdapat pada papan kanban. Inti dari metodologi ini adalah pembuatan “board” dimana kita meletakkan “card” atau kartu yang berisi tugas yang perlu diselesaikan Dalam bentuk paling sederhana, sebuah Kanban Board berisi tiga kolom: yang perlu

dikerjakan, yang sedang dikerjakan, yang telah dikerjakan (Graham, 2018).

### 5.7.1 Prinsip Dasar Kanban

Kanban merupakan cara manajemen dengan memvisualisasikan seluruh alur kerja daripada melihat hasil dari setiap proses yang memiliki tujuan untuk menyeimbangkan permintaan dengan kapasitas yang tersedia dan mengidentifikasi potensi kemacetan dalam proses dan mengatasi kemacetan tersebut (Faizah et al., 2019). Kanban memiliki 3 prinsip dasar yaitu (Hammarberg & Sunden, 2014):

- a. Prinsip 1 (P1) - Visualize Work (visualisasikan pekerjaan).  
Kanban Board merupakan model visual untuk menggambarkan alur kerjanya yang membantu untuk mengamati dan memeriksa alur dari awal hingga akhir. Kanban board dibuat sesuai dengan tahap-tahap untuk melakukan pengembangan perangkat lunak.
- b. Prinsip 2 (P2) - Limit Work In Progress (batasi pekerjaan yang sedang berlangsung).  
Tim pada tahap awal menentukan batas pekerjaan pada setiap alur yang ada pada kanban board yang disebut "Work In Progress (WIP)". Tujuan dari WIP ini untuk mengurangi pemborosan dan membantu tim untuk fokus menyelesaikan pekerjaan yang sedang ada pada progres lalu memulai pekerjaan baru setelahnya.
- c. Prinsip 3 (P3) - Focus on Flow of Work (fokus pada alur kerja).  
Hasil yang efektif yang didapatkan dari kanban adalah tim harus fokus pada aliran pengerjaan proyek dari awal hingga proyek selesai. Dengan mengikuti dua prinsip yang telah dijelaskan diatas untuk membantu dalam pencapaian fokus ini. Fokus pada aliran pengerjaan proyek ini mengarahkan tim untuk menggambarkan kemacetan yang akan terjadi dan menindaklanjutinya, sehingga aliran pengerjaan proyek ini dapat berjalan dengan tetap.

### 5.7.2 Fungsi Kanban

Kanban mempunyai dua fungsi utama yaitu sebagai pengendalian produksi dan sebagai sarana peningkatan produksi. Fungsinya sebagai pengendali produksi diperoleh dengan menyatukan proses bersama dan mengembangkan suatu sistem yang tepat waktu sehingga bahan baku, komponen atau produk yang dibutuhkan akan datang pada saat dibutuhkan dalam jumlah yang sesuai dengan kebutuhan diseluruh workcenter yang ada di rantai produksi, bahkan meluas sampai ke pemasok yang terkait dengan perusahaan. Sedangkan fungsinya sebagai sarana peningkatan produksi dapat diperoleh jika penerapannya dengan menggunakan pendekatan pengurangan tingkat persediaan. Tingkat persediaan dapat dikurangi secara terkendali melalui pengurangan jumlah Kanban yang beredar selama proses produksi. Menurut Yasuhiro Monden secara terperinci sistem kanban digunakan untuk melakukan fungsi sebagai berikut (Monden, 1995) :

- a. Perintah Kanban berlaku sebagai alat perintah antara produksi dan pengiriman. Kanban yang dituliskan merupakan suatu alamat yang menginformasikan proses sebelum tempat penyimpanan komponen yang telah diolah, dan menginformasikan proses yang sesudah tempat komponen yang dibutuhkan.
- b. Pengendalian diri sendiri untuk mencegah over production. Sistem kanhan merupakan mekanisme pengendalian diri sendiri sehingga memungkinkan tiap proses melakukan penyesuaian kecil terhadap pasokan untuk jadwal produksi bulanannya karena adanya fluktuasi permintaan bulanan.
- c. Pengendalian visual sistem Kanban berlaku sebagai alat untuk pengendahan visual karena bukan saja memberikan informasi numerik, tetapi juga informasi fisik dalam bentuk kartu kanban.
- d. Perbaikan proses dan operasi manual penggunaan sistem Kanban untuk membantu perbaikan operasi sangat dibutuhkan karena

peningkatan produktivitas mengakibatkan perbaikan keuangan sehingga memperbaiki perusahaan secara keseluruhan.

- e. Pengurangan biaya pengelolaan sistem Kanban juga berfungsi mengurangi biaya manajemen dengan membantu mengurangi jumlah perencanaan menjadi nol.

### 5.7.3 Cara Kerja Kanban

Sistem kanban merupakan suatu cara mengendalikan material dalam proses manufaktur. Kartu, sebagai sarana penanda adanya pergerakan barang atau material diproses manufaktur/produksi. Implementasi sistem Kanban dapat dilakukan menggunakan sistem yaitu sistem Kanban yang menandai pergerakan barangnya dengan menggunakan 2 jenis kartu. Jenis kartunya adalah Production Withdrawal Kanban (PWK) dan Production Instruction Kanban (PIK). PWK dipakai ketika operator pabrik meminta atau menarik barang yang dibutuhkan untuk diproses. PIK dipakai ketika operator pabrik sedang melakukan pemrosesan barang yang sedang diminta (Naufal et al., 2012).

Pabrik di PT. Inkoasku melakukan manajemen produksi yang menggunakan 2 tipe Kanban, kartu PWK dinamakan kartu Kanban tarik, dan kartu PIK dinamakan kartu Kanban produksi. Dalam sebuah perusahaan manufaktur, sistem tarikan tersebut dapat menjadi dasar pembuatan rancangan penjadwalan produksi barang dari area kerja bahan mentah sampai area kerja barang jadi.

Sistem kanban adalah sistem yang mempermudah operator di lini pabrik untuk melakukan permintaan barang atau material untuk bisa diproses. Gambarnya adalah ketika dari area barang jadi (finished good) akan melakukan proses pengiriman barang sesuai dengan order customer. Ketika muncul kondisi, dimana barang yang butuh untuk dikirim kosong, maka akan dilakukan permintaan pada area kerja yang memproses barang di area sebelumnya. Permintaan tersebut

ditandai dengan menggunakan kartu Kanban tarik. Permintaan tersebut ditanggapi dengan pemberian barang hasil proses ke area peminta, yaitu area barang jadi/finished good. Ketika barang sedang diproses lini penyedia, operator menggunakan kartu Kanban produksi. Demikian seterusnya dan proses berulang sampai ke area kerja paling ujung, misalnya pada bagian gudang raw material/bahan baku, atau bahkan supplier.

Sistem Kanban yang diterapkan secara digital, misalnya sistem Kanban disuatu ERP, memerlukan suatu sistem tracking. Karena sistem tracking dapat memberikan kontribusi pelacakan kartu Kanban yang beredar dialur proses kerja. Sistem tracking tersebut dapat menjadi alat pengumpul data, yang nantinya dapat digunakan untuk laporan yang berhubungan dengan proses produksi, misalnya laporan operasional bulanan.

## **5.8 Lean software development**

Lean adalah pendekatan sistematis untuk mengidentifikasi dan menghilangkan limbah melalui perbaikan secara terus-menerus, selalu meminta masukan kepada pelanggan dalam tujuan untuk mengejar kesempurnaan. Lean Software Development adalah sebuah strategi yang dilakukan untuk mengurangi cacat dan menghemat waktu pengerjaan serta selalu memberikan nilai bisnis kedalam produk secara bertahap. Lean Software merupakan hasil adopsi dari Lean Manufacturing, prinsip Lean IT, Praktik Software Development Domain serta merupakan hasil adaptasi dari Toyota Production System (Monden, 1995).

Lean software development bersumber dari sebuah buku, yang ditulis oleh Mary Poppendieck dan Tom Poppendieck (Marry & Tom, 2003). Buku ini menyajikan beberapa prinsip tradisional Lean dalam bentuk yang telah dimodifikasi. Keterlibatan Poppendieck's di dalam komunitas Agile Software Development, termasuk sebagai pembicara

di beberapa konferensi Agile, yang menghasilkan konsep-konsep lebih luas untuk komunitas Agile

Lean software development adalah suatu proses engineering yang digunakan untuk mengembangkan dan menghasilkan suatu software berkualitas tinggi yang telah terjamin keandalannya sehingga tidak terjadi kegagalan dalam penggunaan software tersebut. Lean software development ini berpedoman pada pemahaman lapangan dan kesesuaian pelaksanaan prinsip lean disepanjang seluruh proses pengembangan software. Slogan yang dipakai yaitu: berpikir besar, bertindak kecil, gagal cepat, dan belajar cepat. Lean dapat mereduksi waktu pengembangan software karena waktu pengembangan software dapat direduksi dengan cara mengurangi error pengerjaan software dengan menggunakan tujuh prinsip Lean, yaitu: Eliminate Waste, Amplifying Learning, Decide As Late As Possible, Deliver As Fast As Possible, Empower The Team, Built Integrity, dan See The Whole.

### 5.8.1 Prinsip Lean Software Development

Pengembangan perangkat lunak untuk dengan pendekatan Lean Software Development memiliki prinsip dasar dalam pengembangannya. Berikut ini prinsip dasar dari Lean Software Development :

- a. *Eliminate waste* (Mengeliminasi Ketidak Effisienan).

Dalam software development, waste adalah sesuatu yang tidak mempengaruhi kuahtas coding, mengurangi waktu dan usaha dalam menghasilkan coding atau sesuatu yang tidak memberikan nilai bisnis kepada customer. Dengan kata lain, aktifitas apapun itu yang tidak “pay for itself” dalam mengurangi usaha dimanapun didalam sistem.

*Tools: Seeing Waste, Value Stream Mapping.*

- b. *Amplify learning* (Mengamplifikasi pembelajaran)

Untuk para programmer, mengembangkan sebuah sistem yang

memberikan nilai bisnis, mereka harus belajar banyak hal. Diantaranya berupa teknikal, seperti kelebihan dan kekurangan pada beberapa pendekatan yang akan dilakukan pada remote communication dalam .NET (contoh: remoting, COM+,web services,dll). Dan lainnya adalah kebutuhan terkait, seperti memahami apa yang pengguna bisnis benar-benar butuhkan dan apa yang pengembang pikirkan untuk kebutuhan pengguna.

*Tools: Feedback, Iterations, Synchronization, Set-based Development.*

- c. *Decide as late as possible* (Menentukan keterlambatan sebagai hal yang mungkin).

Gagasan itu adalah menunggu sampai apa yang disebut dengan istilah “the last responsible moment” untuk membuat sebuah keputusan. Dimana, jika tim tidak membuat keputusan, keputusan akan dibuat untuk mereka. Manfaatnya adalah untuk menghindari atau menunda biaya perubahan, yang tidak jelas, yang dapat terjadi jika belum pernah membatasi pilihan.

*Tools: Options Thinking, The Last Responsible Moment, Making Decisions.*

- d. *Deliver as fast as possible* (Mengantarkan secara cepat sebagai hal yang mungkin).

Merupakan hal dasar dari development berulang. Kebutuhan berubah sebagai sebuah persentase dari kebutuhan original yang meningkatkan non linear sebagai jumlah waktu yang meningkat. Secara umum proyek dengan waktu 9-12 bulan memiliki kira-kira 25% perubahan kebutuhan. Dimana tiap bulannya perubahan kebutuhan dalam rata-rata hanya 1-2persen. Dan hal ini lebih mudah disetujui.

*Tools: Pull Systems, Queuing Theory, Cost of Delay.*

- e. *Empower the team* (Memberdayakan team).

Menguatkan tim. Kualitas sebuah tim perangkat lunak (faktor

manusia) merupakan unsur yang paling penting dalam keberhasilan dalam menghasilkan sebuah software.

*Tools: Self Determination, Motivation, Leadership, Expertise.*

f. *Build integrity in* (Membangun integritas).

Membangun integritas. Integritas terbagi antara integritas *perceived* dan integritas konseptual. Integritas *perceived* adalah pengalaman pelanggan perangkat lunak dalam menggunakan software yang dibuat. Integritas konseptual adalah ukuran seberapa baiknya arsitektur dan komponen sistem yang mengalir bersama-sama mengenai integritas *perceived*. Pengujian, satuan dan integrasi, merupakan bagian utama dari integritas.

*Tools: Perceived Integrity, Conceptual Integrity, Refactoring, Testing.*

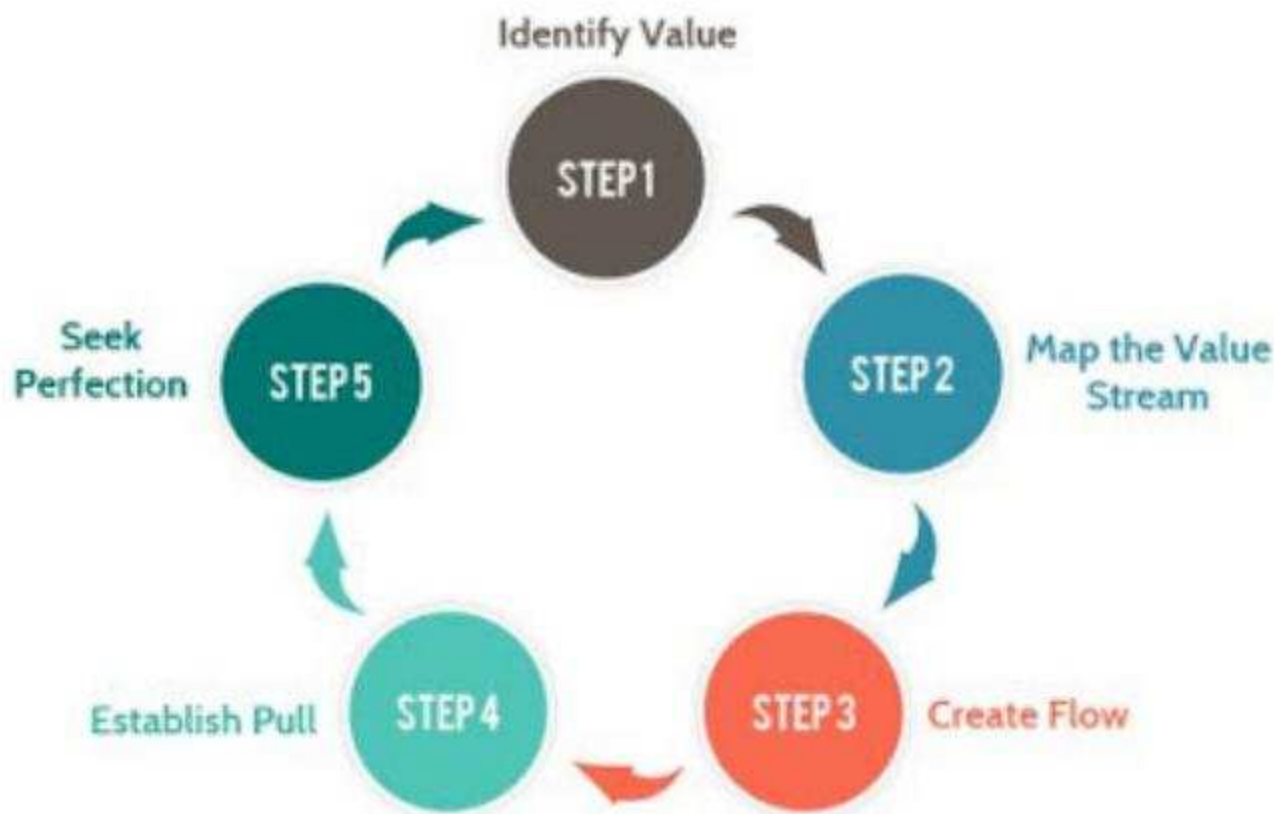
g. *See the whole* (Melihat secara kesatuan/keseluruhan).

Pemikiran sistem sudah ada untuk sementara waktu, tapi respon khas untuk memecahkan masalah adalah untuk memprosesnya menjadi bagian-bagian penyusunnya dan mengoptimalkan setiap bagian individu. Ini adalah sub optimisasi, yang mengarah ke “*tragedy of the commons*.”

*Tools : Measurements, Contracts*

### 5.8.2 Tahapan Lean Software Development

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, Berikut ini merupakan proses atau tahapan Lean Software Development (Tatvasoft, 2015):



- a. Identify Value (Step 1).  
Pada tahapan ini bertujuan untuk Menentukan nilai dari perspektif pelanggan dan nilai ekspres dalam hal pengembangan perangkat lunak atau layanan tertentu.
- b. Map the Value Stream (Step 2).  
Setelah nilai (tujuan) ditentukan, langkah selanjutnya adalah memetakan “aliran nilai” atau semua langkah dan proses yang terlibat dalam mengambil produk tertentu dari masukan dan mengirimkan produk akhir ke pelanggan. Idennya adalah untuk menggambar disatu halaman, dalam sebuah “peta” aliran bahan/ produk melalui proses tersebut. Tujuannya adalah untuk mengidentifikasi setiap langkah yang tidak menciptakan nilai dan kemudian menemukan cara untuk menghilangkan langkah-langkah yang tidak berguna.
- c. Create Flow (Step 3).  
Setelah sampah dibersihkan dari maps value stream, langkah selanjutnya adalah memastikan bahwa proses pengembangan sudah tidak terdapat gangguan, penundaan, atau permasalahan

lain. Pada tahapan ini memerlukan pemecahan permasalahan yang terjadi dan upaya mencari solusi dari permasalahan.

d. Estabilsh Pull (Step 4).

Tahapan ini tidak ada yang dilakukan oleh proses upstream sampai pelanggan downstream mengisyaratkan sebuah kebutuhan, permintaan aktual menarik produk / layanan melalui value stream.

e. Seek Perfection (Step 5).

Penghapusan sampah secara menyeluruh sehingga semua kegiatan menciptakan nilai bagi pelanggan dengan terobosan dan proyek perbaikan berkelanjutan.

Pemakaian LSD tergantung pada karakteristik organisasi dan proyek yang akan dibangun. Adapun karakteristik organisasi yang cocok dalam memakai LSD adalah manufaktur, product development, supply chain development dan back-office operation.

Karakteristik proyek yang cocok memakai LSD yaitu pembangunan software yang memakan waktu sedikit namun produktifitas tinggi dan proyek yang mementingkan kepuasan customer.

### 5.8.3 Kelebihan Lean Software Development

Berikut ini merupakan kelebihan metode Lean Software Development antara lain :

- a. Mengeliminasi ketidak efisienan membantu mempercepat proses, menghemat.
- b. Sumberdaya dan efisiensi.
- c. Membantu memberikan produk lebih awal.
- d. Kekuatan tim membantu dalam membuat keputusan antara tim dan memotivasi tim.
- e. Proyek sangat bergantung pada tim yang saling bekerjasama dan berkomitmen.

#### **5.8.4 Kelemahan Lean Software Development**

Selain memiliki banyak kelebihan metode Lean Software Development ini memiliki kelemahan. Berikut ini merupakan kelemahan dari metode LSD ini :

- a. Time limit dapat berubah-ubah (tidak sesuai jadwal jika ada perubahan).
- b. Kebutuhan biaya dapat membengkak jika semakin banyak perubahan yang diinginkan (karena kelebihan waktu ditanggung oleh client).

#### **5.9 Rational Unified Process (RUP)**

Rational Unified Process sering disingkat dengan (RUP) adalah suatu kerangka kerja proses pengembangan perangkat lunak iteratif yang dibuat oleh Rational Software, yaitu suatu divisi dari IBM sejak 2003. Rational Unified Process (RUP) adalah proses pengembangan perangkat lunak untuk model berorientasi objek. Ia juga dikenal sebagai Model Proses Terpadu. RUP Meningkatkan produktivitas tim untuk memberi setiap anggota tim akses mudah ke basis pengetahuan dengan pedoman, template, dan pembimbing alat untuk semua kegiatan pengembangan penting.

RUP merupakan salah satu model pengembangan perangkat lunak berulang, yang memberikan tugas dan tanggung jawab dalam satu organisasi untuk memastikan produksi perangkat lunak berkualitas tinggi yang berarti dapat memenuhi kebutuhan pengguna dengan jadwal dan anggaran yang dapat diprediksi (Bassil, 2012).

Pada RUP anggota tim yang terlibat dalam proyek dipandu untuk mengikuti setiap langkah yang didasarkan pada setiap elemen. Terdapat tiga elemen dasar pada kerangka kerja RUP, yaitu: kegiatan, peran dan artefak. Pemilihan ketiga elemen tersebut harus disesuaikan dengan proyek perangkat lunak. Setiap proyek berisi sekelompok anggota tim yang bertugas terhadap satu atau lebih peran. Setiap peran

berpartisipasi terhadap satu atau lebih kegiatan dan setiap kegiatan menghasilkan satu atau lebih artefak (Karambir & Thind, 2015).

Pada pengembangan industri perangkat lunak, RUP mendukung penggunaan tim kecil atau tim besar. Terdapat berbagai ukuran pengembangan proyek perangkat lunak, yaitu proyek dengan skala kecil, proyek skala menengah dan proyek skala besar.

#### **5.9.1 Karakteristik Rational Unified Process**

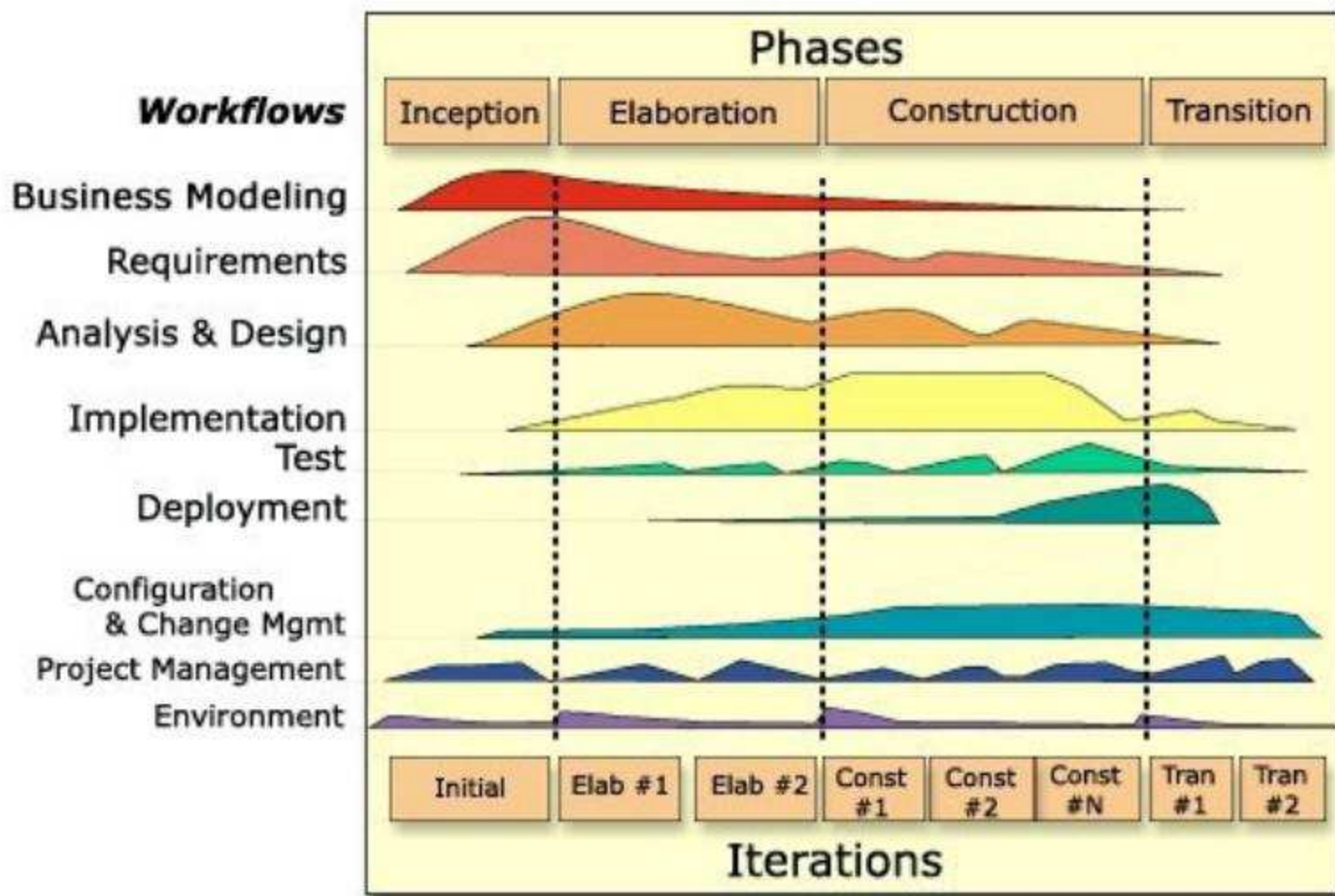
Sebelum membahas lebih lanjut tentang tahap pengembangan dengan metode RUP. Perlu di ketahui terlebih dahulu tentang karakteristik dari metode ini. Berikut ini karakteristik dari Metode RUP:

- a. Berulang (*iterative*).  
Tahap pengembangan untuk setiap produk yang diserahkan (*release*) dilaksanakan secara berulang.
- b. Architecture centric  
Menggunakan arsitektur sistem sebagai artifak utama untuk konseptualisasi, konstruksi, pengelolaan dan penyusunan sistem selama pengembangan.
- c. Use case-driven  
Menggunakan *use case* sebagai artifak utama untuk menetapkan perilaku sistem yang diinginkan dan untuk mengkomunikasikan perilaku sistem tersebut kepada parastakeholder sistem.
- d. Risk-driven  
Menghilangkan atau mengurangi risiko-risiko yang dapat menghambat kesuksesan proyek.

#### **5.9.2 Fase Dalam Rational Unified Process**

Pada RUP didefinisikan terdapat empat fasa siklus proyek. Fasa-fasa ini memungkinkan untuk disajikan dalam bentuk umum mirip dengan pendekatan air terjun, walaupun esensi kunci dari proses

terdapat dalam iterasi dalam setiap fasenya. Setiap fase memiliki sebuah objektif kunci dan titik pencapaian akhir yang menandakan ketercapaian objektif. Berikut ini tahapan pengembangan perangkat lunak dengan metode RUP :



a. Inception

Tahap untuk mengidentifikasi sistem yang akan dikembangkan. Dalam fase ini pengembang perangkat lunak dituntut untuk bisa melakukan interaksi dengan pelanggan, sebagai langkah awal untuk pengidentifikasian kebutuhan-kebutuhan sistem yang hendak dibuat. Langkah ini cukup penting agar para pengembang perangkat lunak punya kesamaan persepsi antara sistem yang akan dibuat dengan kebutuhan pengguna. Fase ini berfokus pada cakupan dan tujuan dari proyek. Tahap-tahap iterasi kerja yang dilakukan developer pada fase ini adalah sebagai berikut :

- 1) *Business Modeling and Requirements*, menganalisa, merumuskan, dan menentukan perencanaan, cakupan dan

kebutuhan utama bisnis.

- 2) *Analysis*, mengadakan studi kelayakan terhadap proyek yang akan dijalani.
- 3) *Design*, mendesain konsep atau prototype teknisnya.
- 4) *Implementation*, membuat prototype konsepnya; dan
- 5) *Test*, tahap ini belum diperlukan pada fase ini.

**b. Elaboration.**

Elaboration merupakan tahap untuk melakukan desain secara lengkap berdasarkan hasil analisis pada tahap inception. Fase ini belum masuk ke tahap pembuatan perangkat lunak secara langsung, tetapi lebih kepada pemantapan konsep dari peninjauan kembali terhadap rencana-rencana yang sudah ditentukan sebelumnya. Dengan demikian diharapkan proyek yang berjalan, risikonya dapat ditekan seminimal mungkin. Fase ini berfokus pada requirement yang didapat dan menentukan strukturisasi sistem. Pada fase ini tahap iterasi kerja yang dilakukan developer pada fase ini adalah :

- 1) *Business Modeling dan Requirements*, memperbaiki cakupan dari kebutuhan sistem.
- 2) *Analysis*, menganalisa kebutuhan sistem dan cara membangun sistem tersebut.
- 3) *Design*, membuat arsitektur yang baik.
- 4) *Implementation*, membuat garis besar arsitektur
- 5) *Test*, melakukan test atau pengujian garis besar arsitektur yang sudah dibuat.

**c. Construction**

*Construction* merupakan tahap untuk mengimplementasikan hasil desain dan melakukan pengujian hasil implementasi. Pada tahap awal construction, ada baiknya dilakukan pemeriksaan ulang hasil analisis dan desain terutama desain pada sequence diagram, class

diagram, component dan deployment. Apabila desain yang dibuat telah sesuai dengan analisis sistem, maka implementasi dengan bahasa pemrograman tertentu dapat dilakukan. Fase ini memiliki tujuan utama membangun sistem perangkat lunak. Tahap-tahap iterasi kerja yang dilakukan developer pada fase ini adalah :

- 1) *Business Modeling dan Requirements*, menganalisa lebih lanjut kebutuhan-kebutuhan proyek yang mungkin belum terpikirkan sebelumnya.
- 2) *Analysis*, menyelesaikan analisis model.
- 3) *Design*, menyelesaikan desain model.
- 4) *Implementation*, membangun *Initial Operational Capability*.
- 5) *Test*, melakukan pengetesan terhadap *Operational Capability* yang telah dibuat.

#### **d. Transition**

Tahap ini dilakukan untuk mematangkan produk akhir yang sudah jadi, hal ini diperlukan untuk menganalisa apakah perangkat lunak sudah dibuat sesuai dengan kebutuhan pengguna atau mungkin terdapat kesalahan atau kekurangan yang perlu diperbaiki. fase ini berhubungan dengan instalasi dan rollout. Tahap-tahap iterasi kerja yang dilakukan developer pada fase ini adalah sebagai berikut :

- 1) *Business Modeling dan Requirement*, tahapan ini seharusnya sudah tidak digunakan lagi karena pada fase ini merupakan fase akhir, tetapi tetap dapat dilakukan jika memang masih dibutuhkan.
- 2) *Analysis*, tahapan ini seharusnya sudah selesai di fase sebelumnya sehingga tidak digunakan lagi, tetapi tidak menutup kemungkinan masih dapat digunakan jika masih dibutuhkan.
- 3) *Design*, melakukan modifikasi terhadap desain sistem jika ditemukan masalah selama testing.
- 4) *Implementation*, melakukan penyesuaian setting perangkat lunak agar bisa dipakai disisi pengguna (misalnya install dan

setting database di server pengguna, penyesuaian setting IP) dan melakukan perbaikan coding yang ditemukan selama testing dilakukan.

- 5) *Test*, melakukan proses testing perangkat lunak dan testing akhir pengguna.

### 5.9.3 Kelebihan Rational Unified Process

Berikut ini adalah beberapa keuntungan/kelebihan dengan menggunakan metode RUP dalam pengembangan perangkat lunak diantaranya :

- a. Menyediakan akses yang mudah terhadap pengetahuan dasar bagi anggota tim.
- b. Menyediakan petunjuk bagaimana menggunakan UML secara efektif.
- c. Mendukung proses pengulangan dalam pengembangan software.
- d. Memungkinkan adanya penambahan-penambahan pada proses.
- e. Memungkinkan untuk secara sistematis mengontrol perubahan.
  - perubahan yang terjadi pada software selama proses pengembangannya.
- f. Memungkinkan untuk menjalankan test case dengan menggunakan Rational Test Manager Tool.

### 5.9.4 Kekurangan Rational Unified Process

Metodologi ini hanya dapat digunakan pada pengembangan perangkat lunak yang berorientasi objek dengan berfokus pada UML (Unified Modeling Language). Cara kerja RUP itu didasarkan pada 6 kunci prinsip bagi perkembangan bisnis yang terkendali yaitu :

- a. Proses penyesuaian terhadap lingkungan pekerjaan.
- b. Menyeimbangkan pengutamaan dari para stakeholders.
- c. Melakukan kerjasama antar tim.
- d. Mendemonstrasikan hasil-hasil yang ada secara berulang-ulang.

- e. Menaikkan tingkat abstraksi dari sebuah software.
- f. Pemusatan pada kualitas secara terus-menerus.

### 5.10 Scrum

Scrum adalah suatu metodologi atau kerangka kerja yang terstruktur untuk mendukung pengembangan produk yang kompleks. Scrum terdiri dari sebuah tim yang memiliki peran dan tugas masing-masing. Setiap komponen dalam kerangka melayani tujuan tertentu dan sangat penting untuk kesuksesan penggunaan scrum. SCRUM adalah salah satu metode rekayasa perangkat lunak dengan menggunakan prinsip-prinsip pendekatan AGILE, yang bertumpu pada kekuatan kolaborasi tim, incremental product dan proses iterasi untuk mewujudkan hasil akhir.

Proses pengembangan menggunakan metode Scrum terdapat lima tahapan pengembangan yaitu: (1) backlog refinement, (2) sprint planning, (3) daily meeting, (4) reviews, dan (5) sprint retrospective. Kelima proses pengembangan tersebut mengikuti tiga prinsip Scrum yaitu: product owner (PO), Scrum master (SM), dan cross functional (Alqudah & Razali, 2016).

Metode Scrum dalam proses pengerjaan sebuah proyek mengedepankan sprint, dimana kondisi ini telah terjadi ketika pertama kali Metode Scrum digunakan dalam proses pengembangan pada tahun 1990 (Robiansyah & Angreani, 2017). Sprint didalam metode Scrum adalah proses pengerjaan pada tiap tahapan. Dimana proses pengerjaan di dalam sprint Scrum membutuhkan waktu yang sama untuk masing-masing sprint yaitu lebih kurang tiga puluh hari pengerjaan.

Sedangkan jenis dari sprint yang dilakukan pada proses pengerjaan Scrum terdiri dari (1) sprint planning, (2) daily scrum, (3) sprint review, dan (4) sprint retrospective (Schwaber & Sutherland, 2012). Scrum juga memiliki kelebihan selain dari kecepatan,

kelebihan tersebut yaitu dalam proses pengembangan selalu dilakukan pengecekan dan perubahan yang diperlukan sesuai dengan kebutuhan dan teknologi yang digunakan (Ependi, 2018).

#### 5.10.1 Tim Scrum

Scrum Team terdiri dari Product Owner, Development Team dan Scrum Master. Scrum Team bersifat swakelola dan lintas-fungsi. Tim memilih cara terbaik dalam mengerjakan pekerjaan mereka, bukan diperintah oleh orang lain di luar tim ini. Tim yang lintas-fungsi memiliki semua keahlian yang dibutuhkan untuk menyelesaikan pekerjaan mereka tanpa bergantung pada orang lain di luar tim ini.

Bentuk tim dalam Scrum dirancang untuk mengoptimalkan fleksibilitas, kreativitas dan produktivitas. Bentuk Scrum Team telah terbukti menjadikan tim semakin efektif dalam mengerjakan semua tipe pekerjaan yang telah disebutkan sebelumnya dan untuk jenis pekerjaan kompleks apa pun.

a. Product Owner.

Product Owner adalah orang yang bertanggung jawab untuk memaksimalkan nilai bisnis dari produk yang dihasilkan oleh Development Team. Cara melakukannya sangat bervariasi antar organisasi, Scrum Team dan individu.

b. Development Team.

Development Team terdiri dari para ahli profesi yang bekerja untuk menghantarkan Increment “Selesai” yang berpotensi untuk dirilis di setiap akhir Sprint. Increment “Selesai” wajib tersedia pada saat Sprint Review. Hanya anggota dari Development Team yang membuat Increment ini. Development Team dibentuk dan diberikan wewenang oleh organisasi untuk menyusun dan mengelola pekerjaan mereka sendiri. Hasil sinergi dari tim ini akan mengoptimalkan efisiensi dan efektivitas Development Team secara keseluruhan.

c. Scrum Master.

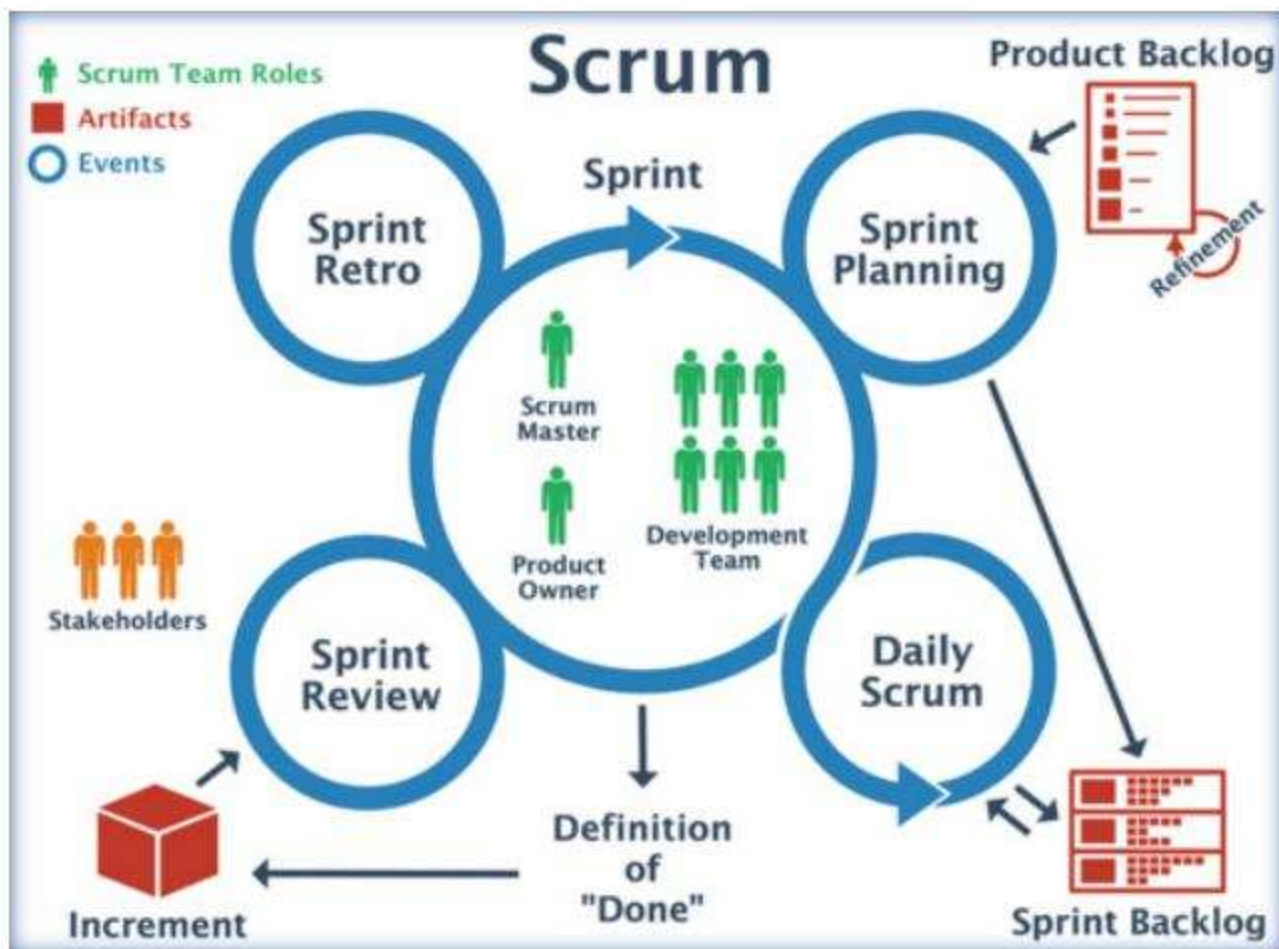
Scrum Master bertanggung jawab untuk mengenalkan dan menyokong penggunaan Scrum sebagaimana dijelaskan di dalam Panduan Scrum ini. Scrum Master melakukan ini dengan membantu orang-orang agar dapat memahami teori, praktik-praktik, aturan-aturan dan tata nilai Scrum. Scrum Master adalah pemimpin yang melayani Scrum Team. Scrum Master membantu orang-orang di luar Scrum Team untuk dapat memahami interaksi mana yang bermanfaat dan tidak bermanfaat. Scrum Master membantu orang-orang untuk mengubah interaksi ini guna memaksimalkan nilai bisnis yang dihasilkan oleh Scrum Team.

#### **5.10.2 Aktifitas Dalam Scrum**

Aktivitas-aktivitas wajib dalam scrum dihadiri untuk menciptakan sebuah kesinambungan dan mengurangi adanya aktivitas-aktivitas lain yang tidak tercantum di dalam scrum. Setiap acara di dalam Scrum memiliki batasan waktu, yang artinya selalu memiliki durasi maksimum.

Pada saat sprint dimulai, durasinya tetap dan tidak dapat diperpendek maupun diperpanjang. Acara-acara lainnya dapat diakhiri saat tujuan dari acara tersebut telah tercapai, memastikan waktu digunakan secukupnya tanpa ada yang terbuang sia-sia di sepanjang proses. Selain sprint itu sendiri, yang memang merupakan kontainer dari acara-acara lain, setiap acara dalam scrum adalah sebuah kesempatan formal untuk meninjau dan merubah sesuatu.

Acara-acara ini dirancang secara khusus untuk menciptakan transparansi dan peninjauan sampai ke tingkatan kritis. Tidak adanya pelaksanaan salah satu acara ini akan mengurangi transparansi dan menghilangkan kesempatan untuk meninjau dan membuat perubahan.



a. Sprint

Jantung dari scrum adalah sprint, sebuah batasan waktu selama satu bulan atau kurang, di mana sebuah potongan produk yang “Selesai”, berfungsi, berpotensi untuk dirilis dikembangkan. Sprint biasanya memiliki durasi yang konsisten sepanjang proses pengembangan produk. Sprint yang baru, langsung dimulai setelah sprint yang sebelumnya berakhir. Sprint memuat dan terdiri dari sprint planning, daily scrum, pengembangan, sprint review, dan sprint retrospective.

b. Sprint Planning.

Pekerjaan yang akan dilaksanakan di dalam sprint direncanakan pada saat sprint planning. Perencanaan ini dibuat secara kolaboratif oleh seluruh anggota tim. Sprint Planning dibatasi maksimum delapan jam untuk sprint yang berdurasi satu bulan. Untuk sprint yang lebih pendek, batasan waktunya biasanya lebih singkat. Scrum Master memastikan bahwa acara ini dilaksanakan dan setiap hadirin memahami tujuannya. Scrum

Master mengedukasi tim untuk melaksanakannya dalam batasan waktu yang telah ditentukan.

c. Daily Scrum.

Merupakan kegiatan dengan batasan waktu maksimum selama 15 menit agar Tim Pengembang dapat mensinkronisasikan pekerjaan mereka dan membuat perencanaan untuk 24 jam ke depan. Hal ini dilakukan dengan meninjau pekerjaan semenjak acara daily scrum terakhir dan memperkirakan pekerjaan yang dapat dilakukan sebelum melakukan daily scrum berikutnya.

d. Sprint Review.

Aktivitas yang diadakan di akhir sprint untuk meninjau potongan produk dan merubah product backlog bila diperlukan. Pada saat sprint review. Tim Scrum dan stakeholder berkolaborasi untuk membahas apa yang telah dikerjakan dalam sprint yang baru usai. Berdasarkan hasil tersebut tersebut dan semua perubahan product backlog pada saat sprint, para hadirin berkolaborasi menentukan apa yang dapat dikerjakan di sprint berikutnya untuk mengoptimalisasi nilai produk.

Pertemuan ini bersifat informal, bukan merupakan status meeting, dan presentasi dari potongan produk diharapkan dapat mengumpulkan masukan dan menumbuhkan semangat kolaborasi.

e. Sprint Retrospective.

Sebuah kesempatan bagi tim untuk meninjau dirinya sendiri dan membuat perencanaan mengenai peningkatan yang akan dilakukan di sprint berikutnya. Sprint Retrospective dilaksanakan setelah sprint review selesai dan sebelum sprint planning berikutnya. Ini adalah acara dengan batasan waktu maksimum selama tiga jam untuk sprint yang berdurasi satu bulan. Untuk sprint yang lebih pendek, batasan waktunya biasanya lebih singkat. Scrum Master memastikan bahwa acara

ini dilaksanakan dan setiap hadirin memahami tujuannya. Scrum Master mengedukasi Tim Scrum untuk melaksanakannya dalam batasan waktu yang telah ditentukan. Scrum Master berpartisipasi sebagai rekan yang bertanggung-jawab terhadap proses scrum.

### 5.10.3 Artefak Scrum

Artefak scrum merepresentasikan pekerjaan atau nilai, bertujuan untuk menyediakan transparansi, dan kesempatan-kesempatan untuk peninjauan dan adaptasi. Artefak yang didefinisikan oleh scrum secara khusus dirancang untuk meningkatkan transparansi dari informasi kunci, dengan begitu semua pihak dapat memiliki pemahaman yang sama terhadap artefak.

a. Product Backlog.

Merupakan daftar terurut, dari setiap hal yang berkemungkinan dibutuhkan di dalam produk, dan juga merupakan sumber utama, dari daftar kebutuhan mengenai semua hal yang perlu dilakukan terhadap produk. Product Owner bertanggung-jawab terhadap product backlog, termasuk isinya, ketersediaannya, dan urutannya.

b. Sprint Backlog.

Merupakan sekumpulan item di dalam product backlog yang telah dipilih untuk dikerjakan di sprint, juga di dalamnya rencana untuk mengembangkan potongan tambahan produk dan merealisasikan sprint goal Sprint backlog adalah perkiraan mengenai fungsionalitas apa yang akan tersedia di Increment selanjutnya dan pekerjaan yang perlu dikerjakan untuk menghantarkan fungsionalitas tersebut menjadi potongan tambahan produk yang “Selesai”.

c. Increment.

Increment (tambahan potongan produk) adalah gabungan dari semua item Product Backlog yang diselesaikan pada Sprint

berjalan dan nilai-nilai dari increment sprint-sprint sebelumnya. Pada akhir Sprint, increment terbaru harus “Selesai” yang artinya berada dalam kondisi yang berfungsi penuh dan memenuhi definisi “Selesai” yang dibuat oleh tim. Terlepas apakah Product Owner akan merilis produknya, produk harus selalu berada dalam kondisi yang berfungsi penuh.

#### **5.10.4 Kelebihan Metode Scrum**

Berikut ini adalah beberapa kelebihan dari SCRUM, antara lain:

- a. SCRUM dapat membantu perusahaan Anda dalam menghemat waktu dan biaya (dalam hal ini uang). Biaya overhead dari proses dan manajemen sangat minim sehingga dapat mengarahkan kita kepada hasil yang lebih cepat dan lebih murah.
- b. Dengan menggunakan metode SCRUM, Anda dapat mentransformasikan bisnis yang sulit untuk diukur menjadi mudah untuk dikembangkan.
- c. Pada metode SCRUM, pergerakan pengembangan cutting edge dapat dengan cepat dikodekan dan diuji menggunakan metode ini. Bagaikan kesalahan yang mudah untuk diperbaiki.
- d. Dengan menggunakan SCRUM, Anda dapat mengontrol dan memonitoring aktivitas peningkatan dan penurunan beban pekerjaan yang bisa terjadi kapan saja.
- e. Seperti metodologi agile pada umumnya, SCRUM merupakan metode iterative yang membutuhkan feedback secara berkelanjutan dari user atau pengguna.
- f. Dengan adanya short sprint dan constant feedback, SCRUM dapat dengan mudah mengatasi setiap perubahan yang terjadi.
- g. Dengan adanya daily scrum meeting, memungkinkan SCRUM untuk mengukur produktivitas individu, hal ini mengarah pada peningkatan produktivitas dari setiap anggota tim.
- h. Dengan SCRUM, setiap ada masalah yang timbul dapat di

identifikasi dengan baik pada pertemuan harian dan oleh karena itu setiap masalah dapat di selesaikan dengan cepat.

- i. Dengan menggunakan metode SCRUM, Anda dapat dengan mudah untuk mengirim produk berkualitas sesuai dengan waktunya.
- j. SCRUM dapat bekerja dengan berbagai teknologi dan bahasa pemrograman. Namun secara khusus berguna untuk pengembangan proyek dengan teknologi web 2.0 ataupun media proyek baru lainnya.

#### **5.10.5 Kelemahan Metode Artefak Scrum**

Selain memiliki banyak kelebihan Scrum sendiri juga memiliki kelemahan. Berikut ini kelemahan Scrum antara lain :

- a. SCRUM bisa menjadi salah satu penyebab utama terjadinya scope creep, kecuali ada tanggal akhir tertentu. Stakeholder proyek atau manajemen akan terus menuntut fungsi dan fitur baru untuk disampaikan.
- b. Setiap tugas harus didefinisikan dengan baik, karena hal ini dapat mempengaruhi perkiraan biaya dan waktu pengerjaan proyek. Jika tidak didefinisikan dengan baik maka semua hal tersebut tidak akan akurat. Dalam kasus seperti ini, biasanya tugas dapat tersebar di beberapa sprint.
- c. Jika anggota tim Anda tidak berkomitmen dengan baik, maka proyek Anda tidak akan selesai atau bahkan bisa gagal.
- d. Metode SCRUM ini hanya membutuhkan anggota tim yang sudah berpengalaman, jika tim Anda berisi orang-orang yang masih pemula maka proyek tidak dapat selesai sesuai dengan waktunya.
- e. SCRUM dapat bekerja dengan baik jika seorang Scrum Master dapat mempercayai tim yang mereka kelola. Jika Scrum Master terlalu mengontrol secara ketat, hal ini dapat menyebabkan

tim menjadi tertekan dan stress, sehingga mengakibatkan demoralisasi dan kegagalan dari proyek tersebut.

- f. Jika sering terjadi pergantian anggota tim saat pengembangan proyek berlangsung, hal ini dapat menyebabkan efek yang kurang baik bagi perkembangan proyek tersebut, proyek akan semakin lama selesai dari waktunya.

## **BAB VI**

# **MODEL PENGEMBANGAN SISTEM LAINNYA**

### **6.1 Rapid Application Development**

Rapid application development (RAD) atau rapid prototyping adalah model proses pembangunan perangkat lunak yang tergolong dalam teknik incremental (bertingkat). RAD menekankan pada siklus pembangunan pendek, singkat, dan cepat. Waktu yang singkat adalah batasan yang penting untuk model ini. Rapid application development menggunakan metode iteratif (berulang) dalam mengembangkan sistem dimana working model (model kerja) sistem dikonstruksikan diawal tahap pengembangan dengan tujuan menetapkan kebutuhan (requirement) pengguna. Model kerja digunakan hanya sesekali saja sebagai basis desain dan implementasi sistem akhir.

Rapid Application Development (RAD) adalah metodologi pengembangan perangkat lunak yang berfokus pada membangun aplikasi dalam waktu yang sangat singkat. Istilah ini menjadi kata kunci pemasaran yang umum menjelaskan aplikasi yang dapat dirancang dan dikembangkan dalam waktu 60-90 hari, tapi itu awalnya ditujukan untuk menggambarkan suatu proses pembangunan yang melibatkan application prototyping dan iterative development. Menurut James Martin “Rapid Application Development (RAD) merupakan pengembangan siklus yang dirancang untuk memberikan pengembangan yang jauh lebih cepat dan hasil yang lebih berkualitas tinggi daripada yang dicapai dengan siklus hidup tradisional. Hal ini dirancang untuk mengambil keuntungan maksimum dari

pengembangan perangkat lunak yang telah berevolusi baru-baru ini”.

Profesor Clifford Kettemborough dari College Whitehead, University of Redlands, mendefinisikan Rapid Application Development sebagai “pendekatan untuk membangun sistem komputer yang menggabungkan Computer-Assisted Software Engineering (CASE) tools dan teknik, user-driven prototyping, RAD meningkatkan kualitas sistem secara drastis dan mengurangi waktu yang diperlukan untuk membangun sistem”.

Sebagai gambaran umum, pengembangan aplikasi berarti mengembangkan aplikasi pemrograman yang bervariasi dari pemrograman umum dalam arti bahwa ia memiliki tingkat yang lebih tinggi dari liability, termasuk untuk kebutuhan capturing dan testing.

Pada 1970-an, Rapid Application Development muncul sebagai respon untuk non-agile processes, seperti model Waterfall. Pengembang perangkat lunak menghadapi masalah waktu dengan metodologi sebelumnya sebagai sebuah aplikasi yang begitu lama untuk membangun. Dengan demikian, metodologi tersebut sering mengakibatkan sistem tidak dapat digunakan.

### **6.1.1 Unsur – Unsur Rapid Application Development**

RAD memiliki banyak unsur-unsur yang membuat sebuah metodologi yang unik termasuk prototyping, iterative development, time boxing, team members, management approach, dan RAD tools.

#### **a. Prototyping**

Sebuah aspek kunci dari RAD adalah pembangunan prototipe untuk tujuan membangkitkan kembali desain untuk kebutuhan pengguna. Tujuannya adalah untuk membangun sebuah fitur ringan yang hasil akhirnya dalam jumlah pendek dengan waktu yang memugkiukan. Prototipe awal berfungsi sebagai bukti konsep untuk klien, tetapi lebih penting berfungsi sebagai titik berbicara dan alat untuk kebutuhan pemurnian. Mengembangkan

prototipe cepat dicapai dengan Computer Aided Engineering CASE tools Software yang berfokus pada menangkap persyaratan, mengkonversi mereka ke model data, mengubah model data ke database, dan menghasilkan kode semua dalam satu alat. CASE tools populer di 80-an dan awal 90-an, tetapi sebagai teknologi telah berubah (dan COBOL telah menjadi usang) beberapa alat mengambil keuntungan penuh dari potensi penuh dari teknologi KASUS alat. Perusahaan rasional adalah yang paling terkenal meskipun prototipe potensi pembangkitnya terbatas. Pada Otomatis Arsitektur produk cetak biru kami berfokus pada peningkatan tingkat aplikasi enterprise web yang berfungsi sebagai prototipe karena kecepatan yang mereka dapat diciptakan (dalam menit).

b. Iterative Development.

Iterative Development berarti menciptakan versi yang lebih fungsional dari sebuah sistem dalam siklus pembangunan pendek. Setiap versi ditinjau dengan klien untuk menghasilkan persyaratan untuk membuat versi berikutnya. Proses ini diulang sampai semua fungsionalitas telah dikembangkan. Panjang ideal iterasi adalah antara satu hari (yang lebih dekat dengan Metodologi Agile) dan tiga minggu.

Setiap siklus pengembangan memberikan pengguna kesempatan untuk memberikan umpan balik, memperbaiki persyaratan, dan kemajuan melihat (dalam pertemuan sesi fokus grup). Hal ini akhirnya pembangunan berulang yang memecahkan masalah yang melekat dalam metodologi fleksibel dibuat pada 1970-an.

c. Time boxing.

Time boxing adalah proses menunda fitur untuk versi aplikasi di masa mendatang untuk melengkapi versi saat ini sebagai ketepatan waktu. Ketepatan waktu merupakan aspek penting

dari RAD, karena tanpa itu ruang lingkup dapat mengancam untuk memperpanjang iterasi pembangunan, sehingga membatasi umpan balik dari klien, meminimalkan manfaat dari pembangunan berulang, dan berpotensi mengembalikan proses kembali ke pendekatan metodologi air terjun.

d. Team Member.

Metodologi RAD merekomendasikan penggunaan tim kecil yang terdiri dari anggota yang berpengalaman, serbaguna, dan motivasi yang mampu melakukan peran ganda. Sebagai klien memainkan peran penting dalam proses pembangunan, sumber daya klien khusus harus tersedia selama awal Joint Application Development (JAD) sesi serta Focus Group Sessions dilakukan pada akhir siklus pengembangan. Pengembangan tim (juga dikenal sebagai SWAT atau Skilled Workers with Advance Tools) idealnya harus memiliki pengalaman di Rapid Application Development dan harus memiliki pengalaman dengan Computer Aided Software Engineering. Pendekatan manajemen Aktif dan manajemen yang terlibat sangat penting untuk mengurangi risiko siklus pengembangan diperpanjang, kesalahpahaman klien, dan melebihi tenggat waktu. Di atas manajemen semua harus kuat dan konsisten dalam keinginan mereka untuk menggunakan metodologi Rapid Application Development. Selain menegakkan waktu yang ketat, manajemen harus fokus pada pemilihan anggota tim, motivasi tim, dan pada kliring hambatan birokrasi atau politik.

e. RAD Tools

Salah satu tujuan utama dari metodologi Rapid Application Development yang dikembangkan oleh James Martin pada tahun 1980-an adalah untuk memanfaatkan teknologi terbaru yang tersedia untuk mempercepat pembangunan. Jelas teknologi tahun 1980 sudah kuno, tetapi fokus RAD tentang alat terbaru

adalah sama pentingnya hari ini seperti ketika metodologi awalnya diciptakan.

### 6.1.2 Siklus Rapid Application Development

RAD adalah suatu pendekatan berorientasi objek terhadap pengembangan sistem yang mencakup suatu metode pengembangan serta perangkat-perangkat lunak. RAD bertujuan mempersingkat waktu yang biasanya diperlukan dalam siklus hidup pengembangan sistem tradisional antara perancangan dan penerapan suatu sistem informasi. Pada akhirnya, RAD sama-sama berusaha memenuhi syarat-syarat bisnis yang berubah secara cepat. Terdapat tiga fase dalam RAD yang melibatkan penganalisis dan pengguna dalam tahap penilaian, perancangan, dan penerapan. Adapun ketiga fase tersebut adalah requirements planning (perencanaan syarat-syarat), RAD design workshop (workshop desain RAD), dan implementation (implementasi). Berikut ini adalah tahap-tahap pengembangan aplikasi dari tiap-tiap fase pengembangan aplikasi (Kendall & Kendall, 2010). Siklus RAD dapat dilihat pada Gambar 6.1.



Gambar 6. 1 Siklus RAD

- a. Requirements Planning (Perencanaan syarat-syarat).  
Dalam fase ini, pengguna dan penganalisis bertemu untuk mengidentifikasi tujuan-tujuan aplikasi atau sistem serta untuk mengidentifikasi syarat-syarat informasi yang ditimbulkan dari tujuan-tujuan tersebut. Orientasi dalam fase ini adalah menyelesaikan masalah-masalah perusahaan. Meskipun teknologi informasi dan sistem bisa mengarahkan sebagian dari sistem yang diajukan, fokusnya akan selalu tetap pada upaya pencapaian tujuan-tujuan perusahaan.
- b. RAD Design Workshop (Workshop Desain RAD).  
Fase ini adalah fase untuk merancang dan memperbaiki yang bisa digambarkan sebagai workshop. Penganalisis dan pemrogram dapat bekerja membangun dan menunjukkan representasi visual desain dan pola kerja kepada pengguna. Workshop desain ini dapat dilakukan selama beberapa hari tergantung dari ukuran aplikasi yang akan dikembangkan. Selama workshop desain RAD, pengguna merespon prototipe yang ada dan penganalisis memperbaiki modul-modul yang dirancang berdasarkan respon pengguna. Apabila seorang pengembangnya merupakan pengembang atau pengguna yang berpengalaman, Kendall menilai bahwa usaha kreatif ini dapat mendorong pengembangan sampai pada tingkat terakselerasi.
- c. Implementation (Implementasi).  
Pada fase implementasi ini, penganalisis bekerja dengan para pengguna secara intens selama workshop dan merancang aspek-aspek bisnis dan nonteknis perusahaan. Segera setelah aspek-aspek ini disetujui dan sistem-sistem dibangun dan disaring, sistem-sistem baru atau bagian dari sistem diujicoba dan kemudian diperkenalkan kepada organisasi.  
Dalam bukunya, "Software Engineering: A Practitioner's Approach". Pressman mengatakan bahwa RAD adalah proses model perangkat lunak inkremental yang menekankan siklus pengembangan yang

singkat (Pressman, 2010). Model RAD adalah sebuah adaptasi “kecepatan tinggi” dari model waterfall, di mana perkembangan pesat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen. Jika tiap-tiap kebutuhan dan batasan ruang lingkup proyek telah diketahui dengan baik, proses RAD memungkinkan tim pengembang untuk menciptakan sebuah “sistem yang berfungsi penuh” dalam jangka waktu yang sangat singkat

### **6.1.3 Keuntungan Rapid Application Development**

RAD mempunyai keuntungan yang dapat disimpulkan sebagai berikut:

- a. Sangat berguna dilakukan pada kondisi user tidak memahami kebutuhankebutuhan apa saja yang digunakan pada proses pengembangan perangkat lunak.
- b. RAD mengikuti tahapan pengembangan sistem seperti umumnya, tetapi mempunyai kemampuan untuk menggunakan kembali komponen yang ada (reusable object) sehingga pengembang tidak perlu membuat dari awal lagi dan waktu lebih singkat berkisar antara 60 hari-90 hari.
- c. Karena mempunyai kemampuan untuk menggunakan komponen yang sudah ada dan waktu yang lebih singkat maka membuat biaya menjadi lebih rendah dalam menggunakan RAD.

### **6.1.4 Kelemahan Rapid Application Development**

Beberapa hal yang perlu diperhatikan dalam implementasi pengembangan menggunakan model RAD :

- a. Proyek yang berskala besar, RAD memerlukan sumber daya manusia yang memadai untuk menciptakan jumlah tim yang baik.
- b. RAD menuntut pengembang dan pelanggan memiliki komitmen dalam aktivitas rapid fire yang diperlukan untuk melengkapi

sebuah sistem dalam waktu yang singkat. Jika komitmen tersebut tidak ada maka proyek RAD akan gagal.

## **6.2 Metode Spiral**

Spiral model adalah salah satu bentuk evolusi yang menggunakan metode iterasi natural yang dimiliki oleh model prototyping dan digabungkan dengan aspek sistematis yang dikembangkan dengan model waterfall. Metode ini baru ditemukan pada tahun 1988 oleh Barry Boehm dalam artikelnya yang berjudul *A Spiral Model of Software Development and Enchanement*.

Tahap desain umumnya digunakan pada model Waterfall, sedangkan tahap prototyping adalah suatu model dimana software dibuat prototype (incomplete model), dan contohnya yang ditunjukkan ke user / customer untuk mendapatkan feedback-nya. Jika prototype-nya sudah sesuai dengan keinginan user / customer, maka proses SE dilanjutkan dengan membuat produk sesungguhnya dengan menambah dan memperbaiki kekurangan dari prototype tadi. Model spiral adalah generator model proses berbasis risiko yang digunakan untuk memandu rekayasa perangkat lunak bersama berbagai pemangku kepentingan dari sistem intensif. Model ini memiliki dua fitur pembeda utama. Salah satunya adalah pendekatan siklis untuk menumbuhkan tingkat definisi dan implementasi sistem secara bertahap sambil mengurangi tingkat risikonya. Yang lainnya adalah serangkaian tonggak jangkar poin untuk memastikan komitmen pemangku kepentingan untuk solusi sistem yang layak dan saling memuaskan (Pressman, 2010).

### **6.2.1 Karakteristik Metode Spiral**

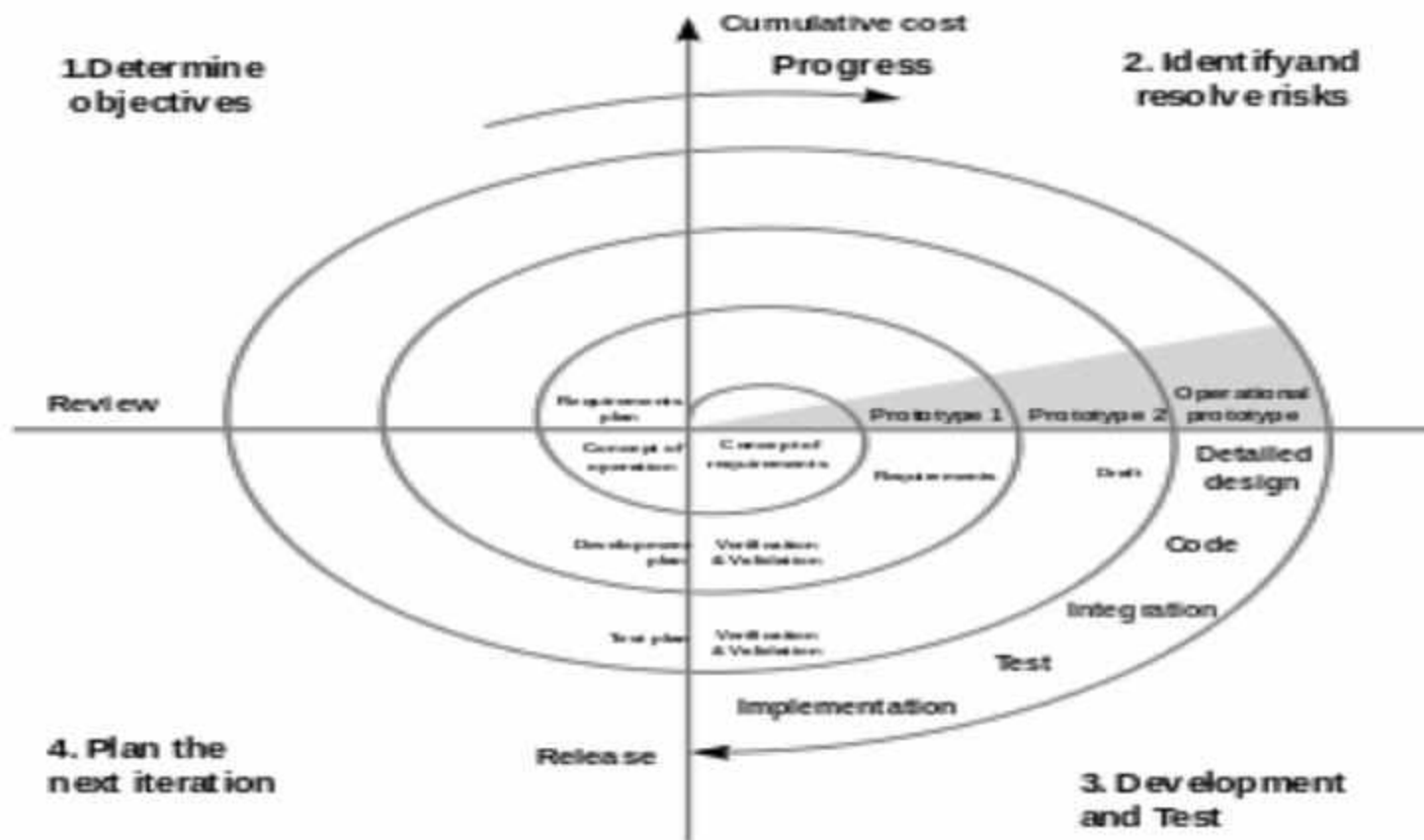
Model spiral merupakan model proses perangkat lunak evolusioner yang memadukan sifat iteratif dari prototyping dengan aspek terkontrol dan sistematis dari model waterfall. Hal ini memberikan potensi untuk pengembangan versi perangkat lunak yang semakin lengkap. Berikut

ini merupakan karakteristik dari metode Spiral:

- Menggunakan prinsip iterasi, namun dalam setiap kali iterasi diperhitungkan dengan “manajemen resikonya”.
- Pada setiap siklus (iterasi) dinilai bagaimana status proyek saat ini, apakah sesuai dengan tujuan (objectives) semula.
- Mempertimbangkan risiko-risiko apa yang dapat muncul bila diadakan perubahan pada suatu iterasi dan melihat alternative apa saja yang tersedia dan menilai dampaknya bagi proyek.
- Tahap iterasi berikutnya harus menitikberatkan pada penanggulangan resiko-resiko.
- Setiap iterasi ditutup dengan pengekskusion rencana.

### 6.2.2 Tahapan pada Metode Spiral

Metode ini berpotensi untuk pengembangan versi pertambahan perangkat lunak secara cepat. Di dalam model spiral, perangkat lunak dikembangkan didalam suatu deretan pertambahan. Selama awal iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap. Beri petunjuk gambar seperti format di gambar 6.1.



Gambar 6.2. Siklus Metode Spiral

a. Communication.

Langkah ini merupakan tahap awal, dimana penulis akan mengumpulkan data untuk mendefinisikan aplikasi yang dibutuhkan oleh pengguna, pada tahap ini dilakukan komunikasi umum seputar aplikasi antara pengembang dan pengguna aplikasi. Dengan adanya tahap ini diharapkan penulis sudah memiliki gambaran umum tentang aplikasi yang akan dibangun, sebagai dasar dalam melakukan proses berikutnya, yaitu proses planning.

b. Planning.

Pada tahap ini penulis melakukan perencanaan pembangunan perangkat lunak, perencanaan yang dilakukan seperti menentukan sumberdaya yang dibutuhkan dalam membangun aplikasi, membuat perkiraan waktu pengerjaan, kemudian menetapkan standar yang ingin dicapai.

c. Modelling.

Pada tahap ini dilakukan proses pemodelan aplikasi yang akan dibangun berdasarkan data maupun bahan yang telah dikumpulkan pada tahap sebelumnya dan menggambarkan sistem yang akan dibangun secara logikal.

d. Construction.

Pada tahap ini dilakukan pengkodean dengan mengacu pada rancangan yang dihasilkan pada tahap modelling. Setelah pengkodean selesai maka akan dilakukan pengujian untuk memastikan bahwa hasilnya sesuai dengan rancangan dan tidak ada kesalahan yang fatal.

e. Deployment.

Pada tahap ini akan dilakukan proses delivery aplikasi kepada pengguna untuk selanjutnya memperoleh feedback dari pengguna tentang aplikasi yang telah dibangun sebagai bahan evaluasi untuk proses pengembangan aplikasi berikutnya.

Model spiral menggabungkan penghindaran perubahan dengan toleransi perubahan. Ini mengasumsikan bahwa perubahan adalah hasil dari risiko proyek dan termasuk kegiatan manajemen risiko yang jelas untuk mengurangi risiko ini. Setiap perulangan pada spiral terbagi menjadi empat bagian :

a. Objective settings.

Menentukan tujuan dari fase yang ditentukan. Batasan-batasan pada proses dan produk sudah diketahui. Perencanaan sudah disiapkan. Risiko dari proyek sudah diketahui. Alternatif strategi sudah disiapkan berdasarkan risiko-risiko yang diketahui, dan sudah direncanakan.

b. Risk assessment and reduction.

Setiap risiko dianalisis secara detil pada sektor ini. Langkah-langkah penanganan dilakukan, misalnya membuat prototype untuk mengetahui ketidakcocokan kebutuhan.

c. Development and Validation.

Setelah evaluasi risiko, maka model pengembangan sistem dipilih. Misalnya jika risiko user interface dominan, maka membuat prototype user interface. Jika bagian keamanan yang bermasalah, maka menggunakan model formal dengan perhitungan matematis, dan jika masalahnya adalah integrasi sistem model waterfall lebih cocok.

d. Planning.

Proyek dievaluasi atau ditinjau-ulang dan diputuskan untuk terus ke fase loop selanjutnya atau tidak. Jika melanjutkan ke fase berikutnya rencana untuk fase selanjutnya.

Perbedaan utama antara model spiral dan model proses perangkat lunak lain adalah pengenalan eksplisit dari risiko. Sebuah siklus spiral dimulai dengan mengelaborasi tujuan seperti kinerja dan fungsi. Cara-cara alternatif untuk mencapai tujuan tersebut, dan berurusan dengan kendala pada masing-masing dari mereka kemudian dicacah.

Setiap alternatif dinilai terhadap masing-masing tujuan dan sumber risiko dari proyek diidentifikasi.

Langkah berikutnya adalah untuk mengatasi risiko ini dengan kegiatan pengumpulan informasi seperti analisis yang lebih rinci, prototyping dan simulasi. Setelah risiko telah dinilai, beberapa pengembangan dilakukan, diikuti dengan kegiatan perencanaan untuk tahap berikutnya dari proses.

Secara informal, risiko hanya berarti sesuatu yang bisa menjadi salah. Misalnya, jika tujuannya adalah untuk menggunakan bahasa pemrograman baru, risiko adalah bahwa para penyusun tersedia tidak dapat diandalkan atau tidak menghasilkan kode obyek cukup efisien. Risiko menyebabkan perubahan perangkat lunak yang diusulkan dan masalah proyek seperti jadwal dan biaya yang keluar batas, sehingga meminimalkan risiko adalah aktivitas manajemen proyek yang sangat penting. manajemen risiko, merupakan bagian penting dari manajemen proyek (Sommerville, 2011).

### **6.2.3 Kelebihan Metode Spiral**

Berikut ini merupakan kelebihan dari metode spiral antara lain :

- a. Sangat mempertimbangkan resiko kemungkinan munculnya kesalahan sehingga sangat dapat diandalkan untuk pengembangan perangkat lunak skala besar.
- b. Pendekatan model ini dilakukan melalui tahapan-tahapan yang sangat baik dengan menggabungkan model waterfall ditambah dengan pengulangan-pengulangan sehingga lebih realistis untuk mencerminkan keadaan sebenarnya.
- c. Baik pengembang maupun pemakai dapat cepat mengetahui letak kekurangan dan kesalahan dari sistem karena proses-prosesnya dapat diamati dengan baik.

#### **6.2.4 Kelemahan Metode Spiral**

Kelemahan metode Spiral antara lain :

- a. Waktu yang dibutuhkan untuk mengembangkan perangkat lunak cukup panjang demikian juga biaya yang besar.
- b. Sangat tergantung kepada tenaga ahli yang dapat memperkirakan resiko.
- c. Terdapat pula kesulitan untuk mengontrol proses. Sampai saat ini, karena masih relatif baru, belum ada bukti apakah metode ini cukup handal untuk diterapkan.
- d. Meyakinkan konsumen (khususnya dalam situasi kontrak) bahwa pendekatan evolusioner bisa dikontrol.

## **BAB VII**

# **PENGUJIAN PERANGKAT LUNAK**

Pengujian software adalah proses verifikasi dan validasi apakah sebuah aplikasi software atau program memenuhi persyaratan bisnis dan persyaratan teknis yang mengarahkan desain dan pengembangan dan cara kerjanya seperti yang diharapkan dan juga mengidentifikasi kesalahan yang penting yang digolongkan berdasarkan tingkat severity pada aplikasi yang harus diperbaiki. pengujian software juga merupakan teknik yang sering digunakan untuk verifikasi dan validasi kualitas suatu software. Pengujian software adalah prosedur untuk eksekusi sebuah program atau sistem dengan tujuan untuk menemukan kesalahan. Pada dasarnya pengujian perangkat lunak bertujuan untuk :

- a. Menemukan kesalahan (fault) sebanyak mungkin dari perangkat lunak yang diuji.
- b. Membuat perangkat lunak yang diuji, setelah perbaikan dilakukan, menjadi perangkat lunak yang berkualitas.
- c. Melakukan pengujian secara efektif dan efisien.
- d. Mengumpulkan kesalahan yang terjadi dan menggunakannya untuk tindakan preventif.

### **7.1 Performance Testing**

Performance test adalah integration dan usability test yang menentukan apakah sistem atau subsistem dapat memenuhi kriteria kinerja berbasis waktu seperti response time atau throughput. Response time menentukan batas waktu maksimum yang diijinkan dari respon

software untuk query dan update. Throughput menentukan jumlah minimum query dan transaksi yang harus diproses per menit atau per jam.

### **7.1.1 Jenis Pengujian Dalam Performance Testing**

Secara sederhana tujuan dari pengujian performance Testing ini adalah menemukan perbedaan antara goal (kebutuhan non fungsional) yang ditentukan pada saat pengembangan sistem dengan yang diimplementasikan dalam sistem. Berikut ini jenis – jenis pengujian dalam performance testing :

a. Load Testing.

Load Testing adalah bentuk pengujian kinerja yang paling sederhana. Uji beban biasanya dilakukan untuk memahami perilaku sistem di bawah beban khusus yang diharapkan. Beban ini dapat berupa jumlah pengguna bersamaan yang diharapkan di aplikasi yang melakukan sejumlah transaksi tertentu dalam durasi yang ditetapkan. Tes ini akan memberikan waktu respons dari semua transaksi penting bisnis yang penting. Basis data, server aplikasi juga dimonitor selama pengujian ini akan membantu dalam mengidentifikasi kemacetan dalam perangkat lunak aplikasi dan perangkat keras tempat perangkat lunak diinstal.

b. Stress testing

Biasanya digunakan untuk memahami batas atas kapasitas dalam sistem. Jenis pengujian ini dilakukan untuk menentukan ketahanan sistem dalam hal beban ekstrem dan membantu administrator aplikasi untuk menentukan apakah sistem akan bekerja secara memadai jika beban saat ini berjalan jauh di atas maksimum yang diharapkan.

c. Soak testing.

Soak Testing juga dikenal sebagai Endurance Testing, biasanya dilakukan untuk menentukan apakah sistem dapat menopang

beban diberikan terus menerus. Selama soak testing, penggunaan memori dipantau untuk mendeteksi potensi kebocoran. Yang juga penting, tetapi sering diabaikan adalah penurunan kinerja, yaitu untuk memastikan bahwa waktu throughput dan / atau respons setelah beberapa periode panjang aktivitas berkelanjutan sama baiknya atau lebih baik daripada di awal pengujian. Ini pada dasarnya melibatkan penerapan beban yang signifikan ke sistem untuk jangka waktu yang lama dan signifikan. Tujuannya adalah untuk menemukan bagaimana sistem berperilaku dalam penggunaan berkelanjutan.

d. Spike testing.

Spike Testing dilakukan dengan tiba-tiba menambah atau mengurangi beban yang dihasilkan oleh sejumlah besar pengguna, dan mengamati perilaku sistem. Tujuannya adalah untuk menentukan apakah kinerja akan terganggu, sistem akan gagal, atau akan mampu menangani perubahan dramatis pada beban.

e. Breakpoint testing.

Breakpoint Testing hampir mirip dengan stress testing. Beban tambahan diterapkan dari waktu ke waktu saat sistem dipantau untuk kondisi kegagalan yang telah ditentukan sebelumnya. Pengujian breakpoint terkadang disebut sebagai pengujian kapasitas karena dapat dikatakan untuk menentukan kapasitas maksimum di bawah yang akan dilakukan sistem sesuai spesifikasi yang diperlukan atau perjanjian tingkat layanan. Hasil analisis breakpoint yang diterapkan pada lingkungan tetap dapat digunakan untuk menentukan strategi penskalaan optimal dalam hal perangkat keras yang diperlukan atau kondisi yang harus memicu peristiwa over load di cloud.

f. Configuration testing.

Pengujian ini dibuat untuk menentukan perubahan konfigurasi pada komponen sistem pada kinerja dan perilaku sistem. Contoh

umum adalah bereksperimen dengan berbagai metode load-balancing.

g. Isolation testing.

Isolation Testing merupakan pengujian yang melibatkan pengulangan eksekusi yang mengakibatkan masalah sistem. Pengujian semacam itu seringkali dapat mengisolasi dan mengkonfirmasi kesalahan domain.

h. Internet testing.

Internet Testing adalah bentuk pengujian kinerja yang relatif baru ketika aplikasi global seperti Facebook, Google dan Wikipedia, diuji kinerja dari generator beban yang ditempatkan di benua target sebenarnya baik mesin fisik atau VM cloud. Tes-tes ini biasanya membutuhkan banyak persiapan dan pemantauan agar dapat dijalankan dengan sukses.

### **7.1.2 Langkah Pengujian Dalam Performance Testing.**

Dalam melakukan pengujian suatu perangkat lunak menggunakan performance testing, pengembangan bisa menjalankan 7 (tujuh) langkah pengujian berikut ini :

a. Identify the Testing Environment.

Mengidentifikasi perangkat keras, perangkat lunak, konfigurasi jaringan dan alat yang tersedia yang memungkinkan tim penguji merancang dan mengidentifikasi tantangan pengujian lebih awal.

b. Identify Performance Metrics.

Selain mengidentifikasi metrik seperti waktu respons, throughput, dan batasan, identifikasi apa saja kriteria keberhasilan untuk pengujian kinerja.

c. Plan and Design Performance Tests.

Identifikasi skenario pengujian performa yang memperhitungkan variabilitas pengguna, data pengujian, dan metrik target. Ini akan membuat satu atau dua model.

- d. **Configure the Test Environment**  
Menyiapkan semua elemen dalam pengujian serta instrumen yang diperlukan untuk memantau sumber daya.
- e. **Implement Your Test Design**  
Mengimpelentasikan desain pengujian serta melakukan costum desain pengujian.
- f. **Execute Tests.**  
Selain menjalankan tes kinerja dari perangkat luank tersebut penguji harus melakukan pemantauan serta menyimpan data yang dihasilkan.
- g. **Analyze, Report, Retest.**  
Melakukan analisis data yang terekam serta melakukan pengujian lagi menggunakan parameter yang sama dan parameter berbeda.

### **7.1.3 Tools Peformance Testing**

Ada berbagai macam alat pengujian yang tersedia di internet. Alat yang dipilih untuk pengujian akan bergantung pada banyak faktor seperti jenis protokol yang didukung, biaya lisensi, persyaratan perangkat keras, dukungan platform, dll. Dalam pengujian kinerja dibagi menjadi 2 (dua) kategori yaitu:

#### **a. Performance scripting.**

Bagian dari pengujian kinerja ini terutama berhubungan dengan pembuatan alur kerja proses bisnis yang teridentifikasi utama. Ini dapat dilakukan menggunakan berbagai macam alat seperti HP LoadRunner, NeoLoad, Apache JMeter, Rational Performance Tester, Silk Performer, Gatling, dan Flood.

Setiap alat yang disebutkan dalam daftar di atas (yang tidak lengkap atau lengkap) baik menggunakan bahasa skrip (C, Java, JS) atau beberapa bentuk representasi visual (seret dan lepas) untuk membuat dan mensimulasikan alur kerja pengguna akhir. Sebagian

besar alat memungkinkan untuk sesuatu yang disebut “Rekam & Putar Ulang”, dimana dalam pengujian kinerja akan meluncurkan alat pengujian, mengaitkannya di browser atau klien tebal dan menangkap semua transaksi jaringan yang terjadi antara klien dan server. Dengan demikian, skrip dikembangkan yang dapat ditingkatkan/dimodifikasi untuk meniru berbagai skenario bisnis.

**b. Performance monitoring.**

Ini membentuk wajah lain dari pengujian kinerja. Dengan pemantauan kinerja, perilaku dan karakteristik respons dari aplikasi yang diuji diamati. Parameter di bawah ini biasanya dipantau selama eksekusi uji kinerja, Parameter perangkat keras server seperti 1) Pemanfaatan CPU; 2) Pemanfaatan Memori; 3) Pemanfaatan disk; 4) Pemanfaatan jaringan.

Sebagai langkah pertama, pola yang dihasilkan oleh 4 parameter ini memberikan indikasi yang baik tentang letak hambatan. Untuk menentukan akar penyebab pasti dari masalah tersebut, insinyur perangkat lunak menggunakan alat seperti profiler untuk mengukur bagian mana dari perangkat atau perangkat lunak yang berkontribusi paling besar terhadap kinerja yang buruk, atau untuk menetapkan tingkat throughput (dan ambang batas) untuk mempertahankan waktu respons yang dapat diterima.

**7.2 System Testing,**

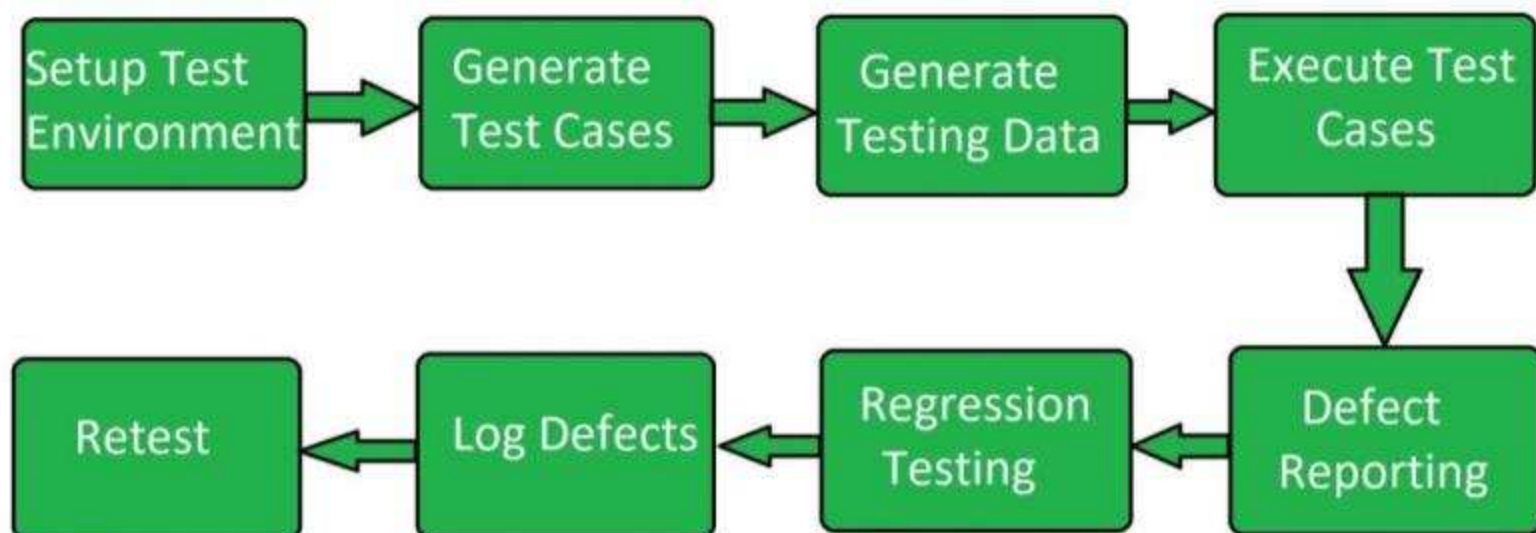
System Testing adalah jenis pengujian perangkat lunak yang dilakukan pada sistem terintegrasi lengkap untuk mengevaluasi kesesuaian sistem dengan persyaratan yang sesuai. Dalam System Testing, hasil dari pengujian integrasi komponen diambil sebagai masukan. Tujuan dari pengujian integrasi adalah untuk mendeteksi ketidakteraturan antara unit-unit yang terintegrasi bersama. System Testing mendeteksi cacat di dalam unit terintegrasi dan seluruh

sistem. Hasil pengujian sistem adalah perilaku yang diamati dari suatu komponen atau sistem ketika diuji.

Pengujian Sistem dilakukan pada keseluruhan sistem dalam konteks spesifikasi kebutuhan sistem atau spesifikasi kebutuhan fungsional atau dalam konteks keduanya. Pengujian sistem menguji desain dan perilaku sistem dan juga harapan pelanggan. Ini dilakukan untuk menguji sistem di luar batas yang disebutkan dalam spesifikasi kebutuhan perangkat lunak (SRS).

### 7.2.1 Proses System Testing.

Pengujian Sistem pada dasarnya dilakukan oleh tim pengujian yang independen dari tim pengembangan yang membantu untuk menguji kualitas sistem secara tidak memihak. Ini memiliki pengujian fungsional dan non-fungsional. Berikut ini merupakan proses pengujian sistem :



### 7.3 Unit Testing

Unit testing adalah proses metode pengujian individual, class, atau komponen sebelum mereka terintegrasi dengan perangkat lunak lainnya. Tujuan dari unit testing adalah untuk mengidentifikasi dan memperbaiki kesalahan sebanyak mungkin sebelum modul – modul digabungkan menjadi unit perangkat lunak yang lebih besar, seperti

program, class dan subsistem. Kesalahan menjadi lebih sulit dan mahal untuk ditemukan dan diperbaiki ketika banyak unit telah digabungkan.

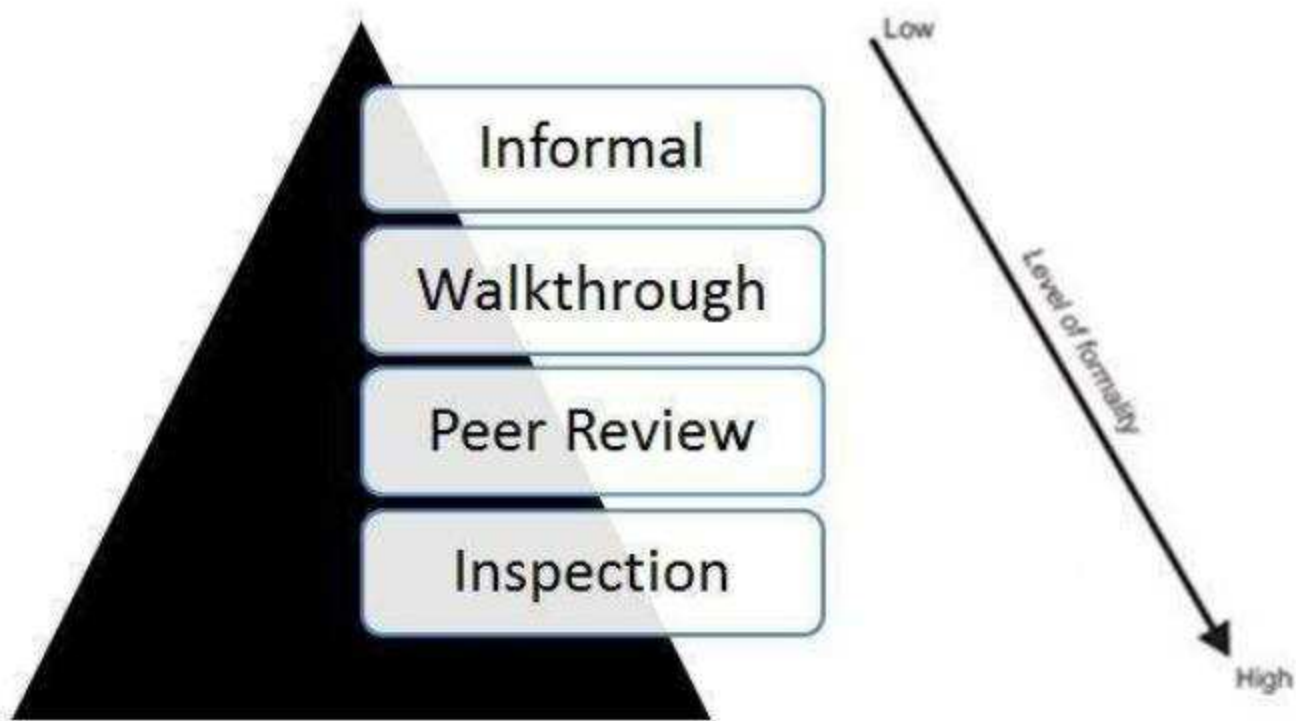
Unit testing memerlukan implementasi dari driver dan/atau stub. Stub adalah class atau method dummy yang dapat dipanggil namun biasanya tidak melakukan apapun kecuali mengembalikan tipe yang diperlukan. Modul driver adalah program yang menjalankan method atau fungsi dari class yang akan dites.

### **7.3.1 Tipe Unit Testing**

Terdapat 2 (dua) tipe dalam pengujian unit yaitu pengujian statis dan pengujian dinamis. Pengujian statis dan pengujian dinamis adalah dua jenis pengujian umum yang dilakukan seseorang sebagai pengembang perangkat lunak. Ini adalah alat yang paling penting yang tersedia baginya untuk mengamankan siklus pengembangan perangkat lunak. Pengembang harus menggunakan kedua alat untuk menentukan apakah perangkat lunak yang dikembangkan siap untuk dirilis di pasar.

#### **7.3.1.1 Pengujian Statis**

Pengujian statis adalah pengujian yang terjadi bahkan sebelum kode tertulis perangkat lunak dieksekusi. Pengembang mendapat kesempatan untuk melakukan pengkodean dengan sisir bergigi halus untuk melihat apakah ada kesalahan. Ini juga memungkinkan mereka untuk melihat apakah kode mematuhi hukum setempat. Pengujian statis mengungkapkan kekurangan dan menyediakan pengembang untuk memperbaikinya sebelum eksekusi terjadi. Dalam beberapa kasus, pengembang juga dapat mengidentifikasi kode berbahaya apa pun yang dapat menyebabkan masalah selama eksekusi. Berikut merupakan jenis pengujian statis :



a. Informal Review.

Informal review adalah teknik pengujian statis informal yang dilakukan pada segala jenis persyaratan, desain, kode, atau rencana proyek. Selama tinjauan informal, produk pekerjaan diberikan kepada ahli domain dan umpan balik / komentar ditinjau oleh pemilik / programmernya.

b. Code Walkthrough.

Code Walkthrough adalah bentuk tinjauan sejawat di mana pemrogram memimpin proses tinjauan dan anggota tim lainnya mengajukan pertanyaan dan menemukan kemungkinan kesalahan terhadap standar pengembangan dan masalah lainnya. Rapat biasanya dipimpin oleh penulis dokumen yang sedang ditinjau dan dihadiri oleh anggota tim lainnya. Sesi review mungkin formal atau informal. Sebelum pertemuan walkthrough, persiapan oleh reviewer dan kemudian laporan review dengan daftar temuan. Juru tulis, yang bukan penulisnya, menandai risalah rapat dan mencatat semua kekurangan / masalah sehingga dapat dilacak hingga penutupan. Tujuan utama dari panduan ini adalah untuk memungkinkan pembelajaran tentang isi dokumen yang sedang ditinjau untuk membantu anggota tim mendapatkan pemahaman

tentang isi dokumen dan juga untuk menemukan cacat.

c. Peer Review.

Peer Review adalah teknik pengujian white Box statis yang dilakukan untuk menemukan cacat di awal siklus hidup yang tidak dapat dideteksi dengan teknik pengujian kotak hitam. Dalam teknik pengujian ini terdapat karakteristik yaitu tinjauan Teknis didokumentasikan dan menggunakan proses deteksi cacat yang memiliki rekan kerja dan pakar teknis sebagai bagian dari proses tinjauan. Proses Review tidak melibatkan partisipasi manajemen. Biasanya dipimpin oleh moderator terlatih yang bukan programmernya. Laporan tersebut disiapkan dengan daftar masalah yang perlu ditangani.

d. Inspection.

Inspection adalah bentuk tinjauan yang paling formal, strategi yang diadopsi selama fase pengujian statis. Inspeksi biasanya dipimpin oleh moderator terlatih, yang bukan pragrammernya. Peran moderator adalah melakukan pemeriksaan sejawat terhadap suatu dokumen. Inspeksi paling formal dan didorong oleh daftar periksa dan aturan. Proses review ini menggunakan kriteria masuk dan keluar. Penting untuk memiliki persiapan sebelum pertemuan. Laporan inspeksi disiapkan dan dibagikan dengan penulis untuk tindakan yang sesuai. Pasca Inspeksi, proses tindak lanjut formal digunakan untuk memastikan tindakan korektif yang tepat waktu dan cepat. Tujuan Inspeksi BUKAN hanya untuk mengidentifikasi cacat tetapi juga untuk meningkatkan proses.

### 7.3.1.2 Pengujian Dimanis

Pengujian dinamis terjadi dalam lingkungan run-time, yang berarti bahwa kode dieksekusi dengan analisis keamanan untuk melihat cara kerjanya. Ini memungkinkan pengembang untuk menentukan apakah perangkat lunak berjalan dan juga mengidentifikasi jika

menghasilkan hasil yang sama seperti yang diharapkan pengembang. Hal ini memungkinkan pengembang untuk menganalisis perilaku fungsional perangkat lunak, dan memantau interaksinya dengan memori sistem, fungsi CPU, dan kinerja sistem secara keseluruhan. Pengujian dinamis sering disebut sebagai validasi evaluasi produk jadi. Pengujian dinamis dibagi menjadi beberapa kategori. Berikut ini jenis-jenis kategori pengujian dinamis :

a. Specification Based.

Specification based adalah teknik pengujian dinamis berdasarkan prosedur tertulis, spesifikasi, persyaratan, manual pengguna, kasus penggunaan, prototipe layar, dan proses bisnis. Ini juga dapat didefinisikan sebagai pengujian Blackbox dan memiliki banyak sub teknik seperti partisi kesetaraan, analisis nilai batas, pengujian berpasangan, pengujian transisi status, pengujian use case, pengujian cerita pengguna, pengujian tabel keputusan, pengujian kombinatorial, pengujian pohon klasifikasi, dan begitu seterusnya.

b. Structure Based.

Structure based merupakan teknik pengujian dinamis yang didasarkan pada struktur internal kode, basis data, arsitektur, atau aliran sistem. Structure base Ini dapat didefinisikan sebagai pengujian white box dan memiliki banyak sub teknik seperti pengujian pernyataan, pengujian keputusan, pengujian beberapa kondisi, pengujian jalur, pengujian cabang, dan pengujian API. Teknik-teknik ini berbeda satu sama lain berdasarkan kriteria cakupan yang mereka miliki.

c. Model Based.

Model based adalah teknik pengujian dinamis yang didasarkan pada representasi model produk perangkat lunak. Kasus uji dirancang dari model bukan dari kode sumber, akibatnya pengujian berbasis model tidak dapat diambil sebagai aktivitas pengujian kotak putih. Sebaliknya, model dapat dianggap sebagai

presentasi parsial dan abstrak dari sistem yang diuji.

d. Risk Based.

Risk based adalah teknik pengujian dinamis yang didasarkan pada risiko produk perangkat lunak. Untuk mengidentifikasi risiko dengan baik, beberapa pemangku kepentingan seperti pengembang, agen help-desk / call-center, audit, pengguna, analis bisnis, penguji, dan staf operasi harus disertakan, dan pandangan mereka harus dipertimbangkan. Untuk identifikasi risiko terperinci, Anda dapat menggunakan teknik seperti sesi curah pendapat, lokakarya risiko, metrik tolok ukur dari proyek sebelumnya, wawancara ahli, dan pertemuan retrospektif.

e. Defect Based.

Defect based merupakan teknik pengujian dinamis yang didasarkan pada jenis, klasifikasi, dan pengelompokan cacat. Kategori cacat seperti cacat antarmuka, masalah waktu, masalah komputasi, masalah bidang teks, masalah bidang tanggal, dan masalah terkait data dapat diambil sebagai dasar untuk pengujian berbasis cacat.

f. Experience Based.

Experience based adalah teknik pengujian dinamis yang didasarkan pada praktik penguji di bidang tertentu. Kesalahan menebak, pengujian berbasis daftar periksa, serangan, dan pengujian eksplorasi dapat dianggap sebagai teknik desain pengujian berbasis pengalaman.

g. Platform Based.

Platform based adalah teknik pengujian dinamis yang didasarkan pada karakteristik dan batasan platform seluler yang berbeda. Pengaturan OS, fitur khusus platform, perilaku toko aplikasi, dan fitur multitasking dapat dianggap sebagai atribut khusus platform yang harus menjadi fokus penguji.

h. Device Based.

Device based adalah teknik pengujian dinamis berdasarkan karakteristik dan batasan perangkat seluler yang berbeda seperti ponsel cerdas, ponsel menengah, tablet, atau phablet (perangkat yang dirancang untuk menggabungkan fungsi ponsel cerdas dan tablet). Pengaturan perangkat, fitur tersembunyi, gangguan, pemberitahuan, penggunaan baterai, gerakan, frekuensi sesi, durasi sesi, dan fitur khusus perangkat dapat diambil sebagai atribut berbasis perangkat yang harus menjadi fokus penguji.

### 7.3.2 Tools Unit Testing

Ada beberapa alat otomatis yang tersedia untuk membantu pengujian unit. Kami akan memberikan beberapa contoh di bawah ini:

a. Junit.

Adalah alat pengujian gratis yang digunakan untuk bahasa pemrograman Java. Ini memberikan pernyataan untuk mengidentifikasi metode pengujian. Alat ini menguji data terlebih dahulu dan kemudian dimasukkan ke dalam potongan kode.

b. Nunit.

Banyak digunakan kerangka pengujian unit yang digunakan untuk semua bahasa.net. Ini adalah alat open source yang memungkinkan penulisan skrip secara manual. Ini mendukung tes berbasis data yang dapat berjalan secara paralel.

c. Jmockit.

Adalah alat pengujian Unit sumber terbuka. Ini adalah alat cakupan kode dengan metrik garis dan jalur. Ini memungkinkan mocking API dengan sintaks perekaman dan verifikasi. Alat ini menawarkan cakupan Jalur, Cakupan Jalur, dan Cakupan Data.

d. EMMA.

Adalah toolkit sumber terbuka untuk menganalisis dan melaporkan kode yang ditulis dalam bahasa Java. Emma mendukung jenis cakupan seperti metode, garis, blok dasar.

Ini berbasis Java sehingga tanpa ketergantungan perpustakaan eksternal dan dapat mengakses kode sumber.

e. **PHPUnit.**

Adalah alat pengujian unit untuk programmer PHP. Dibutuhkan sebagian kecil kode yang disebut unit dan mengujinya secara terpisah. Alat ini juga memungkinkan pengembang untuk menggunakan metode pernyataan yang telah ditentukan sebelumnya untuk menegaskan bahwa sistem berperilaku dengan cara tertentu.

Itu hanyalah beberapa dari alat pengujian unit yang tersedia. Masih banyak lagi, terutama untuk bahasa C dan Java, tetapi Anda pasti akan menemukan alat pengujian unit untuk kebutuhan pemrograman Anda terlepas dari bahasa yang Anda gunakan.

### **7.3.3 Keuntungan Unit Testing**

Berikut ini merupakan keuntungan dalam penggunaan Unit testing dalam pengujian perangkat lunak antara lain :

- a. Pengembang yang ingin mempelajari fungsionalitas apa yang disediakan oleh sebuah unit dan bagaimana menggunakannya dapat melihat pengujian unit untuk mendapatkan pemahaman dasar tentang API unit.
- b. Pengujian unit memungkinkan pemrogram untuk merefaktor kode dikemudian hari, dan memastikan modul masih berfungsi dengan benar (yaitu pengujian Regresi). Prosedurnya adalah menulis kasus uji untuk semua fungsi dan metode sehingga setiap kali perubahan menyebabkan kesalahan, itu dapat dengan cepat diidentifikasi dan diperbaiki.
- c. Karena sifat modular dari pengujian unit, kami dapat menguji bagian-bagian proyek tanpa menunggu bagian lain diselesaikan.

### **7.3.4 Kekurangan Unit Testing.**

Berikut ini kekurangan dalam Unit Testing antara lain:

- a. Pengujian unit tidak dapat diharapkan untuk menangkap setiap kesalahan dalam program. Tidaktah mungkin untuk mengevaluasi semua jalur eksekusi.
- b. Pengujian unit pada dasarnya berfokus pada unit kode. Oleh karena itu tidak dapat menangkap kesalahan integrasi atau tingkat kesalahan sistem yang luas.

## 7.4 Integration Testing

Integration test adalah mengevaluasi behavior dari kelompok method atau class. Tujuan dari integration test adalah untuk mengidentifikasi kesalahan yang tidak dapat dideteksi oleh unit testing. Kesalahan tersebut mungkin disebabkan oleh beberapa masalah, diantaranya :

- a. Interface incompatibility, misalnya sebuah method melewati parameter dengan tipe data yang salah ke method lainnya.
- b. Parameter values, misalnya sebuah method mengembalikan nilai yang tidak terduga seperti nomor negatif untuk harga.
- c. Run-time exceptions, misalnya method menyebabkan kesalahan seperti?, out of memory?, atau file already in use? karena ada konflik kebutuhan sumber daya.
- d. Unexpected state interactions, misalnya state dari dua atau lebih objek yang berinteraksi menyebabkan kesalahan yang kompleks seperti ketika method class Order menjalankan satu kesalahan dari semua kemungkinan state objek Customer.

Beberapa masalah di atas merupakan kesalahan paling umum yang sering ditemui dalam integration testing, tetapi sebenarnya masih banyak masalah lainnya yang dapat menjadi penyebab kesalahan (error).

### 7.4.1 Tingkatan Integration Testing.

Integration testing adalah untuk memastikan modul-modul dan antarmuka dalam suatu aplikasi berinteraksi satu sama lain dengan benar dan aman. Pada dasarnya, integration testing berdasarkan pada

spesifikasi dan rancangan persyaratan fungsional yang digunakan sebagai input dalam proses integration testing. integration testing terdiri dari 5 tingkat dasar, yaitu :

- a. Level transaction.  
Pengujian hanya dilihat dari sebuah transaksi.
- b. Level in-stream.  
Pengujian mencakup sebuah alur yang terdiri dari beberapa transaksi yang saling berkaitan.
- c. Level cross-stream.  
Pengujian yang mencakup sekumpulan alur mulai dari awal sampai akhir dari proses-proses transaksi yang saling berkaitan.
- d. Level regression.  
Pengujian yang sama seperti pengujian level cross-stream tetapi menggunakan kondisi yang tidak diinginkan/tidak seperti biasanya.
- e. Level user acceptance.  
Pengujian dimana dilakukan oleh user yang sebenarnya yang bertujuan untuk menguji validasi yang seharusnya

#### **7.4.2 Langkah Integration Testing.**

Sebuah Modul pada umumnya dirancang oleh pengembang perangkat lunak individu yang pengertian dan logika pemrograman mungkin berbeda dari programmer lain. Integration Testing menjadi perlu untuk memverifikasi perangkat lunak modul bekerja dalam kesatuan. Secara sederhana berikut ini merupakan langkah – dalam integration testing :

- a. Siapkan rencana pengujian integrasi yang efektif dan identifikasi antarmuka unit. Sebelum merencanakan untuk melakukan pengujian integrasi, tim perlu mempersiapkan strategi rencana pengujian. Ini membantu mereka melakukan pengujian yang sedang berjalan. Secara bersamaan, pengembang harus

- mengidentifikasi dan mendokumentasikan kasus uji dan data uji.
- b. Identifikasi modul penting untuk diuji berdasarkan prioritas. Pada langkah ini, penting untuk mempelajari desain arsitektur aplikasi. Ini akan membantu untuk mengidentifikasi modul penting yang memerlukan pengujian integrasi berdasarkan prioritas dan juga memiliki pemahaman tentang modul lain.
  - c. Periksa antarmuka. Saat merancang kasus pengujian dan skenario pengujian, penting untuk mengumpulkan informasi yang terkait dengan templat pengujian integrasi. Antarmuka harus direkonsiliasi, yaitu ketika satu unit X mentransfer data ke unit lain Y, maka harus ada indikasi yang jelas bahwa data telah ditransfer dari unit X. Jika tidak ada indikasi diantara antarmuka tersebut, maka ini perlu dilakukan. diperiksa dan diperbaiki sebelum melangkah lebih jauh untuk pengujian integrasi.
  - d. Peragakan kondisi uji Integrasi. Untuk memproses pengujian integrasi untuk setiap unit program, kondisi pengujian harus disiapkan. Setelah kondisi dicantumkan, ini harus didokumentasikan, sehingga dapat digunakan saat melakukan pengujian integrasi.
  - e. Pilih alat otomatisasi dan lakukan pengujian. Saat skrip pengujian dan kondisi pengujian disiapkan, pilih alat pengujian otomatisasi untuk pengujian integrasi. Jalankan kasus pengujian dan jika ada cacat yang teridentifikasi, laporkan kerusakan tersebut untuk pengujian ulang.
  - f. Tes Ulang untuk Memvalidasi Kelengkapan Tes Integrasi. Untuk kasus uji yang melaporkan cacat, lakukan pengujian ulang. Langkah ini harus dilakukan hingga pengujian integrasi selesai dan semua persyaratan terpenuhi.

#### **7.4.3 Teknik Integration Testing.**

Tujuan pengujian integrasi adalah untuk memverifikasi fungsional, kinerja, dan keandalan antara modul yang terintegrasi. Dalam melakukan Integration Testing terdapat 4 teknik yang sering

digunakan oleh software Tester:

**a. Big Bang Integration.**

Big Bang Integration Testing adalah strategi pengujian integrasi di mana semua unit terhubung sekaligus, menghasilkan sistem yang lengkap. Ketika jenis strategi pengujian ini diadopsi, sulit untuk mengisolasi kesalahan yang ditemukan, karena perhatian tidak diberikan untuk memverifikasi antarmuka di unit individu. Dalam pendekatan ini modul individu tidak terintegrasi sampai semua modul siap. Dalam pengujian integrasi Big Bang semua modul yang terintegrasi tanpa melakukan pengujian integrasi dan kemudian itu dilakukan untuk mengetahui apakah semua modul yang terintegrasi bekerja dengan baik atau tidak. Pendekatan ini umumnya dilakukan oleh para pengembang yang melakukan pendekatan “Jalankan dan lihat”. Karena mengintegrasikan semuanya pada satu waktu maka jika ada kegagalan terjadi akan menyulitkan programmer untuk mengetahui akar penyebab kegagalan itu.

**b. Top-Down Integration Testing.**

Pengujian berlangsung dari atas ke bawah, mengikuti aliran kontrol atau struktur arsitektur (inisialkan mulai dari GUI atau menu utama). Komponen atau sistem yang diganti oleh stub. Berikut adalah diagram dari “Pendekatan Top down Integration Testing”. Kelebihan dari top-download integration testing ini antara lain:

- 1) Lebih mudah melacak bug apabila terjadi kegagalan di sistem.
- 2) Kemungkinan untuk mendapatkan prototipe awal.
- 3) Lebih mudah melacak Cacat desain utama.

**c. Bottom-up integration testing.**

Pengujian berlangsung dari bawah kontrol mengalir ke atas. Komponen atau sistem yang diganti oleh driver. Kelebihan dari

pengujian ini adalah pengujian bisa lebih teliti dan lebih mudah melacak bug apabila terjadi kegagalan di sistem.

**d. Hybrid integration testing.**

Pengujian Integrasi adalah fase dalam pengujian perangkat lunak dimana modul mandiri digabungkan dan diuji sebagai satu kesatuan. Selama fase itu, antarmuka dan komunikasi antara masing-masing modul tersebut diuji. Ada dua pendekatan populer untuk pengujian Integrasi yaitu Pengujian Integrasi Top down dan Pengujian Integrasi Bottom up. Dalam Pengujian ini, memanfaatkan keunggulan pendekatan Top-down dan Bottom-up. Pengujian Integrasi Hibrid memiliki Fitur:

1. Pengujian ini memiliki tiga lapisan; yaitu - Lapisan Target Utama, lapisan di atas lapisan target dan lapisan di bawah lapisan target.
2. Pengujian terutama difokuskan untuk lapisan target tingkat menengah dan dipilih berdasarkan karakteristik sistem dan struktur kode.
3. Pengujian Integrasi Hibrid dapat diadopsi jika pelanggan ingin mengerjakan versi aplikasi yang berfungsi sesegera mungkin yang bertujuan untuk menghasilkan sistem kerja dasar pada tahap awal siklus pengembangan.

**7.4.4 Tools Integration Testing.**

Ada berbagai tingkat pengujian dan satu tingkat yang paling penting adalah “Pengujian Integrasi” yang menggabungkan unit atau modul yang berbeda dan diuji sebagai satu kelompok. Ini juga menguji antarmuka antar modul dan mengidentifikasi cacat kritis yang disebabkan karena integrasi modul yang berbeda.

Tujuan dari pengujian integrasi adalah untuk memastikan bahwa modul individu berfungsi seperti yang diharapkan setelah menggabungkannya dengan modul lain. Banyak organisasi menggunakan pengujian unit gabungan atau pengujian alur kerja

fungsional ujung ke ujung yang digunakan untuk Pengujian Integrasi. Itu selalu baik untuk melakukan pengujian integrasi yang sering sehingga memastikan bahwa setelah menggabungkan modul integrasi bekerja dengan sempurna. Di pasar saat ini, berbagai alat Pengujian Integrasi tersedia yang membantu organisasi membuat kerangka kerja untuk membangun rangkaian pengujian integrasi. Berikut ini tools yang sering digunakan dalam melakukan integration testing : VectorCAST/C++, VectorCAST/Ada, Citrus Integration Testing, LDRA, Smart Integration Test Accelerator (SITA), FitNesse, Rational Integration Tester, Protractor dan masih banyak lagi tools yang bisa digunakan dalam melakukan pengujian ini.

#### **a. Usability Testing**

Usability test adalah test untuk menentukan apakah method, class, subsistem, atau sistem telah memenuhi persyaratan pengguna. Oleh karena banyaknya tipe persyaratan sistem baik yang fungsional maupun non-fungsional, maka banyak tipe dari usability test yang harus dilakukan di waktu yang berbeda. Umumnya usability test mengevaluasi persyaratan fungsional dan kualitas dari user interface. User berinteraksi dengan sistem untuk menentukan apakah fungsi telah seperti yang diharapkan dan apakah user interface membuat sistem dapat mudah digunakan. Pengujian ini sering dilakukan untuk mendapatkan feedback yang cepat dalam meningkatkan interface dan mengkoreksi kesalahan dalam komponen perangkat lunak.

Usability adalah analisa kualitatif yang menentukan seberapa mudah user menggunakan antarmuka suatu aplikasi (Nielsen, 2012). Suatu aplikasi disebut usable jika fungsi-fungsinya dapat dijalankan secara efektif, efisien, dan memuaskan (Nielsen, 1993). Efektivitas berhubungan dengan keberhasilan pengguna mencapai tujuan dalam menggunakan suatu perangkat lunak. Efisiensi berkenaan dengan kelancaran pengguna untuk mencapai tujuan tersebut. Kepuasan

berkaitan dengan sikap penerimaan pengguna terhadap perangkat lunak. Pengujian usability dilakukan untuk mengevaluasi apakah sebuah aplikasi sudah sesuai dengan kebutuhan pengguna atau belum.

### 7.5.1 Aspek Usability

Usability sebagai pengalaman suatu pengalaman penngguna dalam berinteraksi dengan aplikassi atau situs web sampai pengguna dapat mengoperasikannya dengan mudah dan cepat. Dari definisi tersebut, pengujian dalam penelitian menggunakan lima aspek usability atau lima atribut seperti yang dikemukakan oleh Jacob Nielsen dan sejalan dengan usability menurut ISO 9241:11 yaitu (Neilsen, 1993):

1. Kemudahan (learnability) didefinisikan seberapa cepat pengguna mahir dalam menggunakan sistem serta kemudahan dalam penggunaan menjalankan suatu fungsi serta apa yang pengguna inginkan dapat meraka dapatkan.
2. Efisiensi (efficiency) didefenisikan sebagai sumber daya yang dikeluarkan guna mencapai ketepatan dan kelengkapan tujuan.
3. Mudah diingat (memorability) didefinisikan bagaimana kemampuan pengguna mempertahankan pengetahuannya setelah jangka waktu tertentu, kemampuan mengingat didapatkan dari peletakkan menu yang selalu tetap
4. Kesalahan dan keamanan (errors) didefinisikan berapa banyak kesalahankesalahan apa saja yang dibuat pengguna, kesalahan yang dibuat pengguna mencangkup ketidaksesuaian apa yang pengguna pikirkan dengan apa yang sebenarnya disajikan oleh sistem.
5. Kepuasan (satisfaction) didefinisikan sebagai kebebasan dari ketidaknyamanan, dan sikap positif terhadap penggunaan produk atau ukuran subjektif sebagaimana pengguna merasa tentang penggunaan sistem.

### 7.5.2 Karakteristik Usability

Usability dapat diukur sebagai karakteristik kualitas produk, maupun diukur secara langsung yang merupakan bagian dari kualitas penggunaan. Adapun karakteristik dari usability sebagai berikut :

1. Efisiensi.

Efisiensi menurut KBBI menyatakan bahwa efisiensi adalah ketepatan cara (usaha, kerja) dalam menjalankan sesuatu (dengan tidak membuang waktu, tenaga, biaya) kedayagunaan, ketepatangunaan, serta kemampuan menjalankan tugas dengan baik dan tepat. (dengan tidak membuang waktu tenaga biaya). Dari uraian diatas disimpulkan bahwa efisiensi adalah suatu cara dengan bentuk usaha yang dilakukan dalam menjalankan sesuatu dengan baik dan tepat serta meminimalisir pemborosan dalam segi waktu, tenaga dan biaya.

2. Efektifitas.

Efektivitas berasal dari kata efektif, menurut kamus besar bahasa Indonesia efektif adalah ada efeknya, manjur atau mujarab, dapat membawa hasil, berhasil guna dan mulai berlaku. Sementara itu efektivitas memiliki pengertian keefektivan adalah keadaan berpengaruh.kemanjuran, keberhasilan dan hal mulai berlaku. Efektivitas merupakan dampak atau pengaruh dari membuat atau menghasilkan produk yang sesuai dengan keinginan atau sasaran yang ingin dicapai akan tetapi tetap menjadi tanggung jawab yang juga akan dirasakan dan dialami sendiri oleh individu yang menciptakan dan menjalankan.

3. Kemudahan (Learnbility).

Pada era perpustakaan digital disuatu decade belakangan ini, kata kemudahan mengakses sudah menjadi bagian yang tak terlepas dari keterpakaian atau kebergunaan. Beberapa standar pengukuran seberapa jauh sebuah sumber informasi dimanfaatkan, selalu memakai ukuran keudalian mengakses,

termasuk kesan pada pengguna tentang kemudahan mengakses.

4. Kesalahan.

Kesalahan dan keamanan (errors) didefinisikan berapa banyak kesalahan-kesalahan apa saja yang dibuat pengguna, kesalahan yang dibuat pengguna mencangkup ketidaksesuaian apa yang pengguna pikirkan dengan apa yang sebenarnya disajikan oleh sistem.

5. Kepuasan.

Kepuasan adalah perasaan senang atau kecewa seseorang yang berasal dari perbandingan antara kesannya terhadap kinerja yang dirasakan dari suatu produk dan harapan-harapannya (expectations). Kepuasan merupakan fungsi dari persepsi kesan atas kinerja dan harapan. Jika kinerja dibawah harapan maka konsumen tidak puas, jika kinerja memenuhi harapan maka konsumen puas. Jika kinerja melebihi harapan, konsumen sangat puas atau senang. Kepuasan dalam menggunakan suatu produk jasa adalah konsep penting. Oleh karena itu hal ini menjadi perhatian dari banyak pakar. Untuk mengukur kepuasan mengakses situs web selama melakukan penjelajahan, ialah seberapa jauh pengakses tersebut memiliki pengalaman yang menyenangkan pada saat mengakses situs web. Kualitas dan fitur situs web tentunya disediakan agar para pengguna memperoleh manfaat dan kepuasan dalam menggunakan situs web. Fitur merupakan fungsi-fungsi yang disediakan untuk membantu pengakses dalam menjelajahi situs web.

### 7.5.3 Tahapan Pengujian Usability

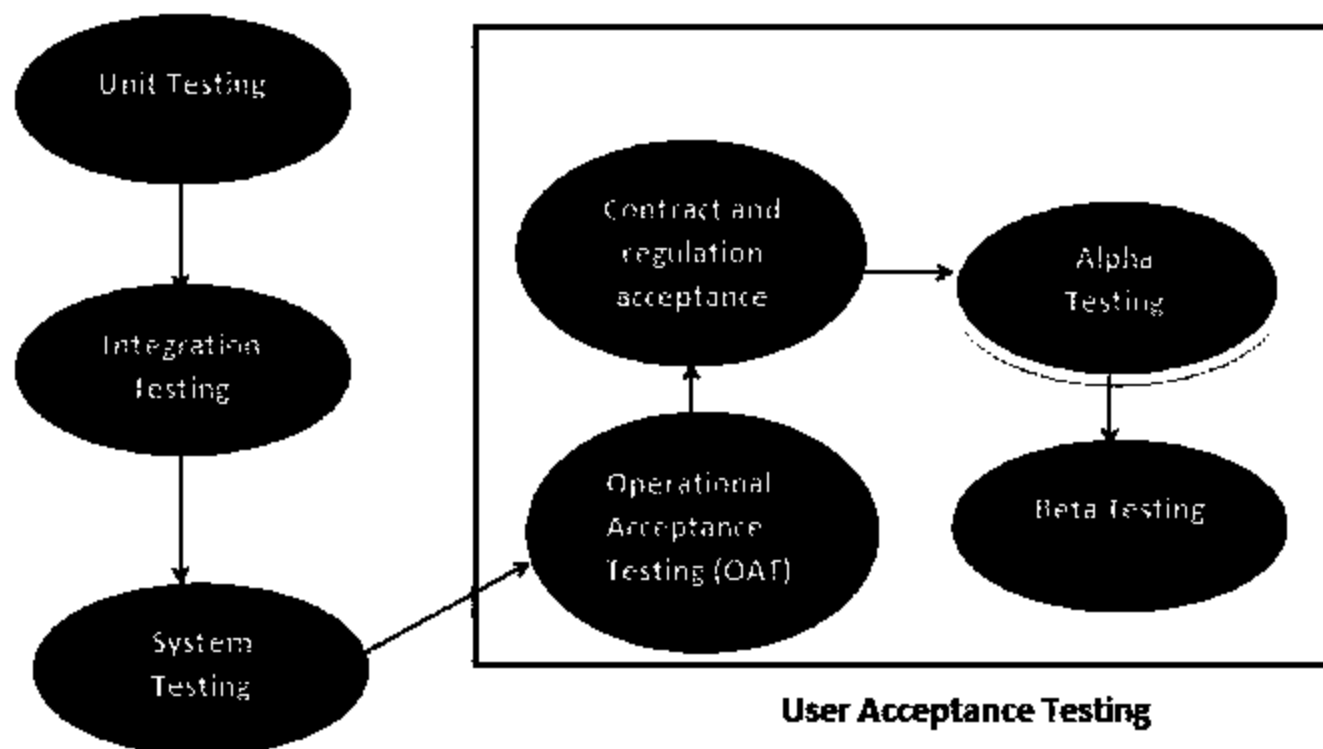
Di dalam proses pengembangan sebuah aplikasi, usability testing dapat dilakukan pada tahap perancangan maupun pada tahap pengembangan serta dapat juga pada saat tahapan evaluasi. Hal ini tergantung pada waktu yang tersedia, anggaran biaya serta tujuan yang ingin dicapai dari usability testing ini. Tahapan yang perlu dilakukan untuk melakukan Usability Testing adalah :

1. Tentukan sasaran yang ingin dicapai, apakah untuk melakukan re-design untuk aplikasi yang sudah ada, atau aplikasi yang belum pernah ada sebelumnya.
2. Siapkan prototype paper ataupun prototype digital yang akan digunakan untuk dinji. Jika pada tahap perancangan maka dapat menggunakan prototype paper maupun prototype digital yang masih berupa wireframe. Pada tahapan berikutnya dapat menggunakan prototype aplikasi maupun produk final dari aplikasi tersebut.
3. Tentukan partisipan yang akan melakukan ujicoba. Partisipan yang dipilih untuk melakukan ujicoba disesuaikan dengan target user dari aplikasi tersebut. Jumlah partisipan yang dipilih tergantung dari jenis aplikasi, biaya dan waktu yang tersedia. Untuk waktu yang terbatas dapat menggunakan hanya lima partisipan.
4. Buat rancangan task (aktivitas) yang akan diuji. Aktivitas yang dirancang harus jelas dan singkat agar tidak membuat partisipan bingung. Contohnya untuk aplikasi web e-commerce, aktivitas yang dirancang dapat berupa carilah produk dengan kategori pakaian anak-anak, atau lakukan pembelian 1 produk pakaian wanita dengan tipe pembayaran tunai, dan lain-lain.
5. Amati proses ujicoba yang berlangsung. Moderator diperlukan untuk melakukan pencatatan hal yang terjadi pada saat terjadi ujicoba. Jika dilakukan secara remote (online) maka dapat menggunakan aplikasi yang dapat merekam proses ujicoba yang berlangsung. Moderator tidak boleh membantu partisipan pada saat mencoba aplikasi, kecuali memang partisipan sudah memang belum berhasil setelah mencoba.
6. Buat rangkuman hasil uji coba. Seluruh hasil uji coba yang telah dicatat dari pengamatan maupun proses perekaman dibuatkan dalam satu laporan yang kemudian dianalisa apa isu – isu yang timbul dari uji coba yang dilakukan para partisipan.

## b. User Acceptance Test (UAT)

Dalam membangun suatu sistem, penting bagi tim proyek untuk mengetahui apakah sistem yang mereka buat atau kembangkan telah sesuai dengan apa yang diharapkan oleh klien.

Untuk mengetahui hal tersebut, tim proyek harus melakukan uji coba testing dengan mengacu pada harapan klien. Harapan klien untuk suatu sistem biasanya ditetapkan di awal pengerjaan atau pembuatan dari suatu sistem. Harapan-harapan yang telah ditetapkan ini merupakan tolak ukur yang membuat tim proyek mengetahui apa saja yang masih kurang dari suatu sistem yang dibuat itu. Beri petunjuk gambar seperti format di Gambar 6.1.



User Acceptance Testing merupakan pengujian yang dilakukan oleh end-user dimana user tersebut adalah staff / karyawan perusahaan yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan / fungsinya. Setelah dilakukan system testing, acceptance testing menyatakan bahwa sistem software memenuhi persyaratan.

Acceptance testing merupakan pengujian yang dilakukan oleh pengguna yang menggunakan teknik pengujian black box untuk menguji sistem terhadap spesifikasinya. Pengguna akhir bertanggung

jawab untuk memastikan semua fungsionalitas yang relevan telah diuji. acceptance testing biasanya berusaha menunjukkan bahwa sistem telah memenuhi persyaratan-persyaratan tertentu.

Pada pengembangan software dan hardware komersial, acceptance test biasanya disebut juga “alpha tests” (yang dilakukan oleh pengguna in-house) dan “beta tests” (yang dilakukan oleh pengguna yang sedang menggunakan atau akan menggunakan sistem tersebut). Alpha dan beta test biasanya juga menunjukkan bahwa produk sudah siap untuk dijual atau dipasarkan. Acceptance testing mencakup data, environment dan skenario yang sama atau hampir sama pada saat live yang biasanya berfokus pada skenario penggunaan produk tertentu.

#### **7.6.1 Proses Pengujian UAT**

Keakuratan User Acceptance Testing (UAT) dapat dipastikan hanya jika prosesnya dijalankan dengan benar dan tanpa hambatan. Ini dapat dicapai dengan mudah, jika penguji mengikuti proses yang ditetapkan. Oleh karena itu, disini disediakan proses User Acceptance Testing (UAT):

- a. Analisis Persyaratan : Pada tahap ini analisis kebutuhan dilakukan untuk mengidentifikasi dan mengembangkan skenario pengujian yang berasal dari berbagai dokumen, seperti Spesifikasi Persyaratan Sistem (SRS), Dokumen Persyaratan Bisnis (BRD), diagram alur proses, dan lainnya.
- b. Pembuatan Rencana Tes UAT. Di sini, strategi untuk menguji produk perangkat lunak diuraikan oleh tim. The rencana uji membantu mendokumentasikan masuk dan keluar kriteria untuk UAT dan digunakan untuk memvalidasi bahwa aplikasi memenuhi adalah kebutuhan bisnis.
- c. Identifikasi skenario pengujian. Setelah rencana pengujian dan strategi pengujian disiapkan oleh tim, mereka beralih ke pembuatan skenario pengujian berdasarkan proses bisnis dan

persyaratan mereka.

- d. Pembuatan kasus uji UAT. Selama tahap ini, kasus uji dibuat dengan langkah uji yang jelas, yang dirancang untuk mencakup sebagian besar skenario uji yang dibuat ditahap sebelumnya. Masukan yang diberikan untuk mempersiapkan kasus uji ini diberikan dalam bentuk kasus penggunaan bisnis.
- e. Eksekusi uji. Akhirnya, kasus uji dijalankan oleh tim di lingkungan UAT. Ini dapat dilakukan secara manual atau dengan bantuan alat manajemen pengujian .
- f. Rekam dan laporkan. Bug dan cacat yang terdeteksi oleh tim dicatat dan dilaporkan oleh tim penguji, untuk diperbaiki dan diselesaikan. Setelah bug diperbaiki, bug akan diuji ulang untuk memvalidasi keakuratan dan kuahtasnya.
- g. Konfirmasi Tujuan Bisnis. Di akhir proses, analis bisnis atau penguji UAT mengirimkan surat tanda tangan untuk melaporkan puncak pengujian. Ini dilakukan bersama dengan berbagai hasil yang ditawarkan oleh pengujian UAT.

#### **7.6.2 Pentingnya Pengujian UAT.**

Setelah sistem menjalani pengujian unit, pengujian integrasi, dan pengujian sistem, kebutuhan untuk pengujian penerimaan menjadi berlebihan. Namun, sistem atau aplikasi perangkat lunak masih memerlukan Pengujian Penerimaan Pengguna (UAT) karena alasan yang tak terhitung jumlahnya, yang tidak boleh diabaikan karena meningkatkan kuahtas produk serta membuatnya sesuai untuk rilis atau 'ditayangkan' secara keseluruhan. proyek. Berikut beberapa alasan pentingnya User Acceptance Testing (UAT) :

1. Dengan bantuan User Acceptance Testing (UAT), pengembang perangkat lunak dapat memastikan bahwa perangkat lunak dikembangkan sesuai kebutuhan klien dan memenuhi semua kebutuhan mereka.

2. Melakukan User Acceptance Testing (UAT) sangat penting karena memeriksa kinerja serta fitur perangkat lunak dan memvalidasi kualitasnya.
3. Dengan melakukan UAT ini untuk menguji keefektifan produk di dunia nyata seperti kondisi, yang selanjutnya mempersiapkan perangkat lunak untuk rilisnya.

### **7.6.3 Tools yang digunakan**

Untuk membuat proses UAT mudah dan nyaman bagi tim penguji dan untuk mendapatkan hasil yang cepat dan tepat, penguji perangkat lunak menggunakan berbagai alat pengujian. Tujuan dari alat ini adalah untuk menyederhanakan proses serta membuatnya hemat biaya. Demikian pula, pengujian penerimaan pengguna UAT juga dapat dijalankan dengan bantuan alat dan dengan mudah bekerja pada beberapa aspek perangkat lunak. Alat UAT ini sangat mudah digunakan dan juga sangat efektif. Beberapa alat yang digunakan untuk Pengujian Penerimaan Pengguna (UAT) disebutkan di bawah ini:

- a. Usersnap: Ini adalah solusi UAT yang mudah digunakan yang membantu tim QA untuk memverifikasi apakah solusi tertentu berfungsi untuk pengguna. Dengan memiliki widget umpan balik sederhana, penguji dapat memberikan umpan balik yang komprehensif tentang prototipe perangkat lunak. Selain itu, tim UAT dapat dengan mudah mengumpulkan dan menganalisis umpan balik kuantitatif dari penguji.
- b. Fitnesse Tool: Sebuah alat pengujian yang digunakan sebagai mesin pengujian. Sangat mudah untuk membuat tes dan mencatat hasil dalam tabel dengan bantuan alat Fitnesse. Pengguna alat memasukkan input yang diformat dan tes dibuat secara otomatis. Selain itu, pengujian kemudian dijalankan dan keluaran dikembalikan ke pengguna.

- c. Watir. Ini adalah kit alat yang digunakan untuk mengotomatiskan pengujian berbasis browser selama UAT. Ruby adalah bahasa pemrograman yang digunakan untuk komunikasi antar proses antara ruby dan internet explorer.

## DAFTAR PUSTAKA

- Abdul, K. (2005). *Pengenalan Sistem Informasi*. Yogyakarta: Andi Offset.
- Al Fattah, H. (2007). *Analisis Dan Perancangan Sistem Informasi Untuk Keunggulan Bersaing Perusahaan Dan Organisasi Modern*. Yogyakarta: Andi Offset.
- Alqudah, M., & Razali, R. (2016). A Review Of Scaling Agile Methods In Large Software Development. *International Journal On Advanced Science Engineering Information Technology*, 6(6), 828–837.
- Anggraeni, E. Y., & Irviani, R. (2017). *Pengantar Sistem Informasi*. Yogyakarta: Andi Offset.
- Bassil, Y. (2012). A Siniulation Model For The Spiral Software Development Life Cycle. *International Journal Of Innovative Research In Computer And Communication Engineering*, 2(05), 2049–3444. <https://doi.org/10.15680/Ijircce.2015.0305013>
- Bucanac, C. (1999). *The V-Model*. University Of Karlskrona/Ronneby. [http://www.bucanac.com/Documents/The\\_V-Model.Pdf](http://www.bucanac.com/Documents/The_V-Model.Pdf)
- Diana, A., & Setiawati, L. (2011). *Sistem Informasi Akuntansi, Perancangan, Prosedur Dan Penerapan*. Yogyakarta: Andi Offset.
- Eka Y. R, W., Bukhori, S., & Ismoyo, D. (2013). Perbandingan V-Model Tradisional Dan Advance V-Model. *Seminar Nasional Ilmu Komputer (Senaik)*.
- Ependi, U. (2018). Implementasi Model Scrum Pada Sistem Informasi Seleksi Masuk Mahasiswa Politeknik Pariwisata Palembang.

- Jurnal Informatika: Jurnal Pengembangan It (Jpit)*, 3(1), 49–55.
- Faizah, N., Santoso, N., & Soebroto, A. A. (2019). Pengembangan Sistem Aplikasi Manajemen Proyek Menggunakan Kanban Framework. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 3(10), 9747–9754.
- Graham, K. (2018). Techmatters: Getting On The “Kanban” -Wagon: Using Kanbanflow For Time And Project Management. *Loex Quarterly*, 43, 4–7.
- Gunadi, A., & Al Fatta, H. (2012). Anahsis Dan Pembuatan Game “Petualangan Si Argo” Berbasis Flash. *Data Manajemen Dan Teknologi Informasi (Dasi)*, 13(1), 42.
- Hall, J. A., & Singleton, T. (2007). *Information Technology Auditing And Assurance* (2nd Ed.). Jakarta: Salemba.
- Hammarberg, M., & Sunden, J. (2014). *Kanban In Action* (1st Ed.). Manning Publications.
- Hartono, B. (2013). *Sistem Informasi Manajemen Berbasis Komputer*. Rineka Cipta.
- Hartono M, J. (2005). *Analisa Dan Desain Sistem Informasi: Pendekatan Terstruktur Teori Dan Praktik Aplikasi Bisnis* (Andi (Ed.)). Yogyakarta: Andi Offset.
- Hutahaeen, J. (2015). *Konsep Sistem Informasi*. Deepublish.
- Indah, R., Mursityo, Y. T., & Saputra, M. C. (2018). Pengembangan Sistem Informasi Izin Kerja Dan Praktik Tenaga Kesehatan ( Sinkes ) Dinas Kesehatan Kota Xyz. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 2(12), 6176–6185.
- John W., S., Robert B., J., & Stephen D., B. (2012). *Systems Analysis And Design In A Changing World*. Joe Sabation.
- Karambir, & Thind, S. (2015). A Simulation Model For The Spiral Software Development Life Cycle. *International Journal Of Innovative Research In Computer And Communication*

- Engineering*, 03(05), 3823–3830. <https://doi.org/10.15680/Ijircce.2015.0305013>
- Kendall, K. E., & Kendall, J. E. (2010). *Analisis Dan Perancangan Sistem* (5th Ed.). Pt Indeks.
- Kenneth C., L., & Jane P., L. (2003). *Management Information Systems: Managing The Digital Firm* (8th Ed.). Prentice Hall.
- Kniberg, H., & Skarin, M. (2010). *Kanban And Scrum - Making The Most Of Both Enterprise Software Development Series Info Queue*. C4media Inc.
- Kotonya, G., & Sommerville, I. (1998). *Requirements Engineering: Processes And Techniques*. John Wiley And Sons.
- Kumar, D. (2019). *Software Engineering | Sdlc V-Model*. Geeksforgeeks. Org. <https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/?ref=lbp>
- Kusrini, & Koniyo, A. (2007). *Tuntunan Praktis Membangun Sistem Informasi Akuntansi Dengan Visual Basic Dan Microsoft Sql Server*. Andi.
- Larman, C., & Basili, V. R. (2003). Iterative And Incremental Development: A Brief History. *Computer*, 36(6), 47–56. <https://doi.org/10.1109/Mc.2003.1204375>
- Marry, P., & Tom, P. (2003). *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional.
- Mirzoyan, V. (2020). *Why Should You Use A Feature-Driven Development*. Aist.Global. <https://aist.global/en/use-a-feature-driven-development>
- Mohammed, N., Munassar, A., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. *International Journal Of Computer Science Issues*, 7(5), 94–101.
- Monden, Y. (1995). *Sistem Produksi Toyota:: Suatu Ancangan Terpadu Untuk Penerapan Just-In-Time* (2nd Ed.). Pustaka Binaman Pressindo.

- Mubarok, F., & Hadijah, I. (2015). Perbandingan Antara Metode Rup Dan Prototype Dalam Aplikasi Penerimaan Siswa Baru Berbasis Web. *Citec Journal*, 2(2), 114–127.
- Mulyadi. (2014). *Sistem Akuntansi* (5 (Ed.)). Andi.
- Mulyani, S. (2016). *Metode Analisis Dan Perancangan Sistem*. Abdi Sistematika.
- Mulyono, A. (2009). *Sistem Informasi Konsep Dan Aplikasi*. Pustaka Pelajar.
- Munir. (2008). *Kurikulum Berbasis Teknologi Informasi Dan Komunikasi*. Alfabeta.
- Mustakini. (2009). *Sistem Informasi Teknologi*. Andi Offset.
- Naufal, A., Jaffar, A., Yusoff, N., & Hayati, N. (2012). Development Of Kanban System At Local Manufacturing Company In Malaysia-Case Study. *Procedia Engineering*, 41(Iris), 1721–1726. <https://doi.org/10.1016/j.proeng.2012.07.374>
- Neilsen, J. (1993). *Usability Engineering*. Morgan Kaufmann Publishers Inc.
- Neilsen, J. (2012). *How Many Test Users In A Usability Study?* Nngroup. Com. <https://www.nngroup.com/articles/how-many-test-users/>
- Noor Santi, R. C. (2018). Perancangan Interaksi Pengguna (User Interaction Design) Menggunakan Metode Prototyping. *Jurnal Teknik Informatika*, 9(2), 108–113. <https://doi.org/10.15408/jti.v9i2.5599>
- Ogedebe, P. M., & Jacob, B. P. (2012). Software Prototyping: A Strategy To Use When User Lacks Data Processing Experience. *Arpn Journal Of Systems And Software*, 2(6), 219–224.
- Pal, S. K. (2018). *Software Engineering | Iterative Waterfall Model*. Geeksforgeeks.Org. <https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/?ref=lbp>

- Palmer, S. R., & Felshing, J. M. (2002). *A Practical Guide To Feature Driven Development*. Prentice Hall.
- Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach*. Mcgraw-Hill Education.
- Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach*. Mcgraw-Hill Education.
- Purnomo, D. (2017). Model Prototyping Pada Pengembangan Sistem Informasi. *Jurnal Informatika Merdeka Pasuruan*, 2(2), 54–61.
- Robiansyah, H., & Angreani, L. S. (2017). Sistem Informasi Mahasiswa Asing. *Matics: Jurnal Ilmu Komputer Dan Teknologi Infomrasi*, 9(1), 23–26.
- Sagala, J. R. (2014). Implementasi Sistem Aplikasi Belajar Rumus Matematika Metode Adaptive Software Development Pada Smp Swasta Methodist 7 Medan. *Mantik Penusa*, 15(1), 8–15.
- Saputra, A. (2012). Kajian Kebutuhan Perangkat Lunak Untuk Pengembangan Sistem Informasi Dan Aplikasi Perangkat Lunak Buatan Lapan Bandung. *Berita Dirgantara*, 13(2), 50–56.
- Saxena, S. (2019). *Software Engineering | Incremental Process Model*.Geeksforgeeks.Org. <https://www.geeksforgeeks.org/software-engineering-incremental-process-model/?ref=lbp>
- Schwaber, K., & Sutherland, J. (2012). *Software In 30 Days: How Agile Managers Beat The Odds, Delight Their Customers, And Leave Competitors In The Dust*. John Wiley & Sons.
- Sommerville, I. (2011). *Software Engineering* (9th Ed.). Pearson Education.
- Sugianto, Y., & Tjandra, S. (2016). Aplikasi Point Of Sale Pada Toko Retail Dengan. *Dinamika Teknologi*, 8(1), 1–8.
- Supriyanto, A. (2007). *Pengantar Teknologi Informasi*. Penerbit Salemba Infotek.

- Supriyatna, A. (2018). Metode Extreme Programming Pada Pembangunan Web Aplikasi Seleksi Peserta Pelatihan Kerja. *Jurnal Teknik Informatika*, 11(1), 1–18. <https://doi.org/10.15408/Jti.V11i1.6628>
- Susanto, A. (2000). *Sistem Informasi Manajemen Konsep Dan Pengembangannya*. Linggajaya.
- Susanto, R., & Andriana, A. D. (2016). Perbandingan Model Waterfall Dan Prototyping Untuk Pengembangan Sistem Informasi. *Majalah Ilmiah Unikom*, 14(1), 41–46.
- Sutabri, T. (2012). *Analisis Sistem Informasi*. Andi.
- Sutanta, E. (2003). *Sistem Informasi Manajemen*. Graha Ilmu.
- Syarifudin, A., & Ani, N. (2019). Perancangan Sistem Informasi Pengajuan Dan Pelaporan Pembayaran Tunjangan Kinerja Kementerian Keuangan Menggunakan Metode Prototype. *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, 8(2), 149. <https://doi.org/10.32736/Sisfokom.V8i2.641>
- Tatvasoft. (2015). *Top 12 Software Development Methodologies & Its Advantages / Disadvantages*. <https://www.tatvasoft.com/>.
- Tedja, R. T. (2019). *Software Development Model: Incremental Model*. Sis.Binus.Ac.Id. <https://sis.binus.ac.id/2019/07/02/software-development-model-incremental-model/>
- Yadav, R. S. (2012). Improvement In The V-Model. *International Journal Of Scientific & Engineering Research*, 3(1), 2229–5518.
- Yakub. (2012). *Pengantar Sistem Informasi*. Graha Ilmu.
- Yulianeu, A., & Noer, Z. M. (2016). Sistem Informasi Pengolahan Data Produksi Dan Distribusi Di Perusahaan Pabrik Tahu Jajang Suparman Js Kecamatan Cihaurbeuti Kabupaten Tasikmalaya. *Jurnal Manajemen Informatika (Jumika)*, 3(1).

### **Cacatan Revisi**

1. Tolong kata yang memakai bahasa inggris atau istilah asing, harus di italic kan.
2. Setelah titik/paragraph baru, jangan ada kata sambung atau kata keterangan tempat, seperti: Di..., Dan..., Yang...
3. Penomoran dan penamaan gambar tidak konsisten.
4. jarak paragraph terlalu jauh.
5. perlu adanya daftar istilah asing / kamus bahasa.
6. gambar terlalu besar, sehingga tidak efisien terhadap halaman.
7. Daftar isi dirubah semua halamannya, krn bergeser. Sesuaikan dengan isi bab.
8. Pada daftar pustaka, masih ada penerbit yang belum dicantumkan kota terbitnya.
9. ini belum ada indexing nya.
10. belum ada profil penulisnya.
11. revisilah sesuai point yang sudah sy berikan komentar pada setiap kalimat yang perlu direvisi.

terima kasih

Avin Wb



Lembaga Penelitian dan Pengabdian  
Kepada Masyarakat (LP2M)  
Intitut Agama Islam Negeri (IAIN) Salatiga

ISBN 978-602-5916-88-5



9 786025 916885