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.

2.2 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, ULP/PP 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 :
   - Dokumen Perjalanan ke Luar Negeri
   - Tiket Penerbangan
   - Kelebihan Bagasi (*Excess Baggage*)
   - Bagasi yang Tidak Dibawa Sendiri (*Unaccompanied Baggage*)
   - Biaya Perjalanan Darat (*Local / Inland Travel*)

- Biaya Pembelian Kebutuhan Proyek
- 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 :

- Sewa Kendaraan dan O&M
- Sewa Kantor Proyek
- Sewa Peralatan Kantor
- Sewa Furniture Kantor
- Biaya Operasional Kantor Proyek
- Biaya ATK (*Office Consumables*)
- Biaya Komputer & *Printer Consumables*
- Biaya Komunikasi
- Tunjangan Harian (*Per Diem Allowance*)
- Tunjangan Perumahan (*Housing Allowance*)
- Penempatan Sementara (*Temporary Lodging*)
- Tunjangan Penempatan (*Relocation Allowance*)
- Tunjangan Tugas Luar (*Out of Station Allowance / OSA*)
- Penginapan Tugas Luar
- Cuti Tahunan (*Annual Leave*)
- 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 :

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

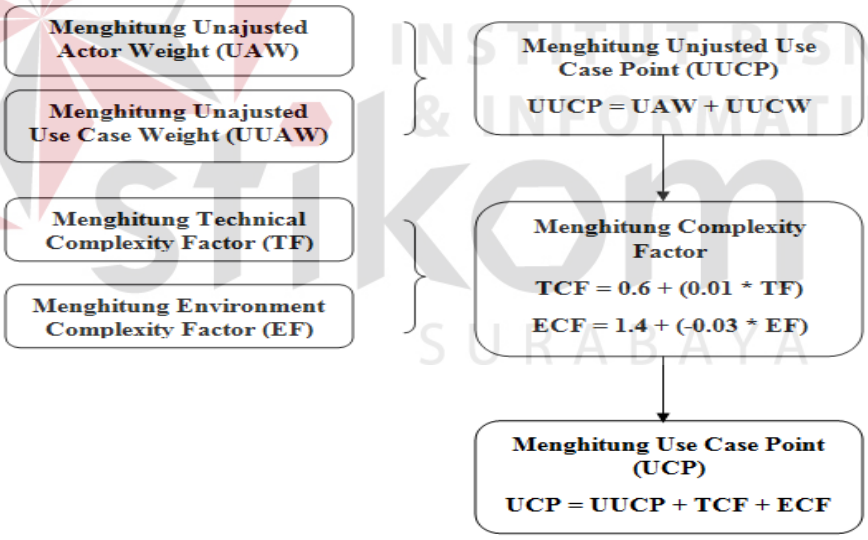
2.2.3 Use Case Point (UCP)

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:

1. UCP memiliki deviasi sebesar 6% (Nageswaran, 2001),
2. UCP memiliki deviasi sebesar 19%, sementara estimasi para ahli memiliki deviasi sebesar 20% (Anda, 2002),
3. 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. 1 Langkah-langkah Metode *Use Case Point* (UCP)

2.2.3.1.1 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

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.

2.2.3.1.2 *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 :

$$UUCP = UAW + UUCW$$

2.2.3.2 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 :

1. *Technical Complexity Factor* (TCF)
2. *Environmental Complexity Factor* (ECF)

Berikut penjelasan masing-masing dari *complexity factor* :

2.2.3.2.1 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
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).

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

2.2.3.2.2 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.

$$UCP = UUCP * TCF * ECF$$

2.2.4 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 x 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

2.2.5 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).

2.2.6 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

2.2.7 Agile Development

Kata *Agile* berarti bersifat cepat, ringan, bebas bergerak, waspada. Kata ini digunakan sebagai kata yang mengambarkan konsep model proses yang berbeda dari konsep model-model proses yang sudah ada (Beck, 2000). Konsep *Agile Software Development* dicetuskan oleh Kent Beck dan 16 rekannya.

Dalam *Agile Software Development* interaksi dan personel lebih penting dari pada proses dan alat, software yang berfungsi lebih penting daripada dokumentasi yang lengkap, kolaborasi dengan klien lebih penting dari pada negosiasi kontrak, dan sikap tanggap terhadap perubahan lebih penting daripada mengikuti rencana. Namun demikian, sama seperti model proses yang lain, *Agile Software Development* memiliki kelebihan dan tidak cocok untuk semua jenis proyek, produk, orang dan situasi. *Agile Software Development* memungkinkan

model proses yang toleransi terhadap perubahan kebutuhan sehingga perubahan dapat cepat ditanggapi.

Agile Software Development juga melihat pentingnya komunikasi antara anggota tim, antara orang-orang teknis dan *businessman*, antara developer dan managernya. Ciri lain adalah klien menjadi bagian dari tim pembangun *software*. Ciri-ciri ini didukung oleh 12 prinsip yang ditetapkan oleh Agile Alliance. Menurut Agile Alliance, 12 prinsip ini adalah bagi mereka yang ingin berhasil dalam penerapan Agile Software Development (Ambler, 2001) :

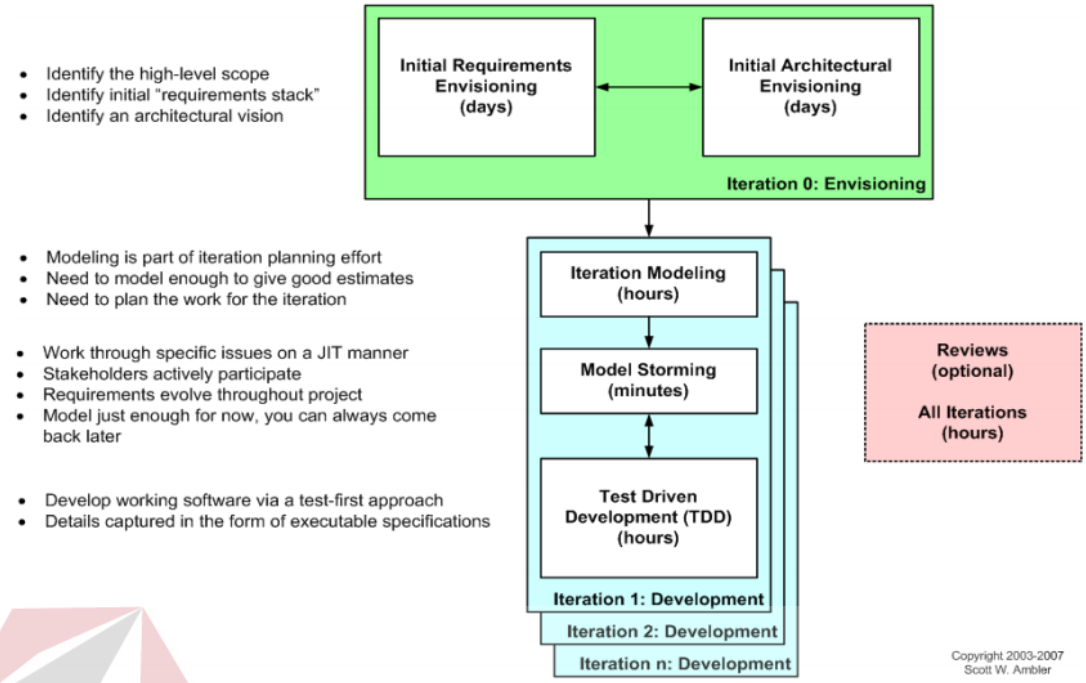
1. Prioritas tertinggiadalah memuaskan pelanggan melaluipenyerahan awal dan berkelanjutan perangkat lunak yang bernilai,
2. Menerima perubahan *requirements* meskipun perubahan tersebut diminta pada akhir pengembangan,
3. Memberikan perangkat lunak yang sedang dikerjakan dengan sering, beberapa minggu atau beberapa bulan, dengan pilihan waktu yang paling singkat,
4. Pihak bisnis dan pengembang harus bekerja sama setiap hari selama pengembangan berjalan,
5. Bangun proyek bersama individu-individu yang bermotivasi tinggi dengan memberikan lingkungan dan dukungan yang diperlukan, dan mempercayai mereka sepenuhnya untuk menyelesaikan pekerjaannya,
6. Metode yang paling efektif dan efisien dalam menyampaikan informasi kepada tim pengembangan adalah dengan komunikasi langsung *face-to-face*,
7. Perangkat lunak yang dikerjakan merupakan pengukur utama kemajuan,

8. Proses *agile* memberikan proses pengembangan yang bisa ditopang. Sponsor, pengembang, dan user harus bisa menjaga ke-konstanan langkah yang tidak pasti,
9. Perhatian yang terus menerus terhadap rancangan dan teknik yang baik meningkatkan *agility*,
10. Kesederhanaan seni untuk meminimalkan jumlah pekerjaan adalah penting,
11. Arsitektur, *requirements*, dan rancangan terbaik muncul dari tim yang mengatur sendiri,
12. Pada interval reguler tertentu, tim merefleksikan bagaimana menjadi lebih efektif, kemudian menyesuaikannya.

Model-model proses yang termasuk *agile process model* :

1. *Extreme Programming* (XP),
2. *Adaptive Software Development* (ASD),
3. *Dynamic Systems Development Method*,
4. *Scrum*,
5. *Agile Modeling*.

Berikut aktivitas dalam model agile dapat dilihat pada gambar 2.2 :



Gambar 2. 2 Aktivitas dalam *Agile Development*

2.2.8 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