



BAB II

TINJAUAN PUSTAKA

Bab II ini menjelaskan teori yang berkaitan dengan judul tugas akhir, perancangan program, Perangkat Pembuatan Pemrograman tugas akhir, dan metodologi pengembangan sistem.

2.1 Pengertian Sistem

Menurut Iswandy (2015 : 72), Sebuah sistem terdiri dari berbagai unsur yang saling melengkapi dalam mencapai tujuan dan sasaran. Unsur-unsur yang terdapat dalam sistem itulah yang disebut dengan subsistem.

Menurut Satzinger, Jackson, dan Burd (2010 : 6), sistem merupakan sekumpulan komponen yang saling berhubungan dan bekerja bersama untuk mencapai suatu tujuan.

Berdasarkan kedua pengertian di atas, maka dapat disimpulkan bahwa pengertian sebuah sistem adalah sekumpulan komponen yang terintegasi dan bekerja bersama guna mencapai suatu tujuan tertentu.

2.2 Pengertian Repository

Menurut Lynch (2003) dalam Hasugian (2012 : 5-6), mendefinisikan repositori pada perguruan tinggi adalah serangkaian pelayanan yang diberikan oleh perguruan tinggi kepada anggota komunitasnya untuk mengelola dan menyebarkan bahan-bahan digital yang dihasilkan oleh institusi tersebut. Bahan-bahan digital yang dimaksud adalah seluruh karya ilmiah dan/atau output intelektual yang dihasilkan oleh suatu perguruan tinggi. Ada juga yang mendefinisikan repositori internal adalah tempat menyimpan seluruh karya yang dihasilkan oleh sivitas akademika suatu perguruan tinggi dan/atau karya lain mengenai perguruan tinggi yang bersangkutan. Akses terhadap karya tersebut sangat tergantung kepada kebijakan masing-masing perguruan tinggi.

Menurut Jones.et.al (2006) dalam Hasugian (2012 : 5-6), menganggap bahwa repositori adalah unsur-unsur konstituen dari perpustakaan digital, atau



yang melengkapi perpustakaan digital dengan menyeleksi koleksi-koleksi tertentu apakah berdasarkan lingkup institusi ataupun menurut disiplin ilmu tertentu (disipliner) untuk disediakan sebagaimana halnya sebuah perpustakaan. Penekanan secara institusi ataupun disipliner adalah bergantung kepada ruang lingkup dari sebuah repositori atau perpustakaan digital, apakah pengelolaan dan pelayanannya berdasar kepada bahan yang dihasilkan oleh satu institusi atau hanya mengumpulkan bahan-bahan yang berhubungan dengan suatu disiplin ilmu yang spesifik.

Berdasarkan kedua pengertian di atas, maka dapat disimpulkan bahwa pengertian *repository* adalah pelayanan yang diberikan oleh perguruan tinggi kepada anggota komunitasnya untuk mengelola dan menyebarkan karya ilmiah dan/atau output intelektual yang dihasilkan perguruan tinggi menurut disiplin ilmu tertentu (disipliner).

2.2.1 Alasan Membangun *Repository*

Menurut Pfister (2008) dalam Hasugian (2012 : 5-6) Terdapat berbagai alasan untuk membangun repository, mengemukakan sedikitnya ada tiga alasan membangun repositori,

Pertama adalah peningkatan visibilitas dan dampak dari *output* penelitian. Para peneliti dan lembaga mendapatkan manfaat dari repositori dalam cara yang sama yaitu mengetahui kejelasan dan dampak dari hasil penelitian. Membangun dan mempertahankan reputasi dalam komunitas ilmiah sangat penting bagi kegiatan akademik dan insitusi dan hal itu dapat dicapai dengan repositori. Untuk mengukur dampak penelitian misalnya, metode bibliometrik seperti analisis sitiran terhadap jurnal akademik yang dikelola oleh suatu institusi sering digunakan untuk mengukur atau mengetahui tingkat penggunaan jurnal tersebut. Sehingga melalui repositori akan lebih mudah diukur seberapa sering sebuah jurnal digunakan, seberapa sering sebuah artikel dalam jurnal ilmiah dibaca atau di-*download*, seberapa sering suatu laporan penelitian dibaca atau di-*download* dan sebagainya.



Kedua, yaitu berkaitan dengan perubahan dalam paradigma publikasi ilmiah. Munculnya gerakan untuk menyediakan akses gratis terhadap publikasi ilmiah. *Content* ilmiah dihasilkan dan dipublikasikan sendiri dan penyediaan akses gratis terhadap bahan-bahan tersebut adalah merupakan aktivitas utama dalam gerakan akses terbuka (*open access movement*).

Alasan ketiga membangun repositori adalah didasarkan atas kemungkinan perbaikan komunikasi internal. Dengan menyediakan penyimpanan bahan-bahan digital secara terpusat akan mendapatkan manfaat dari bahan yang telah dipublikasikan pada satu sisi, dan pada sisi yang lain menjadi dasar untuk mengetahui bahan-bahan yang belum dipublikasikan secara digital.

2.2.2 Keuntungan *Repository*

Menurut Hasugian, (2012 : 5-6), membangun repositori akan menghasilkan keuntungan baik bagi individu maupun bagi lembaga. Hasil-hasil penelitian, artikel ilmiah, makalah, tesis, disertasi dan karya ilmiah lainnya yang tersedia secara *online* dapat diakses, *download*, dan/atau disitir lebih cepat dan lebih sering dibanding dengan yang tersedia dalam format tercetak. Sehingga, menaruh karya akademis (karya ilmiah) pada sebuah repositori dengan akses terbuka, maka akan meningkatkan profil seorang penulis di bidangnya pada tingkat yang lebih luas (internasional), termasuk penyebaran dan dampak dari hasil penelitiannya. Repositori pada perguruan tinggi menjadi sarana penting untuk mempublikasikan penelitian dan karya-karya akademik yang dimilikinya. Reputasi perguruan tinggi akan semakin dikenal melalui peran repositori. Karya akademik perguruan tinggi tidak hanya tersebar melalui jurnal, akan tetapi dapat juga melalui repositori, sehingga akan meningkatkan visibilitas dan prestise.

2.3 Pengertian Tugas Akhir

Menurut Soedjono (1992 : 12), tugas akhir merupakan suatu karya ilmiah berdasarkan suatu kegiatan penelitian mandiri mahasiswa, disusun dalam jangka waktu satu semester dibawah bimbingan seorang dosen pembimbing dan dapat dibantu seorang pembantu pembimbing. Tugas akhir dilaksanakan secara mandiri



oleh mahasiswa, dimaksudkan bahwa inisiatif perancangan penelitian, pelaksanaan penelitian dan penulisan laporan tugas akhir ada pada diri mahasiswa sendiri.

Tugas Akhir merupakan suatu karya ilmiah yang disusun berdasarkan suatu kegiatan pengamatan/observasi mandiri mahasiswa. Yang dimaksud dengan mandiri, yaitu bahwa perencanaan, pelaksanaan, dan penulisan laporan hasil pengamatan/observasi dilakukan oleh mahasiswa secara individual, meskipun masih diperlukan bimbingan dari dosen.

Berdasarkan kedua pengertian di atas, maka dapat disimpulkan bahwa pengertian tugas akhir adalah suatu karya ilmiah yang disusun oleh mahasiswa sendiri, yang dibantu oleh dosen pembimbing berdasarkan suatu kegiatan penelitian/ observasi.

2.4 Algoritma *Brute Force*

Menurut samo dalam Pratiwi (2016 : 120), algoritma *brute force* adalah algoritma yang digunakan untuk mencocokkan *pattern* dengan semua teks antara 0 dan n-m untuk menemukan keberadaan *Pattern* teks. Algoritma *brute force* memecahkan masalah dengan sangat sederhana, langsung, dan jelas. Algoritma *brute-force* merupakan suatu teknik yang biasa digunakan bila si penyusun algoritma lebih mempertimbangkan memperoleh solusi dari problem secara langsung apa adanya.

Menurut Munir, dalam Saragih (2013 : 85), algoritma *Brute Force* adalah sebuah pendekatan langsung (*straight forward*) untuk memecahkan suatu masalah, yang biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan. Pada dasarnya algoritma *Brute Force* adalah alur penyelesaian suatu permasalahan dengan cara berpikir yang sederhana dan tidak membutuhkan suatu pemikiran yang lama.

Di dalam pencocokkan string, terdapat istilah teks dan pattern. Teks merupakan kata yang dicari dan dicocokkan dengan pattern. Sedangkan pattern merupakan kata yang diinputkan untuk dicocokkan.



2.4.1 Kelebihan Algoritma *Brute force*

Menurut Mesran (2014 : 102), berikut ini beberapa kelebihan yang dimiliki oleh *brute force*, yaitu:

1. Algoritma brute force dapat digunakan untuk memecahkan hampir sebagian besar masalah.
2. Sederhana dan mudah dimengerti.
3. Menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, perkalian matriks.
4. Menghasilkan algoritma baku (standar) untuk tugas-tugas komputasi seperti penjumlahan/perkalian N buah bilangan, menentukan elemen minimum atau maksimum ditabel.

2.4.2 Kekurangan Algoritma *Brute Force*

Menurut Mesran (2014 : 102), berikut ini beberapa kelemahan yang dimiliki oleh *brute force*, yaitu:

1. Jarang menghasilkan algoritma yang mangkus/efektif.
2. Lambat sehingga tidak dapat diterima.
3. Tidak kreatif teknik pemecahan masalah lainnya.

2.5 Pengertian *Website*

Menurut Puspitosari (2010), *Webiste* merupakan halaman yang memberikan informasi yang disediakan melalui jaringan internet sehingga bisa diakses diseluruh dunia, selama terkoneksi dengan jaringan internet.

Menurut Lukmanul (2004), *Website* merupakan fasilitas internet yang menghubungkan dokumen dalam lingkup lokal maupun jarak jauh. Dokumen pada website disebut dengan web page dan link dalam website memungkinkan pengguna bisa berpindah dari satu page ke page lain (*hyper text*), baik diantara page yang disimpan dalam server yang sama maupun server diseluruh dunia. Pages diakses dan dibaca melalui browser seperti Netscape Navigator, Internet Explorer, Mozilla Firefox, Google Chrome dan aplikasi browser lainnya.



Bedasarkan kedua pengertian di atas, maka dapat disimpulkan bahwa pengertian website merupakan fasilitas yang memberikan informasi yang disediakan jaringan internet, yang bisa diakses dalam server yang sama maupun server diseluruh dunia, melalui *browser Mozilla Firefox, Google Chrome* dan aplikasi *browser* lainnya.

2.6 Unified Modeling Language (UML)

Menurut Sukamto dan Shalahuddin dalam Aprianti (2016 : 22) *unified Modeling Language (UML)* adalah standarisasi bahasa pemodelan untuk membangun perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek

Diagram-diagram yang digunakan pada UML antara lain adalah *class diagram, object diagram, use case diagram, activity diagram, dan sequence diagram*.

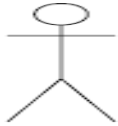
2.6.1 Use Case Diagram

Menurut Sukamto dan Shalahuddin dalam Aprianti (2016 : 23), *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin agar dapat dipahami. Ada dua hal utama pada *use case* yaitu mendefinisikan apa yang disebut aktor dan *usecase*.

Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat sistem itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

Use case merupakan fungsionalitas yang disediakan sistem sebagai berikut unit unit yang saling bertukar pesan antar unit atau aktor.

**Tabel 2.1** Simbol-simbol dalam *Use case*

No	Simbol	Nama	Keterangan
1.		Aktor/ <i>Actor</i>	Aktor adalah pengguna sistem. aktor tidak terbatas hanya manusia saja, jika sebuah sistem berkomunikasi dengan aplikasi lain dan membutuhkan input atau memberikan output, maka aplikasi tersebut juga bisa dianggap sebagai aktor.
2.		<i>Use case</i>	<i>Use case</i> digambarkan sebagai lingkaran elips dengan nama <i>use case</i> dituliskan didalam elips tersebut, digunakan sebagai unit-unit yang saling bertukar pesan antar unit / aktor.
3.		<i>Association</i> / Asosiasi	Asosiasi digunakan untuk menghubungkan <i>actor</i> dengan <i>use case</i> . Asosiasi digambarkan dengan sebuah garis yang menghubungkan antara <i>Actor</i> dengan <i>Use Case</i> .
4.		<i>Generalization</i> / Generalisasi	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
5.		<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> .
6.		<i>Extend</i> / Ekstensi	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan

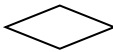
Sumber: Sukamto dan Shalahuddin dalam Aprianti (2016 : 23)



2.6.2 Activity Diagram

Menurut Sukamto dan Shalahuddin dalam Aprianti (2016 : 23) *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Diagram aktivitas menggambarkan aktivitas –aktivitas sistem bukan apa yang dilakukan aktor.

Tabel 2.2 Simbol-simbol *Activity Diagram*

No	Simbol	Nama	Keterangan
1.		<i>Start state</i>	Titik awal atau permulaan
2.		<i>End state</i>	Titik akhir atau akhir dari aktivitas
3.		<i>Activity</i>	<i>Activity</i> atau aktivitas yang dilakukan oleh aktor
4.		<i>Decision</i>	Pilihan untuk mengambil keputusan
5.		<i>Join</i>	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.

Sumber: Sukamto dan Shalahuddin dalam Aprianti (2016 : 23)

2.6.3 Sequence Diagram

Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirim dan diterima antar objek. Oleh karena itu untuk menggambar diagram sequence maka harus diketahui objek-objek yang terlihat pada sebuah *use case* beserta metode- metode yang dimiliki kelas yang diinstansi menjadi objek itu.

**Table 2.3** Simbol-simbol dalam *Sequence Diagram*

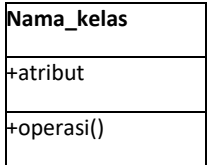
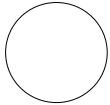
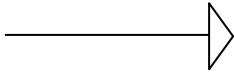
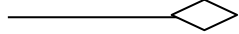
SIMBOL	NAMA	KETERANGAN
	<i>Actor</i>	Orang ataupun pihak yang akan mengelola system
	<i>Lifeline</i>	Menggambarkan sebuah objek dalam sebuah sistem atau salah satu komponennya
	<i>Create Message</i>	Pembuatan sebuah <i>message</i> sederhana antar elemen dan juga mengindikasikan komunikasi antara objek
	<i>Synchronous Message</i>	<i>Message</i> ini mengaktifkan sebuah proses dan sampai selesai, baru bisa mengirimkan sebuah <i>message</i> baru
	<i>Message to self</i>	Suatu hasil kembalian sebuah operasi dan berjalan pada objek itu sendiri

Sumber: Sukamto dan Shalahuddin dalam Aprianti (2016 : 23)

2.6.4 Class Diagram

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan di buat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi

**Tabel 2.4** Simbol-simbol dalam *Class Diagram*

SIMBOL	NAMA	KETERANGAN
	<i>Class/</i> Kelas	Kelas pada struktur system
	Antarmuka <i>/interface</i>	Sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek.
	Asosiasi/ <i>Association</i>	Relasi antarkelas dengan makna umum, asosiasi biasanya disertai dengan <i>multiplicity</i> .
	Asosiasi berarah/ <i>directed association</i>	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i> .
	Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus).
	Kebergantungan/ <i>Dependency</i>	Relasi antar kelas dengan makna kebergantungan antar kelas.
	Agregasi/ <i>aggregation</i>	Relasi antar kelas dengan makna semua-bagian.

Sumber: Sukamto dan Shalahuddin dalam Aprianti (2016 : 22)

2.7 Pengertian MySQL

Menurut Nugroho (2014 : 133) dalam Kurniawan MySQL merupakan *database* yang paling digemari di kalangan programmer web, dengan alasan bahwa program ini merupakan database yang sangat kuat dan cukup stabil untuk digunakan sebagai media penyimpanan data. Sebagai sebuah *database* server yang



mampu untuk memanajemen *database* dengan baik, *MySQL* terhitung merupakan *database* yang paling digemari dan paling banyak digunakan dibanding *database* lainnya.

Menurut Winarno dan Zaki (2014 : 102) dalam Kurniawan *MySQL* adalah sebuah *software database*. *MySQL* merupakan tipe data relasional yang artinya *MySQL* menyimpan datanya dalam bentuk tabel- tabel yang saling berhubungan.

Berdasarkan kedua pengertian di atas, maka dapat disimpulkan bahwa pengertian *MySQL* adalah tipe data relasional dalam bentuk tabel yang saling berhubungan dan cukup stabil sebagai sebuah *database server*.

2.8 Perangkat Pembuatan Pemrograman

2.8.1 Pengertian PHP

Menurut Winarno dan Zaki (2014 : 49) dalam Kurniawan PHP adalah sebuah bahasa pemrograman web berbasis *server (server-side)* yang mampu memarsing kode PHP dari kode web dengan ekstensi *.php*, sehingga menghasilkan tampilan website yang dinamis di sisi *client(browser)*. PHP adalah bahasa *script* yang sangat cocok untuk pengembangan web dan dapat dimasukkan.

Menurut Binarso (2012 : 76) PHP (*Hypertext Preprocessor*) merupakan bahasa *scripting* yang tergabung menjadi satu dengan HTML dan dijalankan pada *server side* atau semua perintah yang diberikan akan secara penuh dijalankan pada *server*, sedangkan yang dikirimkan ke klien (*browser*) hanya berupa hasilnya saja ke dalam HTML.

Berdasarkan kedua pengertian di atas, maka dapat disimpulkan bahwa pengertian PHP merupakan bahasa *scripting* dengan ekstensi *.php* sehingga dapat dijalankan pada *server side* menghasilkan tampilan website yang dinamis di sisi *client(browser)*.



2.8.2 Pengertian XAMPP

Menurut Aditya dalam Priyanti (2013 : 56) XAMPP adalah perangkat lunak bebas, yang mendukung banyak system operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (localhost), yang terdiri atas program Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMP merupakan singkatan dari X (empat system operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU *General Public License* dan bebas, merupakan web server yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis.

Menurut Binarso (2012 : 76) XAMPP adalah sebuah *software web server Apache* yang didalamnya sudah tersedia *database server MySQL* dan mendukung *PHP programming*. XAMPP merupakan singkatan dari X (untuk empat sistem operasi), *Apache*, *MySQL*, *PHP*.

Berdasarkan kedua pengertian di atas, maka dapat disimpulkan bahwa XAMPP adalah singkatan dari X (empat system operasi apapun), *Apache*, *MySQL* dan bahasa pemrograman *PHP* dan *Perl*. Fungsinya sebagai server yang berdiri sendiri (localhost).

2.9 Metode Pengembangan Sistem

Menurut Fatoni dan Dwi (2016 : 17) Metode *Extreme Programming* sering juga dikenal dengan metode *XP*. Metode ini dicetuskan oleh *Kent Beck*, seorang pakar *software engineering*. *Extreme programming* adalah model pengembangan perangkat lunak yang menyederhanakan berbagai tahapan pengembangan sistem menjadi lebih efisien, adaptif dan fleksibel.

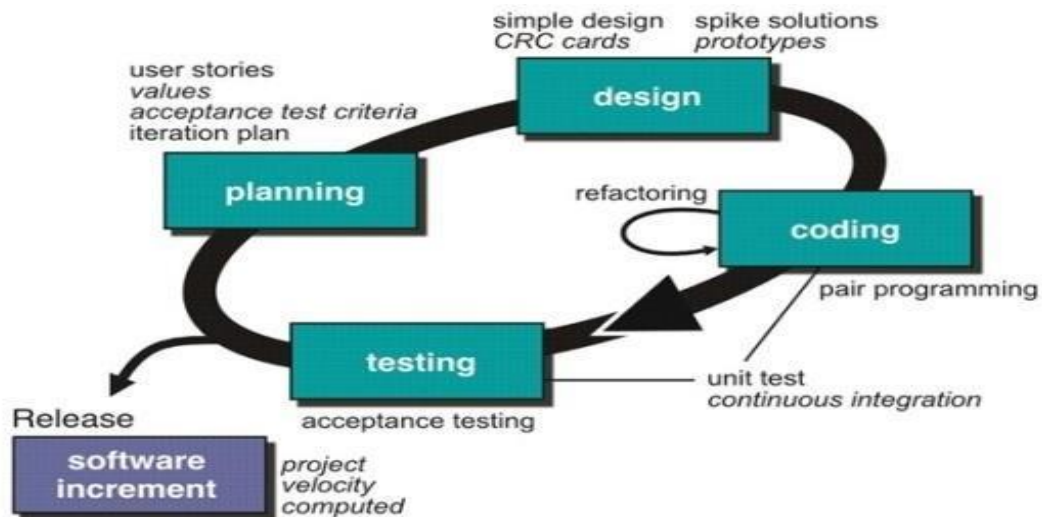
Nilai dasar metode *Extreme programming* :

1. *Communication* : Memfokuskan komunikasi yang baik antara *programmer* dengan *user* maupun antar *programmer*.
2. *Courage* : Pengembang perangkat lunak harus selalu memiliki keyakinan, keberanian dan integritas dalam melakukan tugasnya.
3. *Simplicity* : Lakukan semua dengan sederhana.



4. *Feedback* : Mengandalkan *feedback* sehingga dibutuhkan anggota tim yang berkualitas.

5. *Quality Work* : Proses berkualitas berimplikasi pada perangkat lunak yang berkualitas sebagai hasil akhirnya.



Gambar 2.1 Tahapan *Extreme Programming*

1. *Planning*, Aktivitas *planning* dimulai dengan membentuk *user stories*. Anggota *Extreme Programming* (XP) team kemudian menilai setiap *story* dan menentukan *cost* diukur dalam *development week*. Customer dan *Extreme Programming* (XP) team bekerja bersama untuk memutuskan bagaimana *group story* untuk *release* berikutnya (*software increment*) berikutnya untuk dibangun oleh *Extreme Programming* (XP) team. Jika komitmen telah dibuat, *Extreme Programming* (XP) team akan membangun *story-story* dengan cara:

1. Semua *story* segera diimplementasikan (dalam beberapa minggu).
Story dengan *value* tertinggi akan dipindahkan dari jadwal dan diimplementasikan pertama.
2. *Story* dengan resiko paling tinggi akan diimplemetasikan lebih dulu.
Setelah *project* pertama *di-release* dan *didelivery*, *Extreme Programming* (XP) team memperhitungkan kecepatan *project*. Selama *development*,



customer dapat menambah *story*, merubah *value*, membagi *story* atau menghapusnya.

2. *Design. Extreme Programming* menggunakan CRC card, untuk mengenali dan mengatur *object oriented class* yang sesuai dengan software increment.
3. *Coding*. Sebelum membuat *code*, lebih baik membuat unit test tiap *story* untuk dimasukkan dalam *software increment*. *Extreme Programming (XP)* menyarankan agar dua orang bekerja bersama pada satu komputer *workstation* untuk membuat *code* dari satu *story (pair programming)*, untuk menyediakan *real time problem solving* dan jaminan *real time quality*. Setelah *pair programming* selesai, *code* diintegrasikan dengan kerja lainnya (*continuou sintegration*).
4. *Testing*. Unit test yang telah dibuat harus diimplementasikan menggunakan suatu *framework* dan diatur ke dalam *universal testing suite*, integrasi dan validasi sistem dapat dilakukan setiap hari. *Customer test (acceptance test)* dilakukan oleh *customer* dan fokus pada keseluruhan fitur dan fungsional sistem. *Acceptance test* diperoleh dari *customer stories* yang telah diimplemetasikan sebagai bagian dari *software release*.