BAB II

LANDASAN TEORI

Pada bab ini dijelaskan mengenai teori-teori dan penelitian terdahulu yang terkait dalam pengerjaan tugas akhir ini.

2.1 Penelitian Terdahulu

2.1.1 Penelitian Kelompok Aktivitas Pengembangan Perangkat Lunak Sebelumnya

Aktivitas SDLC (*Software Development Life Cycle*) memiliki beberapa tahapan/aktivitas utama yaitu: analisis kebutuhan dan spesifikasi, arsitektur, desain, membangun dan menguji unit, serta uji integrasi sistem. Selain itu terdapat sumber daya manusia (tim proyek) yang memiliki peran penting untuk setiap aktivitas proyek meliputi: *Project Manager, Technical Architect, Business Analyst, Programers* dan *Testers* (Parthasarathy, 2007).

Aktivitas pengembangan perangkat lunak dikelompokkan menjadi tiga aktivitas utama dan ditambahkan effort disetiap aktivitasnya sebagai berikut (Shaleh, 2011):

a. Ongoing Activity

Aktivitas ini akan mengkoordinir aktivitas manajemen berupa manajemen proyek, manajemen konfigurasi proyek, dokumentasi proyek, penerimaan dan penyebaran proyek, terhitung 21 % dari total *effort*.

b. Software Development

Aktivitas ini akan mengkoordinir aktivitas pembangunan perangkat lunak berupa kebutuhan perangkat lunak, spesifikasi perangkat lunak, desain serta implementasi (pengkodean), terhitung 42 % dari total *effort*.

c. Quality & testing

Aktivitas ini merupakan aktivitas yang biasanya ada setelah dua aktivitas sebelumnya dikerjakan. Aktivitas ini akan mengkoordinir aktivitas pengujian terintegrasi, penjaminan kualitas serta evaluasi, terhitung 37 % dari total *effort*.

Untuk lebih jelasnya, presentase nilai *effort* tersebut dapat dilihat pada tabel 2.1. Seperti berikut :

Tabel 2. 1. Pembagian Kelompok Aktivitas Pembuatan Proyek

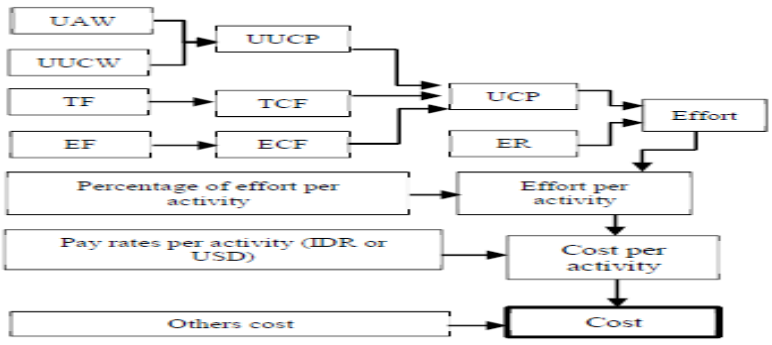
No	Kelompok Aktivitas	%Effort
1	**Software Development**	
a	Requirement	7,5%
b	Specification & Design	17,5%
c	Coding	10,0%
d	Integration Testing	7,0%
	Total	**42,0%**
2	**On Going Activity**	
a	Project Management	7,0%
b	Configuration Management	4,0%
c	Documentation	4,0%
d	Acceptance & Deployment	6,0%
	Total	**21,0%**
3	**Quality & Testing**	
a	Quality Assurance & Control	12,5%
b	Evaluation & Testing	24,5%
	Total	**37,0%**

Dari penelitian diatas, maka dapat disimpulkan bahwa pengelompokan aktivitas pengembangan yang perangkat lunak yang dilakukan oleh Shaleh terdapat 3 aktivitas utama yang masing-maing mempunyai segementasi peran dan presentase *effort* disetiap aktivitasnya. Namun dalam penelitian tersebut pengelompokan aktivitas pengembangan perangkat lunak yang dilakukan oleh Shaleh mempunyai segmentasi peran dan persentase effort untuk pengembangan perangkat lunak skala menengah ke besar, sedangkan dalam penelitian ini segmentasi peran dan persentase *effort* yang digunakan yaitu skala kecil ke menengah. Maka dari itu penelitian terdahulu tersebut perlu dilakukan penyesuaian terhadap penelitian yang akan dilakukan.

2.1.2 Estimasi Penentuan Harga Perkiraan Sendiri Untuk Pengembangan Perangkat Lunak Skala Kecil Menengah

Estimasi penentuan harga perkiraan sendiri (HPS) untuk pengembangan perangkat lunak skala menengah kecil-menengah mempunyai beberapa tahapan dengan menggunakan metode Use Case Point (UCP) dalam menentukan estimasi biaya langsung (Sholiq dkk, 2016).

Berikut ini merupakan gambar dari model estimasi penentuan harga perkiraan sendiri untuk pengembangan perangkat lunak



Gambar 2. 1. model estimasi penentuan harga perkiraan sendiri

Dari penelitan yang dilakukan oleh (Sholiq dkk, 2016) tidak mengunakan aktivitas secara khusus dalam pengembangan perangkat lunak, dan juga dalam penentuan aktifitas effort dilakukan perhitungan satu dimensi dengan berdasarkan aktivitas yang ada.

2.2 Teori Pendukung

2.2.1 Harga Perkiraan Sendiri (HPS)

HPS merupakan total harga yang diperkirakan cukup untuk membiayai pekerjaan yang akan dilaksanakan dalam pengadaan barang/jasa, dan ditetapkan oleh PPK, kecuali untuk Kontes/Sayembara.

Nilai total HPS sebagaimana tersebut diatas adalah hasil perhitungan seluruh volume pekerjaan dikalikan dengan Harga Satuan ditambah dengan seluruh beban pajak dan keuntungan. HPS disusun paling lama 28 (dua puluh delapan) hari kerja sebelumbatas akhir pemasukan penawaran. Dalam proses pemilihan penyedia barang/jasa menggunakan HPS sebagai :

1. Alat untuk menilai kewajaran harga penawaran termasuk rinciannya,
2. Dasar untuk menetapkan batas tertinggi penawaran yang sah untuk Pengadaan Barang/Pekerjaan Konstruksi/Jasa Lainnya dan Pengadaan Jasa Konsultansi yang menggunakan metode evaluasi Pagu Anggaran,
3. Dasar untuk menetapkan besaran nilai Jaminan Pelaksanaan bagi penawaran yang nilainya lebih rendah dari 80% (delapan puluh perseratus) nilai total HPS.

HPS bukan sebagai dasar untuk menentukan besaran kerugian negara. Penyusunan HPS didasarkan pada data harga pasar setempat, yang diperoleh

berdasarkan hasil survei menjelang dilaksanakannya pengadaan, dengan mempertimbangkan informasi yang meliputi :

1. Informasi biaya satuan yang dipublikasikan secara resmi oleh Badan Pusat Statistik (BPS),
2. Informasi biaya satuan yang dipublikasikan secara resmi oleh asosiasi terkait dan sumber data lain yang dapat dipertanggung jawabkan,
3. Daftar biaya/tarif barang/jasa yang dikeluarkan oleh pabrikan/distributor tunggal dan instansi yang berwenang,
4. Biaya kontrak sebelumnya atau yang sedang berjalan dengan mempertimbangkan faktor perubahan biaya,
5. Inflasi tahun sebelumnya, suku bunga berjalan dan/atau kurs tengah Bank Indonesia,
6. Hasil perbandingan dengan Kontrak sejenis, baik yang dilakukan dengan instansi lain maupun pihak lain,
7. Perkiraan perhitungan biaya yang dilakukan oleh konsultan perencana (engineer’s estimate),
8. Norma indeks; dan/atau,
9. Informasi lain yang dapat dipertanggung jawabkan.

HPS disusun dengan memperhitungkan keuntungan dan biaya overhead yang dianggap wajar, antara lain untuk kebutuhan dalam penerapan manajemen Keselamatan dan Kesehatan Kerja (K3) serta penerapan manajemen mutu (LKPP, 2012).

Komponen Harga Perkiraan Sendiri (HPS) untuk pengembangan perangkat lunak terdiri dari biaya-biaya, sebagai berikut :

1. Biaya Lagsung Personil (Remunerasi)
    a. Biaya untuk pengadaan tenaga ahli (*Professional staff*), asisten tenaga ahli (*sub professional staff*) dan tenaga pendukung (*supporting staff*);
    b. Biaya Langsung personil dihitung berdasarkan jumlah orang bulan (*man month*) dari masing-masing tenaga ahli, asisten tenaga ahli, dan tenaga pendukung yang diperlukan;
    c. Harga Satuan untuk biaya tenaga ahli, asisten tenaga ahli ditentukan berdasarkan tingkat pendidikan dan lamanya pengalaman sesuai bidang keahlian masing-masing tenaga ahli, dan mengikuti harga pasar (LKKP, 2012).

Perhitungan Konersi Minuimum Biaya Langsung Personil menurut satuan waktu adalah sebagai berikut :

SBOM = SBOB / 4.1

SBOH = (SBOB / 22) x 1.1

SBOJ = (SBOH / 8) x 1.3

Catatan :

SBOB = Satuan Biaya Orang Bulan (*Person Month Rate*)

SBOM = Satuan Biaya Orang Minggu (*Person Week Rate*)

SBOH = Satuan Biaya Orang Hari (*Person Day Rate*)

SBOJ = Satuan Biaya Orang Jam (*Person Hour Rate*)

Perhitungan Biaya Langsung Personil (BLP) dilakukan sebagai berikut :

BLP = GD + BBS + BBU + T + K

Dimana :

GD = Gaji Dasar (*Basic Salary*)

BBS = Beban Biaya Sosial (*Social Cost*)

BBU = Beban Biaya Umum (*Overhead Cost*)

T = Tunjangan (*Allowance*)

K = Keuntungan (Profit) (Inkindo, 2014).

2. Biaya Langsung Non Personil

Biaya Langsung Non Personil adalah biaya lansung yang diperlukan untuk menunjang pelaksanaan kegiatan proyek yang dibuat dengan mempertimbangkan dan berdasarkan Harga Pokok Pasar yang wajar dan dapat dipertanggungjawabkan sera sesuai dengan perkiraan kegiatan. Biaya Langsung Non Personil ini terdiri dari 3 (tiga) komponen yaitu :

a. *Reimbusable* adalah biaya yang dapat diganti yang sebenarnya dikeluarkan oleh konsultan untuk pengeluaran-pengeluaran yang sesungguhnya (*at cost*) dan kegiatan yang ditetapkan, seperti :

   1. Dokumen Perjalanan ke Luar Negeri
   2. Tiket Penerbangan
   3. Kelebihan Bagasi (*Excess Baggage*)
   4. Bagasi yang Tidak Dibawa Sendiri (*Unaccompanied Baggage*)
   5. Biaya Perjalanan Darat (*Local / Inland Travel*)
   6. Biaya Pembelian Kebutuhan Proyek
   7. Biaya Instalasi Telepon / Internet

b. *Fixed Unit Rate* adalah biaya yang sebenarnya dikeluarkan oleh konsultan berdasarkan harga satuan yang pasti dan tetap untuk setiap item/unsur pekerjaan dengan volume yang diperkirakan, seperti :

   1. Sewa Kendaraan dan O&M

2. Sewa Kantor Proyek
3. Sewa Peralatan Kantor
4. Sewa Furniture Kantor
5. Biaya Operasional Kantor Proyek
6. Biaya ATK (*Office Consumables*)
7. Biaya Komputer & *Printer Consumables*
8. Biaya Komunikasi
9. Tunjangan Harian (*Per Diem Allowance*)
10. Tunjangan Perumahan (*Housing Allowance*)
11. Penempatan Sementara (*Temporary Lodging*)
12. Tunjangan Penempatan (*Relocation Allowance*)
13. Tunjangan Tugas Luar (*Out of Station Allowance / OSA*)
14. Penginapan Tugas Luar
15. Cuti Tahunan (*Annual Leave*)
16. Biaya Pelaporan

c. *Lump Sum* adalah biaya satuan atau beberapa item/unsur pekerjaan dalam batas waktu teretentu, dengan jumlah harga yang pasti dan tetap serta dibayarkan sekaligus, seperti :

1. Pengumpulan Data Sekunder
2. Seminar, *Workshop*, Sosialisasi, *Training*, Desiminasi, Loka Karya, Diskusi, Koordinasi antar Instansi, FGD (*Focus Group Discussion*)
3. *Survey*
4. Biaya Test Laboratorium

5. dst. nya

3. Pajak Pertambahan Nilai (PPN) (Inkindo, 2014).

2.2.2 Estimasi *Effort*

Salah satu aspek terpenting dalam tahapan perencanaan adalah melakukan estimasi atau perkiraan, baik dari segi biaya, waktu maupun sumber daya. Definisi dari estimasi adalah sebuah pengukuran yang didasarkan pada hasil secara kuantitatif atau dapat diukur dengan angka tingkat akurasinya (Tockey, 2004).

Sisi penting estimasi dalam perencanaan proyek adalah munculnya jadwal serta anggaran yang tepat, meski tidak sepenuhnya sebuah estimasi akan berakhir dengan tepat. Tetapi, tanpa sebuah estimasi dalam pelaksanaan proyek perangkat lunak maka dapat dikatakan bahwa proyek perangkat lunak tersebut adalah sebuah blind project. Yang diibaratkan seperti seorang buta yang harus berjalan di sebuah jalan raya yang sangat ramai (Rizky, 2011).

Estimasi yang dilakukan pada penelitian ini diaplikasikan dalam proyek pengembangan perangkat lunak kepemerintahan skala kecil menengah dengan model *prototype*. Definisi dari estimasi perangkat lunak yaitu suatu kegiatan melakukan prediksi atau ramalan mengenai keluaran dari sebuah proyek dengan meninjau jadwal, usaha, biaya bahkan hingga ke resiko yang akan ditanggung dalam proyek tersebut (Galorath, 2006). Meski estimasi tidak mungkin dapat menghasilkan sebuah hasil yang sangat akurat, tetapi ketidakakuratan tersebut dapat diminimalkan dengan menggunakan beberapa metode yang sesuai dengan proyek yang akan dilakukan estimasi.

Effort (usaha) dari sebuah proyek pengembangan perangkat lunak dapat didefinisikan sebagai waktu yang dikonsumsi oleh proyek yang dinyatakan

dengan hitungan orang dalam jam, hari, bulan atau tahun tergantung pada ukuran proyek, sebagai contoh adalah *effort = people * time* (Chatters,1999) dalam (Haapio, 2011).

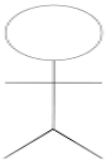
Dari pengertian estimasi dan effort di atas, maka dapat disimpulkan bahwa estimasi effort adalah suatu kegiatan melakukan prediksi atau ramalan mengenai berapa banyak pekerja dan berapa lama waktu yang diperlukan untuk menyelesaikan proyek tersebut. Estimasi effort pada penelitian ini akan didapatkan setelah melakukan perhitungan menggunakan metode use case point (UCP).

2.2.3. *Use Case Diagram*

Use Case Diagram menggambarkan sekumpulan *use case* dan aktor serta hubungannya (Booch et al, 1999, p234). Yang ditekankan adalah "apa" yang dilakukan terhadap sistem dan bukan "bagaimana". Sebuah *use case* menggambarkan interaksi antara aktor dengan sistem. Dibawah ini dijelaskan bagian *use case diagram* :

a. Aktor

Aktor adalah segala sesuatu yang melakukan tatap muka dengan sistem, seperti orang, piranti lunak, piranti keras, atau jaringan (Schneider dan Winters, 1997, p12). Tiap-tiap aktor menunjukkan perannya masing-masing. Contohnya, seorang actor dapat memberikan *input* ke dalam dan menerima informasi dari aplikasi piranti lunak. Notasi aktor dengan nama aktor tersebut dibawahnya:



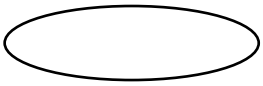
Pengguna

b. *Use Case*

Use Case menggambarkan segala sesuatu yang aktor ingin lakukan terhadap sistem.

Use Case harus merupakan "apa" yang dikerjakan piranti, bukan "bagaimana" aplikasi piranti lunak mengerjakannya. Suatu sistem yang kompleks memiliki banyak *use case*, sehingga perlu diorganisasi.

Notasi *use case* :



Untuk menghubungkan antara aktor dengan *use case* digunakan simbol garis yang disebut sebagai *relationship*. Dengan adanya sebuah *use case diagram* maka akan membantu dalam menyusun kebutuhan sebuah sistem dan mengkomunikasikannya dengan klien.

2.2.4. *Use Case Point* (UCP)

Metode *use case point* (UCP) adalah metode yang mempunyai kemampuan untuk memberikan estimasi effort yang diperlukan untuk membuat suatu proyek berdasarkan jumlah dan kompleksitas *use case* yang dimiliki oleh proyek perangkat lunak tersebut (Karner, 1993). Menurut pendapat lain, UCP adalah metode yang dapat menganalisa *actor*, *use case*, dan berbagai faktor teknis dan faktor lingkungan hingga menjadi suatu persamaan (Clemmons, 2006).

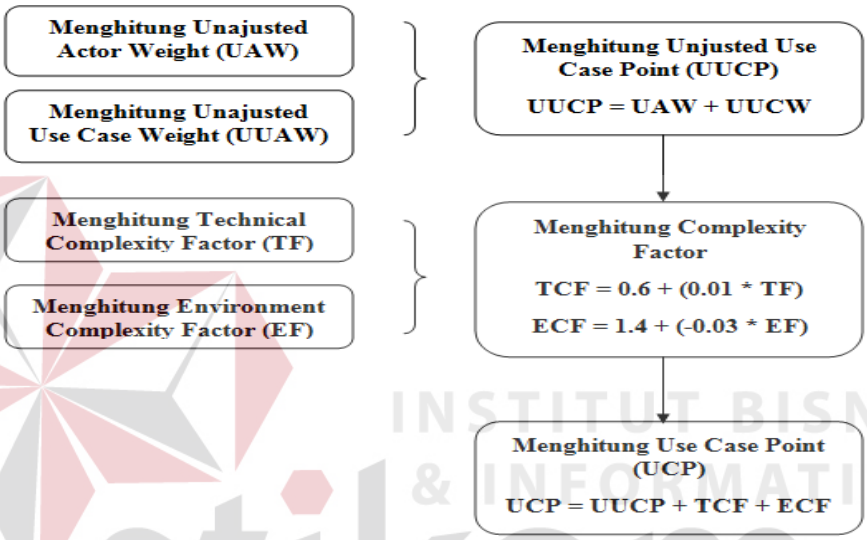
Kelebihan dari metode use case point yaitu dapat memberikan estimasi yang hampir mendekati estimasi sebenarnya yang dihasilkan dari pengalaman pembuatan atau pengembangan software. Hal tersebut dibuktikan oleh beberapa penelitian yang pernah dilakukan sebelumnya, dan menghasilkan pernyataan sebagai berikut:

a. UCP memiliki deviasi sebesar 6% (Nageswaran, 2001),

b. UCP memiliki deviasi sebesar 19%, sementara estimasi para ahli memiliki deviasi sebesar 20% (Anda, 2002),

c. UCP memiliki deviasi sebesar 9% (Carroll, 2005).

Langkah-langkah yang dilakukan dalam proses estimasi *effort* dengan *use case point* digambarkan dalam gambar 2.1. berikut ini (Karner, 1993) :



Gambar 2. 2 Langkah-langkah Metode *Use Case Point* (UCP)

1. *Unjusted Use Case Point* (UUCP)

a. Unadjusted Actor Weights *(UAW)*

Langkah pertama adalah menentukan terlebih dahulu aktor sebagai *simple*, *average*, atau *complex* sesuai tabel 2.2. seperti berikut:

Tabel 2. 2. Tipe, Bobot, dan Deskripsi *Actor*

Actor	Weight	Description
Simple	1	Didefinisikan degan API
Medium	2	Berinteraksi dengan protokol TCP/IP
Complex	3	Berinteraksi dengan GUI atau Web Page

1. Application Program Interface (API)

API adalah singkatan dari Application Program Interface, yakni serangkaian instruksi dan standar pemrograman untuk mengakses aplikasi atau layanan berbasis web. Sebuah perusahaan software atau penyedia layanan berbasis web merilis API mereka kepada publik. Dengannya, pengembang lain dapat mendesain aplikasi yang memanfaatkan layanan mereka.

API memungkinkan sebuah aplikasi berkomunikasi dengan aplikasi lain di Internet melalui serangkaian panggilan (call). Sebuah API, berdasarkan definisinya, adalah sesuatu yang mendefinisikan cara dua entitas untuk berkomunikasi. Entitas di sini adalah sebuah software yang nyata berbeda (dalam layanan) dengan software lain.

Dengan API, panggilan-panggilan yang bolak-balik antar aplikasi diatur melalui web service. Web service adalah kumpulan standar teknis dan protokol, termasuk XML (Extensible Markup Language), bahasa umum yang digunakan oleh aplikasi-aplikasi tersebut selama berkomunikasi di Internet.

API sendiri merupakan sekumpulan kode software yang ditulis sebagai serangkaian pesan XML. Setiap pesan XML berhubungan dengan fungsi spesifik dari aplikasi yang akan diajak berkomunikasi. Sebagai contoh, pada API Facebook, terdapat pesan XML yang berhubungan dengan fungsi spesifik wall post, wall comment, wall like.

Pengembang aplikasi pihak ketiga menggunakan pesan-pesan XML yang berhubungan dengan fungsi-fungsi spesifik dari layanan web yang akan diajak berkomunikasi. Pengembang bebas memilih fungsi khusus apa saja yang akan diajak berkomunikasi, dan ditampilkan pada aplikasi rancangannya. Sebagai

contoh, kita bisa membuat Facebook client yang hanya menampilkan update status teman-teman kita.

Dengan demikian, API adalah standar komunikasi yang dibuka oleh perusahaan software, agar dapat dimanfaatkan oleh pengembang pihak ketiga untuk mendesain aplikasi yang memanfaatkan layanan mereka dengan mudah.

2. Transmission *Control* Protocol/Internet Protocol (TCP/IP)

TCP/IP (singkatan dari Transmission *Control* Protocol/Internet Protocol) adalah standar komunikasi data yang digunakan oleh komunitas internet dalam proses tukar-menukar data dari satu komputer ke komputer lain di dalam jaringan *Internet*. Protokol ini tidaklah dapat berdiri sendiri, karena memang protokol ini berupa kumpulan protokol (protocol suite). Protokol ini juga merupakan protokol yang paling banyak digunakan saat ini. Data tersebut diimplementasikan dalam bentuk perangkat lunak (*software*) di sistem operasi. Istilah yang diberikan kepada perangkat lunak ini adalah TCP/IP *stack.*

Protokol TCP/IP dikembangkan pada akhir dekade 1970-an hingga awal 1980-an sebagai sebuah protokol standar untuk menghubungkan komputer-komputer dan jaringan untuk membentuk sebuah jaringan yang luas. TCP/IP merupakan sebuah standar jaringan terbuka yang bersifat independen terhadap mekanisme *transport* jaringan fisik yang digunakan, sehingga dapat digunakan di mana saja. Protokol ini menggunakan skema pengalamatan yang sederhana yang disebut sebagai alamat IP 6.

3. Graphical User Interface (GUI)

Pengertian GUI adalah Graphical User Interface dalam dunia komputer. Pada komputer terdapat GUI atau antarmuka pengguna secara grafis. Istilah ini

bukan hal yang lumrah pada saat awal kemunculan komputer. Namun setelah komputer generasi keempat mulai diciptakan, munculnya televisi berwarna (yang mendorong pada penciptaan layar monitor berwarna) serta evolusi pada perangkat penampil gambar (graphic adapter atau graphic card atau video card) membuat komputer mulai mendapatkan suatu sistem baru.

Secara sederhana, GUI adalah suatu media virtual yang dapat membuat pengguna memberikan perintah tertentu pada komputer tanpa mengetik perintah tersebut, namun menggunakan gambar yang tersedia. Pengguna tidak mengetikkan perintah seperti pada komputer dengan Shell atau teks. Dengan GUI, perintah dapat dikonversi menjadi ikon dalam layar monitor yang dapat diklik untuk memulai fungsinya. Sebagai contoh, tentu anda paham dengan sebuah ikon berbentuk kertas dengan huruf W diatasnya kan? Itu adalah ikon untuk menjalankan Microsoft Word, sebuah aplikasi yang digunakan untuk mengetik. Atau anda pasti familiar dengan tombol di pojok kiri bawah, yakni tombol bertuliskan Start atau logo Windows itu. Segala sesuatu yang anda lihat di Komputer anda saat ini adalah GUI.

Total *Unadjusted Actor Weights* (UAW) didapat dari menghitung jumlah *actor* dari masing-masing jenis (tingkat kompleksitas), dikali dengan total faktor berat masing-masing sesuai dengan tabel.

2. Unadjusted Use Case Weights *(UUCW)*

Cara menghitung UUCW sama dengan cara menghitung UAW, yaitu masing-masing use case dibagi menjadi 3 kelompok yaitu *simple*, *average*, dan *complex*, tergantung dari jumlah transaksi yang dilakukan. Untuk penjelasan lebih detil tentang deskripsi *use case* dapat dilihat pada tabel 2.3. seperti berikut :

Tabel 2. 3. Tipe, Bobot, dan Deskripsi *Use Case*

Use Case	Weight	Description
Simple	5	Menggunakan <=3 transaksi
Medium	10	Menggunakan 4 sampai 7 transaksi
Complex	15	Menggunakan >7 transaksi

Total *Unadjusted Use Case Weights* (UUCW) didapat dari menghitung jumlah *use case* dari masing-masing tingkat kompleksitas dikali dengan total faktor setiap use case. Kemudian jumlahkan UAW dan UUCW untuk mendapatkan *Unadjusted Use Case Point* (UUCP), seperti rumus berikut :

$$UUCP = UAW + UUCW$$

3. Menghitung Technical Complexity Factor (TCF) dan Enviromental Complexity Factor (ECF)

Pada perhitungan nilai *Use Case Point* (UCP) terdapat nilai *complexity factor*. Pengertian dari *complexity factor* adalah faktor-faktor yang berpengaruh secara langsung dalam proses pengerjaan proyek perangkat lunak tersebut. *Complexity factor* dibagi menjadi 2 kelompok, yaitu :

a. *Technical Complexity Factor* (TCF)

b. *Environmental Complexity Factor* (ECF)

Berikut penjelasan masing-masing dari *complexity factor* :

a. Technical Complexity Factor (TCF)

Tabel 2. 4. *Technical Factor* dan Bobot

Technical Factor		Bobot
1	Distributed System Required	2
2	Response time is Important	1
3	End User Effficiency	1
4	Complex Internal Processing Required	1
5	Reusable Code Must Be a Focus	1
6	Installation Easy	0.5

Technical Factor		Bobot
7	Usability	0.5
8	Cross-Platform Support	2
9	Easy to Change	1
10	Highly Concurrent	1
11	Custom Security	1
12	Dependence on Third-part Code	1
13	User Training	1

Nilai-nilai pada *technical factor* tersebut dikalikan dengan bobot nilai masing-masing. Bobot nilai yang diberikan pada setiap *factor* tergantung dari seberapa besar pengaruh dari factor tersebut. 0 berarti tidak mempengaruhi, 3 berarti rata-rata, dan 5 berarti memberikan pengaruh yang besar. Hasil perkalian nilai dan bobot tersebut kemudian dijumlahkan untuk mendapatkan total *Technical Factor* (TF), yang kemudian digunakan untuk mendapatkan *Technical Complexity Factor* (TCF).

$$TCF = 0.6 + (0.01*TF)$$

b. *Environmental Complexity Factor (ECF)*

Tabel 2. 5. *Environmental Factor* dan Bobot

Environmental Factor		Bobot
1	Familiarity with the Project	1.5
2	Application Experience	0.5
3	OO Programming Experience	1
4	Lead Analyst Capability	0.5
5	Motivation	1
6	Stable Requirements	2
7	Part Time Staff	-1
8	Difficult Programming Language	-1

Nilai-nilai pada *environmental factor* tersebut dikalikan dengan bobot nilai masing-masing. Bobot nilai yang diberikan pada setiap factor tergantung dari seberapa besar pengaruh dari faktor tersebut. 0 berarti tidak mempengaruhi, 3

berarti rata-rata, dan 5 berarti memberikan pengaruh yang besar. Hasil perkalian nilai dan bobot tersebut kemudian dijumlahkan untuk mendapatkan total *Environmental Factor* (EF), yang kemudian digunakan untuk mendapatkan *Environmental Complexity Factor* (ECF).

$$ECF = 1.4 + (-0.03x\ EF)$$

Sehingga akhirnya kita bisa mendapatkan nilai dari *Use Case Point* (UCP) yang didapatkan melalui perkalian UUCP, TCF, dan ECF.

$$UCP = UUCP * TCF * ECF$$

2.2.5. Perhitungan Nilai *Effort Rate*

Effort rate didefinisikan sebagai jumlah usaha per *use case point*. Pendekatan yang dijelaskan bersifat umum dan dapat digunakan untuk menganalisa berbagai data, tidak hanya data untuk pengembangan perangkat lunak, tetapi juga data pemeliharaan perangkat lunak dan jenis lain dari rekayasa perangkat lunak (Stewart, 2002).

Effort rate adalah rasio jumlah jam orang per *use case point* berdasarkan proyek-proyek di masa lalu. Jika proyek tersebut merupakan proyek baru dan tidak terdapat data histori yang telah terkumpul, maka digunakan nilai yang berkisar antara 15 sampai 30. Namun, nilai yang paling sering dipakai adalah angka 20 (Clemmons, 2006).

Rumus perhitungan *estimasi effort* menggunakan metode UCP adalah sebagai berikut :

$$Estimasi\ Effort = UCP \times ER$$

Apabila nilai ER dihitung dari satu proyek saja maka nilai ER didapatkan dari pembagian antara nilai *actual effort* dengan nilai UCP, sebagai berikut :

Effort Rate = Actual Effort/UCP

2.2.6. Perangkat Lunak

Komputer atau perangkat keras dapat beroperasi mengikuti instruksi manusia secara persis melalui perangkat lunak. Perangkat lunak sendiri merupakan kumpulan program komputer yaitu dalam bentuk sistem pengoperasian komputer yang berbentuk instruksi tertulis dalam bahasa komputer (Amsyah, 2005).

2.2.7. Skala Perangkat Lunak

Ukuran atau skala dari proyek perangkat lunak diperkirakan berdasarkan beberapa parameter, yaitu jumlah programmer, durasi waktu penyelesaian, dan jumlah baris kode (Donna, 2006). Kategori dalam ukuran proyek perangkat lunak dapat dilihat pada tabel 2.6 sebagai berikut :

Tabel 2. 6. Ukuran Proyek Perangkat Lunak

No	Category	∑Programmer	Time Required	∑Lines
1	Trivial	1	1-4 week	500
2	Small	1	1-6 month	1K-2K
3	Medium	2-5	1-2 year	5K-50K
4	Large	5-20	2-3 year	50K-100K
5	Very Large	100-1K	4-5 year	1M
6	Extra Large	2K-5K	5-10 year	1M-10M

2.2.8 *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) adalah suatu kerangka yang menggambarkan kegiatan-kegiatan yang dilakukan pada setiap tahap pembuatan sebuah *software* (Fatta, 2007: 24). Terdapat banyak metode untuk mendeskipsikan

SDLC ini, pada dasarnya setiap metode menggambarkan tahap-tahap sebagai berikut:

1. Identifikasi, seleksi dan perencanaan

Tahap ini merupakan tahap *preliminary* dari pembuatan suatu *software*. Pada tahap ini, dikembangkan suatu rancang bangun dari suatu *software*. Langkah-langkah yang dilakukan dalam tahap ini antara lain.

a) Mengidentifikasi kebutuhan *user*.
b) Menyeleksi kebutuhan *user* dari proses identifikasi diatas, dengan menyesuaikan dengan kapasitas teknologi yang tersedia serta efisiensi.
c) Merencanakan sistem yang akan digunakan pada *software* yang dibuat, Dengan kebutuhan-kebutuhan sebagai berikut: kebutuhan fungsional dan non-fungsional, kebutuhan *user*, kebutuhan sistem, kebutuhan dokumen dan perangkat lunak.

2. Analisis sistem

Tahap ini merupakan tahap penyempurnaan, yang bertujuan memperoleh kebutuhan *software* dan *user* secara lebih spesifik dan rinci. Tujuan dilakukan tahap ini adalah untuk mengetahui posisi dan peranan teknologi informasi yang paling sesuai dengan kebutuhan perusahaan yang bersangkutan, serta mempelajari fungsi-fungsi manajemen dan aspek-aspek bisnis terkait yang akan berpengaruh atau memiliki dampak tertentu terhadap proses desain, konstruksi dan implementasi *software*. Analisis sistem terbagi dua, yaitu.

a. Permodelan data, yang mencakup *Entity Relationship Diagram* (ERD), *Conceptual Data Model* (CDM), dan *Physical Data Model* (PDM).
b. Permodelan proses, dengan *Unified Modeling Language*.

3. Desain sistem

Setelah melakukan identifikasi serta analisis sistem, tahap selanjutnya adalah menerjemahkan konsep-konsep tersebut kedalam suatu sistem yang berwujud. Tahap ini meliputi pembuatan dan pengembangan sebagai berikut.

a. Desain form dan laporan (*reports*).
b. Desain antarmuka dan dialog (*message*).
c. Desain basis data dan *file* (*framework*).
d. Desain proses (*process structure*).

Pada tahap ini akan dihasilkan sebuah dokumen berupa *Software Architecture Document* (SAD). SAD ini adalah dokumen yang menjelaskan tentang arsitektur proyek perangkat lunak yang berhubungan dengan *project*.

4. Implementasi sistem

Tahap implementasi sistem ini diawali dengan pengetesan *software* yang telah dikembangkan. Beberapa tahap pengetesan adalah sebagai berikut.

a. *Developmental*, yakni pengetesan *error* per *module* oleh *programmer*.
b. *Alpha testing*, yakni *error testing* ketika *software* digabungkan dengan antarmuka *user*.
c. *Beta testing*, yakni pengetesan dengan lingkungan dan data yang sebenarnya.

5. Pemeliharaan sistem

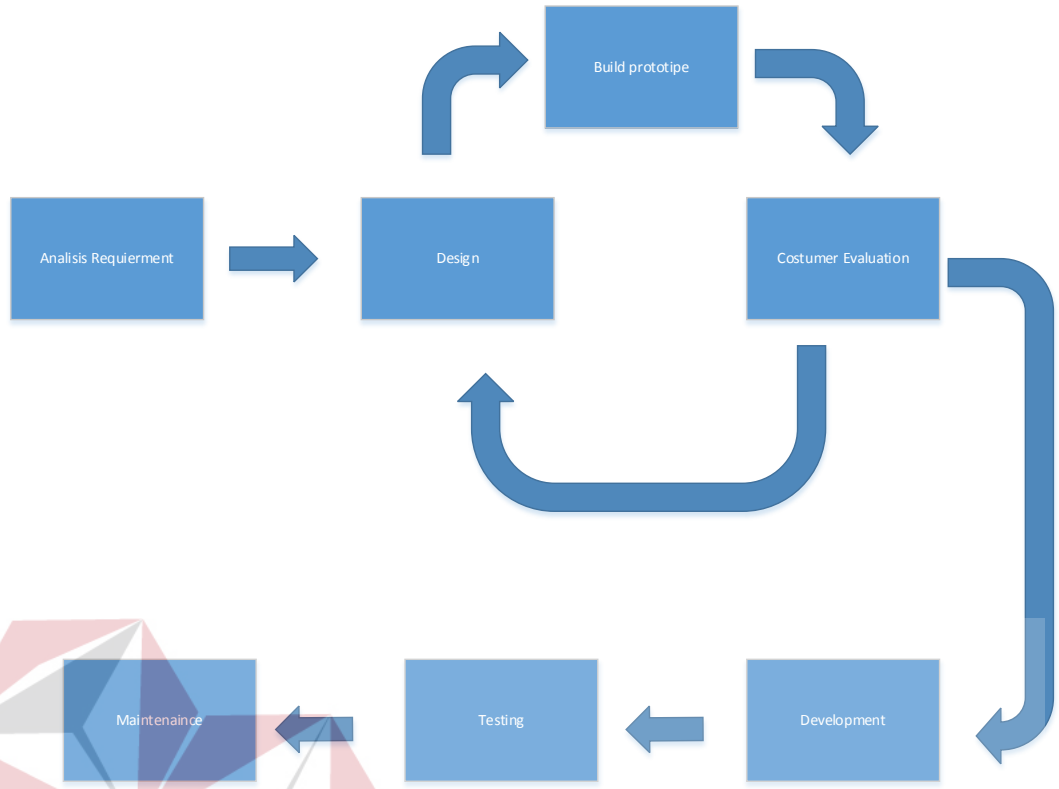
Tahap pemeliharaan sistem adalah sebagai berikut.

a. *Korektif*, yaitu memperbaiki desain dan *error* pada program (*troubleshooting*).

b. *Adaptif*, yaitu memodifikasi sistem untuk beradaptasi dengan perubahan lingkungan.
c. *Perfektif*, yaitu melibatkan sistem untuk menyelesaikan masalah baru atau menambah fitur baru pada sistem yang telah ada.
d. *Preventif*, yaitu menjaga sistem dari kemungkinan masalah di masa yang akan datang.

2.2.9 Prototipe Model

Merupakan model pengembangan system yang proses iterative dalam pengembangan sistem dimana requirement diubah ke dalam sistem yang bekerja (working system) yang secara terus menerus diperbaiki melalui kerjasama antara user dan analis. *Prototype* juga bisa dibangun melalui beberapa tool pengembangan untuk menyederhanakan proses. Prototyping merupakan bentuk dari Rapid Application Development (RAD). Dalam metode ini, pengembang dan pelanggan dapat saling berinteraksi selama proses pembuatan system (Pressman, 2002).



Gambar 2. 3 Tahapan model prototype pengembangan perangkat lunak

Tahapan-tahapan dalam Prototyping adalah sebagai berikut:

1. Analisis Requierment

   Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

2. Design

   Pengembang membuat design sesuai dengan hasil analisis sesuai dengan kebutuhan pelanggan agar lebih mudah dalam pembuatan prototipe untuk pelanggan.

3. Build Prototipe

   Membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat input dan format output)

4. Costumer Evalution

   Evaluasi ini dilakukan oleh pelanggan apakah prototyping yang sudah dibangun sudah sesuai dengan keinginann pelanggan. Jika sudah sesuai maka langkah 5 akan diambil. Jika tidak prototyping direvisi dengan mengulang langkahb 1, 2, dan 3.

5. Mengkodekan sistem

   Dalam tahap ini prototyping yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai

6. Menguji system

   Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan. Pengujian ini dilakukan dengan White

7. Maintaince

   Setelah aplikasi diterapkapan oleh perusahaan maka pihak pengembang akan melakukan pemeliharaan aplikasi agar dapat memperbaiki sistem apabila terjadi kerusakan atau ketidaksesuain sistem.

Keunggulan dan kelemahan model prototyping

Keunggulan prototyping adalah :

1. Adanya komunikasi yang baik antara pengembang dan pelanggan
2. Pengembang dapat bekerja lebih baik dalam menentukan kebutuhan pelanggan

3. Pelanggan berperan aktif dalam pengembangan sistem
4. Lebih menghemat waktu dalam pengembangan sistem
5. Penerapan lebih mudah karena pemakai mengetahui apa yang diharapkanya

Kelemahan prototyping adalah :

1. Pelanggan terkadang tidak mengetahui atau menyadari bahwa perangkat lunak yang ada belum mencantumkan kualitas perangkat lunak secara keseluruhan dan juga belum memikirkan kemampuan pemeliharaan untuk jangka waktu yang lama.
2. Pengembang biasanya ingin cepat menyelesaikan proyek sehingga menggunakan algoritma dan bahasa pemrograman yang sederhana untuk membuat prototyping lebih cepat selesai tanpa memikirkan lebih lanjut bahwa program tersebut hanya cetak biru sistem.
3. Hubungan sistem dengan komputer yang disediakan mungkin tidak mencerminkan teknik perancangan yang baik.

2.2.10. Manajemen Proyek

Manajemen proyek adalah suatu teknik yang digunakan untuk merencanakan, mengerjakan, dan mengendalikan aktivitas suatu proyek untuk memenuhi kendala waktu dan biaya proyek (Muslich, 2009). Teknik ini berorientasi pada pencapaian tujuan, di mana tujuan tersebut mungkin pembangunan gedung, pembukaan kantor baru, atau pengendalian kegiatan penelitian dan pengembangan. Perencanaan suatu proyek terdiri dari tiga tahap (Prasetya, Hery dan Lukiastuti, Fitri 2009), yaitu:

a. Perencanaan. Membuat uraian kegiatan-kegiatan, menyusun logika urutan kejadian-kejadian, menentukan syarat-syarat pendahuluan, menguraikan interaksi dan interdependensi antara kegiatan-kegiatan.
b. Penjadwalan. Penaksiran waktu yang diperlukan untuk melaksanakan tiap kegiatan, menegaskan kapan suatu kegiatan berlangsung dan kapan berakhir.
c. Pengendalian. Menetapkan alokasi biaya dan peralatan guna pelaksanaan tiap kegiatan.

2.2.11 Analisis Devisiasi Rata-Rata

Deviasi rata-rata (*Mean Deviation* atau *Average Deviation*) adalah rata-rata dari deviasi nilai-nilai dari *mean* dala suatu distribusi, diambil nilainya yang *absolute*. Yang dimaksut dengan deviasi *absolute* adalah nilai-nilai yang negatif. Secara aritmatika *mean* deviasi dapat didefinisikan sebagai *mean* dari harga mutlak dari deviasi nilai-nilai individual.

Yang Pertama dilakukan adalah menghitung *mean*, kemudian ditentukan berapa besarnya penyimpangan tiap-tiap nilai dari *mean* itu. Misalnya, jika seorang mempunyai IQ 110, sedangkan *mean* Iq dari kelompoknya = 100, maka deviasi IQ orang tersebut adalah 110 – 100 = +10. Jika orang lain dalam grup itu mempunyai IQ 85, maka deviasi orang itu adalah 85 – 100 = -15. Deviasi yang bertanda plus menunjukkan deviasi di atas *mean*, sedangkan yang bertanda minus menunjukkan deviasi di bawah *mean*. Akan tetapi dalam perhitungan mean deviasi tanda minus ditiadakan. Dalam statistika, deviasi diberi simbol dengan huruf-huruf kecil seperi x, y, d, dan sebagainya. Rumus adalah x = X – M atau y = Y – M, d = D – M, dan sebagainya.

Adapun rumus dari *Mean* deviasi adalah sebagai berikut (Samsudi, 2008) :

$$MD = \frac{\sum |x|}{N}$$

Dimana : MD = *Mean* Deviasi

$\sum|x|$ = Jumlah deviasi dalam harga mutlaknya

N = Jumlah individu/kasus