BAB II
TINJAUAN PUSTAKA

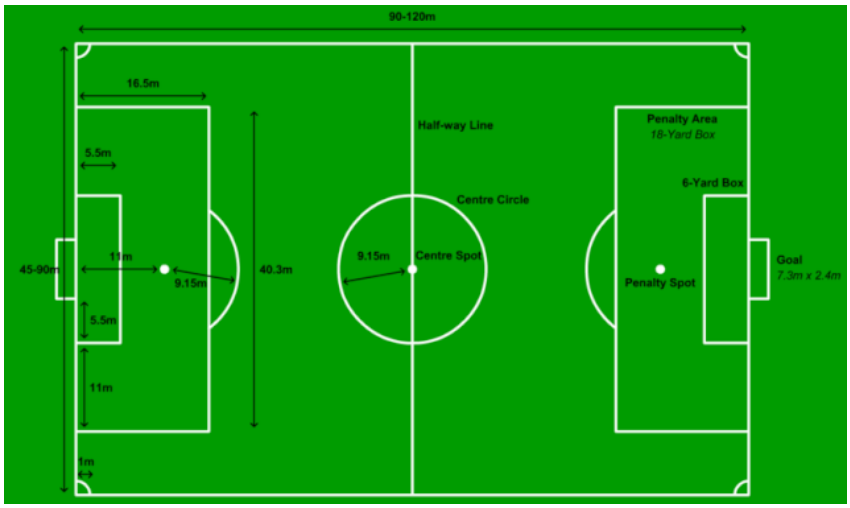
2.1. Permainan dan Permainan Simulasi

Menurut kamus besar Bahasa Indonesia, simulasi dapat diartikan sebagai suatu tindakan yang seolah-olah atau pura-pura. Artinya, simulasi adalah tindakan yang seolah-olah menyerupai kondisi yang sesungguhnya. Berdasarkan pengertian di atas, dapat dikatakan bahwa unsur-unsur pada model permainan simulasi adalah sistem sosial atau fisik (*physical or social system*), abstraksi (*abstracted*), realitas (*reality*), dan penyederhanaan (*simplified*) dan alasan studi (*study purposes*).

Rahadian (2012) menjabarkan bahwa dalam membuat suatu permainan terdapat beberapa aspek yang harus diperhatikan. Pertama, efisiensi *script* yang artinya tidak mengulangi *script* yang sama untuk kegiatan yang sama. Kedua, simbol atau objek harus menggunakan simbol yang sama sehingga tidak membuat kebingungan dalam memahami objek. Ketiga, jika objek dirasa kurang memberikan penekanan terhadap permainan, maka gunakanlah *tools* yang dapat mempertajam objek. Lia (2012) menuturkan bahwa permainan dalam komputer pada dasarnya merupakan suatu hiburan. Sebagian besar permainan tidak mengarahkan kepada hal yang bersifat edukatif. Oleh karena itu seleksi terhadap berbagai permainan harus dilakukan agar suasana edukatif dapat terjadi melalui suatu permainan komputer.

2.2. Sepak Bola

Sepak bola merupakan olah raga permainan yang terdiri atas 22 orang pemain dari dua tim. Tim yang mampu mencetak lebih banyak *goal* adalah tim juara. Menurut Knet (dalam Regent, 2003), lapangan sepak bola profesional berukururan 100-110 meter dengan lebar 64-75 meter.



Gambar 2.1. Lapangan sepak bola

Terdapat beberapa istilah kondisi bola dalam permainan sepak bola, diantaranya adalah :

1. Bola hidup

   Bola hidup terjadi selama permainan berlangsung, kecuali bola keluar lapangan atau terjadi pelanggaran.

2. Bola mati

   Kondisi bola mati terjadi apabila terjadi pelanggaran, atau bola meninggalkan lapangan permainan.

Knet (dalam Regent, 2003), juga menjabarkan bahwa pada umumnya terdapat 4 posisi bermain yakni penjaga gawang, pemain bertahan, pemain tengah atau gelandang, dan pemain menyerang.

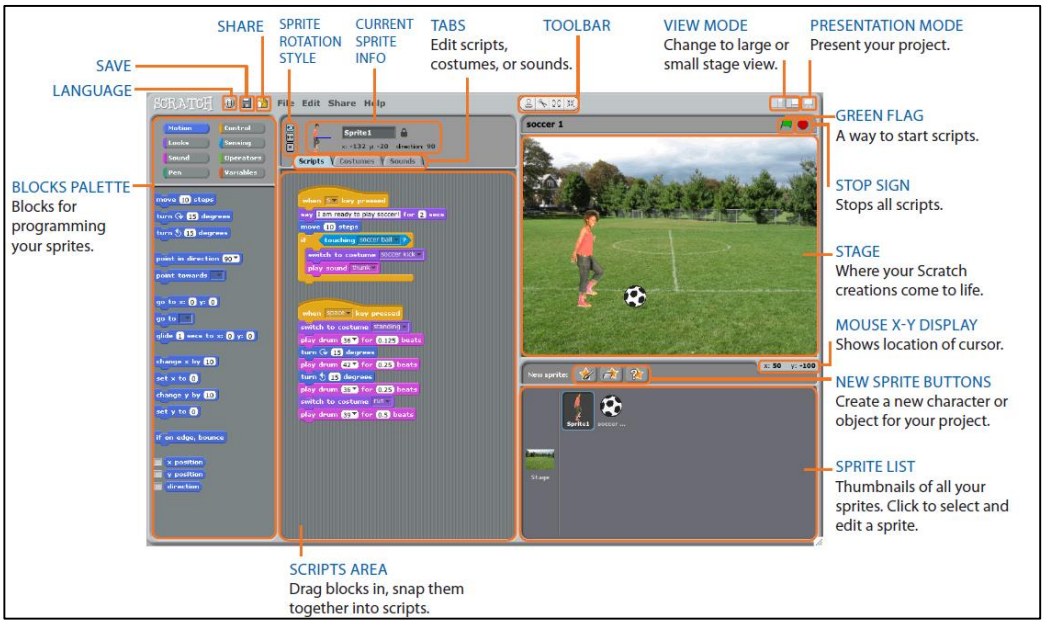
Dunning (2000) juga menjelaskan, dalam permainan sepak bola, benturan fisik baik yang disengaja maupun tidak disengaja dalam konten benturan, sah maupun tidak sah dapat terjadi kapan saja. Adanya kontak fisik secara langsung dapat terjadi antara dua pemain atau lebih dari masing-masing tim. Setiap pemain dalam permainan sepak bola memiliki respon yang berbeda dalam bermain, hal tersebut dikarenakan oleh porsi latihan yang berbeda dari setiap pemain (Mejri et al., 2005).

Menurut Lomax (1999), studi pada tahun 1970an membedakan 2 jenis lokasi permainan, yakni pemain tengah, dan pemain keliling. Rotasi pemain terjadi dalam internal permainan lokasi, yakni rotasi pemain tengah dan rotasi pemain keliling. Saat ini sepak bola telah menjadi olah raga permainan yang sangat diminati. Sehingga, sepak bola telah menjadi strategi global bagi pihak komersil pendukung kegiatan ini, karena respon terhadap barang pasar yang berkaitan dengan sepak bola cukup tinggi (Croci, 1999).

2.3. Pemrograman Scratch

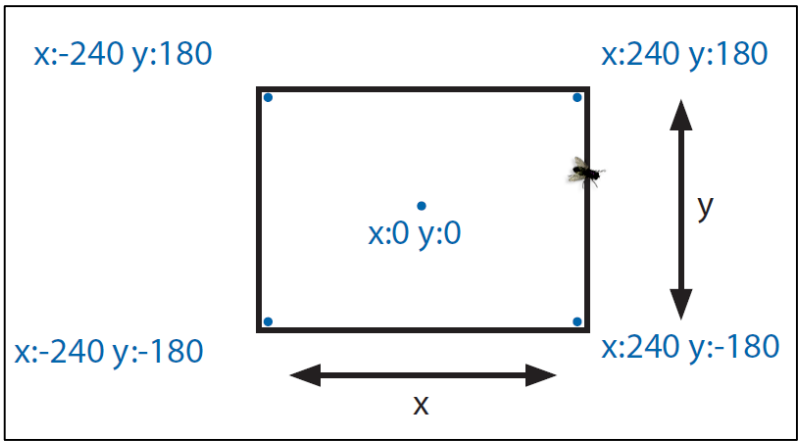
Scratch merupakan bahasa pemrograman yang dirancang untuk membuat kisah interaktif, *games*, dan animasi lainnya. Setiap objek disebut dengan *Sprite* dan dapat diberikan perlakuan yang disebut *Script*. Scratch dikembangkan oleh *Lifelong Kindergarten Group at the MIT Media Lab*, dengan dukungan dana oleh *National*

Science Foundation, *Microsoft*, *Intel Foundation*, *Nokia*, dan *MIT Media Lab research consortia* (Anonim, 2009).



Gambar 2.2 Scratch Interface
Source : Anonim (2009)

Area untuk menciptakan *project* disebut dengan *Stage* dengan lebar 480 satuan *stage* dan tinggi 360 satuan *stage*.



Gambar 2.3 Ukuran Stage
Source : Anonim (2009)

Scratch dapat berjalan pada sistem operasi Windows XP, Windows 2000, Windows Vista, Windows 7, Windows 8, dan Mac OS X 10.4. *Display* minimal layar komputer berukuran 800x480 dengan 16-bit *color* atau lebih. Dalam proses instalasinya, Scratch membutuhkan sekurang-kurangnya 120 *megabytes disk* kosong.

2.4. *Knowledge Base* (Basis Pengetahuan)

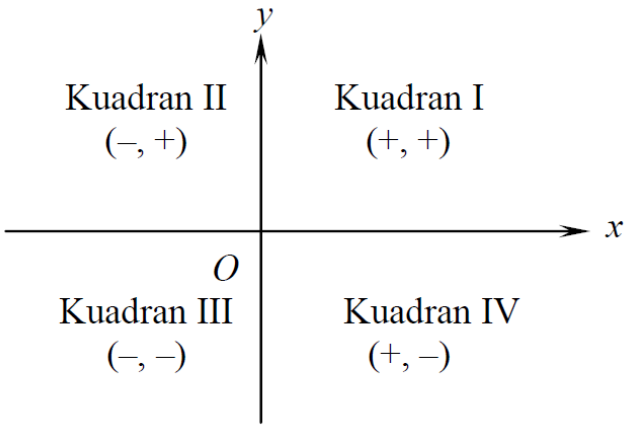
Knowledge Base merupakan suatu bagian dari kecerdasan buatan (*Artificial Intellegence*) yang diterapkan pada suatu sistem agar sistem dapat mengambil keputusan sendiri terhadap berbagai kondisi. (Elter, 2009)

Langkah utama dalam *Knowledge Base* adalah memisahkan antara pengetahuan program dengan pengetahuan yang sebenarnya, misalnya bagaimana untuk mengambil kesimpulan dari logika kondisi yang ada. Kamus bahasa menjadi dasar berpikir dari sistem cerdas sehingga sistem seolah-olah dapat berpikir baik seperti manusia, bahkan dapat berpikir lebih baik dari manusia. Pengetahuan yang disimpan dalam kamus bahasa adalah deskripsi terhadap suatu pengetahuan pada umumnya, dengan adanya penggabungan dari beberapa kamus bahasa maka terjalin suatu langkah inferensi untuk menetapkan kesimpulan.

2.5. Sistem Koordinat Kartesius

Koordinat Kartesius merupakan potongan dari dua garis bilangan riil yang saling tegak lurus di titik nolnya (Sembiring dkk, 2012). Sumbu X merupakan sumbu

mendatar dengan nilai positif menuju ke arah kanan, sedangkan sumbu Y merupakan sumbu tegak dengan nilai positif menuju ke atas. Skala pada sumbu-sumbu koordinat bernilai sama. Sumbu-sumbu koordinat membagi bidang datar menjadi empat daerah yang disebut dengan kuadran, disimbolkan dengan angka Romawi (Noormandiri, 2012).



Gembar 2.4. Kuadran pada koordinat kartesius

2.6. Teori Permainan

Teori permainan secara umum digunakan untuk menyelesaikan masalah yang berkaitan dengan tindakan sebuah perkara bisnis. Misalnya untuk memperoleh kemenangan dalam permainan usaha yang dilakukan. Dalam dunia nyata persaingan bisnis atau non bisnis sering terjadi. Dalam proses persiangan ini dibutuhkan strategi yang tepat agar dapat menang dalam sebuah persaingan. Strategi yang paling layak digunakan disebut dengan strategi optimal bagi pihak yang bersaing.

Menurut Ririez (2012), teori permainan (*game theory*) adalah suatu pendekatan matematis untuk merumuskan situasi persaingan dan konflik antara berbagai kepentingan. Teori ini dikembangkan untuk menganalisis proses pengambilan keputusan dari situasi-situasi persaingan yang berbeda-beda dan melibatkan dua atau lebih kepentingan.

Dalam sebuah *Pay off Matrix* atas strategi yang digunakan, pemain baris disebut sebagai pihak pertama, sementara pemain kolom merupakan pihak kedua. Setiap pemain (individual atau kelompok) diasumsikan memiliki kemampuan untuk mengambil keputusan secara bebas dan rasional.

		Perusahaan B		
		Strategi Harga Murah (S1)	Strategi Harga Sedang (S2)	Strategi Harga Mahal (S3)
Perusahaan A	Strategi Harga Murah (S1)	**1**	**9**	**2**
	Strategi Harga Mahal (S2)	**8**	**5**	**4**

Gambar 2.5. Pay off Matrix

Nilai positif merupakan keuntungan yang diperoleh bagi pemain baris yang artinya kerugian bagi pemain kolom. Sedangkan nilai negatif merupakan keuntungan bagi pemain kolom dan merupakan kerugian bagi pemain baris.

Suatu permainan/persaingan dikatakan adil apabila hasil akhir permainan atau persaingan menghasilkan nilai nol (0), atau tidak ada pemain atau perusahaan yang

menang/kalah atau mendapat keuntungan/kerugian. Sedangkan suatu strategi dikatakan dominan terhadap strategi lainnya apabila memiliki nilai *pay off* (pembayaran) yang lebih baik dari strategi lainnya (Nisan et al., 2007). Dalam permainan banyak orang (n-*person game*) kumpulan keputusan yang layak diambil berdasarkan dan berpengaruh kepada strategi yang dipilih oleh pemain lainnya (Topkis, 1979). Bagi pemain baris, nilai positif (keuntungan) yang diperoleh dari suatu strategi yang digunakan, menghasilkan nilai positif yang lebih besar dari hasil penggunaan strategi lainnya. Bagi pemain kolom, nilai negatif (kerugian) yang diperoleh dari suatu strategi yang digunakan, menghasilkan nilai negatif yang lebih kecil dari hasil penggunaan strategi lainnya.

2.7. Algoritma

Dalam buku yang ditulis oleh Sutanta (2004), istilah algoritma dapat dijabarkan sebagai kumpulan prosedur atau langkah yang harus dilaksanakan dalam menyelesaikan permasalahan yang orientasinya pada pemrograman komputer. Dengan kata lain, algoritma adalah penerjemahan proses yang dilakukan komputer menjadi bahasa yang dipahami oleh logika berpikir manusia.

Algoritma dapat dituliskan dalam berbagai bentuk, karena tidak adanya aturan baku mengenai penulisan algoritma. Namun terdapat beberapa cara penyajian algoritma, diantaranya :

1. *Pseudocode*

   Bentuk ini merupakan bentuk yang sederhana dan mudah dipahami, karena *pseudocode* merupakan deskripsi dari algoritma pemrograman komputer

yang menggunakan struktur dari beberapa bagian bahasa pemrograman tetapi ditujukan agar logika atau alur proses mudah dipahami. Artinya, *pseudocode* tampak seperti program namun tidak dapat di-*running*. Pada umumnya yang ditampilkan dalam *pseudocode* adalah variabel dan fungsi. Penggunaan *pseudocode* sering dijumpai dalam penyajian algoritma pada penulisan jurnal ilmiah. Contoh :

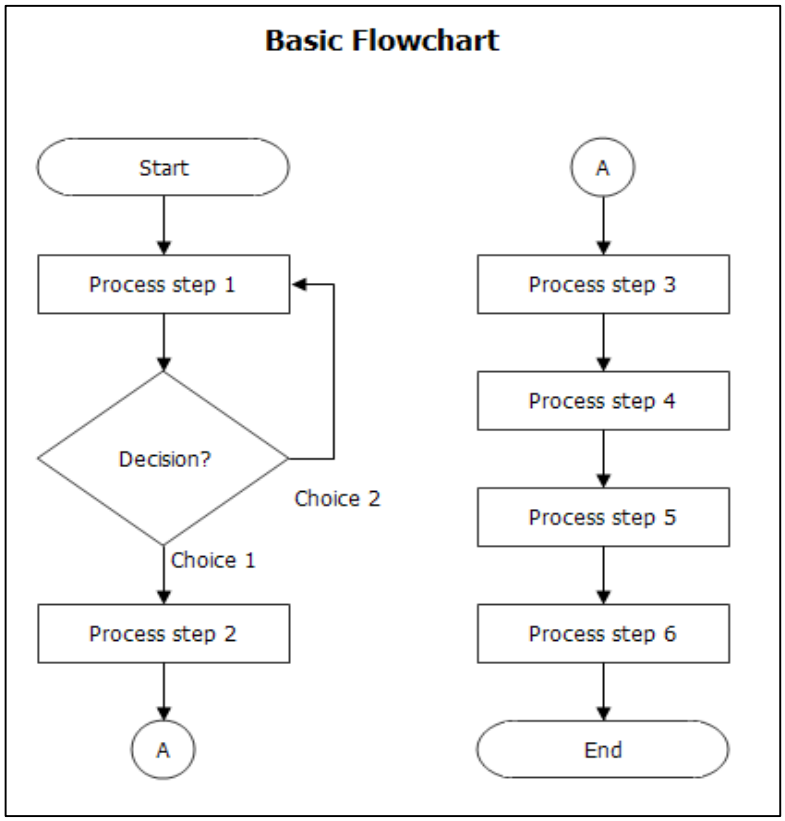
```
Mulai
 For i=1 to n-1
      For j=i+1 to n
            Tukar Xi dengan Xj jika Xi < Xj
            Next j
      End for
      Next i
 End for

 For k=1 to n
      Cetak Xk
Akhir.
```

Gambar 2.6. Penyajian Algoritma (Pseudocode)

2. Diagram Alir (*Flowchart*)

Bentuk algoritma bisa saja menjadi lebih mudah dipahami apabila ditungkan dalam bentuk diagram berarah yang menunjukkan alur proses. Diagram alir tidak hanya menunjukkan urutan proses, namun memberikan gambaran yang lebih efisien mengenai langkah berpikir jika algoritma yang dibuat memasuki tingkatan yang kompleks. Contoh :



Gambar 2.7. Diagram Alir (Flowchart)
Sumber : http://google.co.id/imgphp? hl=id&tab=wi

3. Detail Langkah Demi Langkah

Penyajian algoritma dalam bentuk ini menandakan bahwa algoritma telah benar-benar siap diimplementasikan dalam bahasa pemrograman. Artinya, detail langkah demi langkah menjelaskan keseluruhan struktur proses secara rinci. Contoh :

- *Step 1. Mulai*
- *Step 2. For a=1 to n*
- *Step 3. Definisikan X[a]*
- *Step 4. Teruskan a*
- *Step 5. For i=1 to n-1*
- *Step 6. For j=i+1 to n*
- *Step 7. Jika x[i] < x[j]*
- *Step 8. Isi variabel temp dengan x[i]*
- *Step 9. Isi variabel x[i] dengan x[j]*
- *Step 10. Isi variabel x[j] dengan temp*
- *Step 11. Teruskan j*
- *Step 12. Teruskan i*
- *Step 13. For k=1 to n*

- *Step 14. Cetak x[k]*
- *Step 15. Selesai*

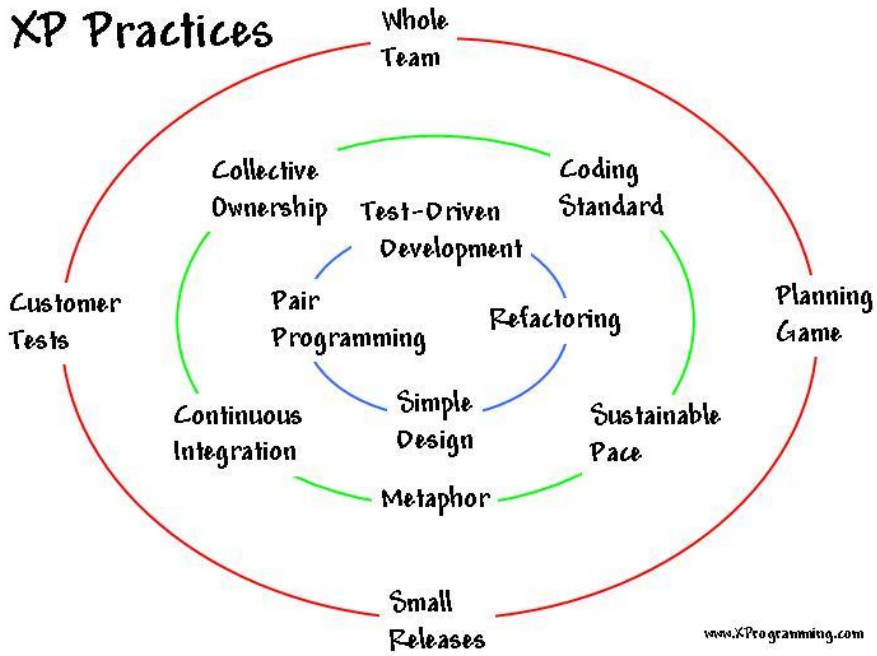
Secara umum, struktur proses yang terjadi dalam algoritma dapat dibedakan menjadi tiga, yakni proses berurutan (*sequence*), proses seleksi (*selection*), dan proses perulangan (*looping*).

2.8. *Extreme Programming*

Dalam pengembangan proyek perangkat lunak sering kali terjadi perubahan permintaan (*requirement*) dengan begitu cepat. Perubahan itu bisa saja terjadi karena aspek bisnis ataupun perubahan teknologi ketika pengembangan sistem sedang dilakukan. Untuk mengantisipasi hal tersebut, maka dibuatlah suatu pendekatan perangkat lunak untuk meningkatkan efisiensi dan efektifitas dari suatu proyek pengembangan perangkat lunak dengan mengombinasikan ide sederhana tanpa mengurangi kualitas *software*, pendekatan tersebut kita kenal dengan metode *Extreme programing* (XP) (Widhiarta, 2008).

XP melakukan pendekatan atau modeling perangkat lunak dengan menyederhanakan tahapan proses, sehingga sifatnya lebih adaptif dan fleksibel. Tujuan utama dari XP adalah menurunkan biaya, khususnya biaya perubahan *software*. *Requirement* dari *client* yang kurang spesifik dan intensitas pertemuan dengan *client* (dosen pembimbing) yang bisa sering dilakukan sangat mendukung *Extreme Programming* dipilih oleh penulis dalam mengembangkan sistem pada penelitian ini.

Tahapan dalam melaksanakan XP tampak pada Gambar 2.8.



Gambar 2.8. Tahapan dalam XP
Sumber : Whidhiarta, 2008

1. *Planning Game*

   Proses merencanakan yang cepat, mengutamakan aspek teknis dan pertemuan yang intensif antara pengembang dan klien menjadi pendekatan pertama untuk menentukan kebutuhan (*requirement*). Penggunaan redaksional "*game*" merupakan saran dari Kent Beck (penemu XP) untuk menentukan kebutuhan berdasarkan teknis *score card*. Semakin sulit aspek teknis yang dibutuhkan semakin tinggi pula skor pada kartu rencana tersebut (Widhiarta, 2008).

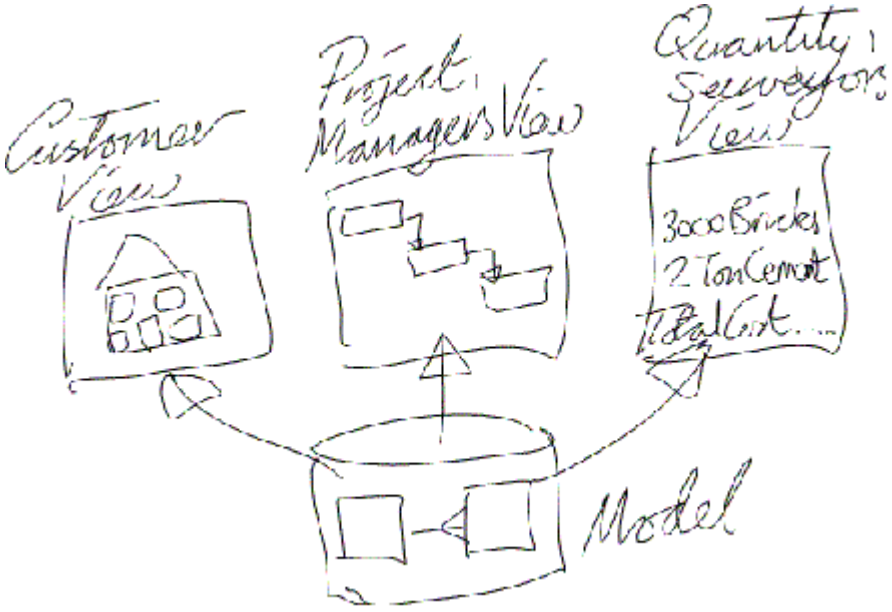
2. *Small Release*

   Sistem sederhana di *release* dalam lingkup yang cepat, sehingga dapat merilis versi terbaru dalam jangka waktu yang cepat. Hasil rilis dari bagian

perangkat lunak dapat segera dipresentasikan dan didiskusikan serta dilakukan tes awal.

3. *Metaphor*

Methapor menggambarkan visi secara keseluruhan serta tujuan dari perangkat lunak, sehingga pada dasarnya *methapor* sama dengan arsitektur pada perangkat lunak namun lebih naratif dan deskriptif. Dengan demikian diharapkan komunikasi antara klien dengan *developer* dapat berlangsung lebih baik dan lancar dengan penggunaan *metaphor*.



Gambar 2.9. Ilustrasi metaphor
Sumber : http://google.co.id/imgphp? hl=id&tab=wi

4. *Simple Design*

Penerapan desain yang *simple* tanpa mengurangi esensi dari fungsi sistem merupakan tahapan yang dilakukan untuk mengantisipasi perubahan cepat yang dapat terjadi. Artinya jika terjadi perubahan, maka perubahan tersebut dapat dengan mudah diatasi dan resiko kegagalan desain dapat diperkecil.

5. *Refactoring*

   Refactoring adalah proses pengubahan penulisan program untuk peningkatan kualitas hasil. Melalui proses ini, pengembang dapat melakukan berbagi usaha peningkatan kualitas tanpa mengulang proses sebelumnya (desain).

6. *Testing*

   Pengujian dilakukan pada setiap tahapan pengembangan tanpa harus menunggu keseluruhan sistem selesai dikerjakan. Hal ini bertujuan untuk meminimalisir kesalahan yang terjadi pada pengembangan perangkat lunak.

7. *Pair Programming*

   Pair programming adalah proses penulisan program yang dilakukan berpasangan, bisa oleh dua orang *programmer* atau didampingi oleh *owner*.

8. *Collective Ownership*

   Kode program harus dipahami betul oleh tim *programmer*, hal tersebut dapat terjadi dengan berbagi pengetahuan untuk setiap baris program bahkan tidak hanya sekedar berbagi, namun hak akses untuk saling melengkapi baris program dapat dilakukan.

9. *Continous Integration*

   Dengan melakukan *build* sesering mungkin dalam tiap hari kerja, berbagi kesalahan dalam program dapat dideteksi serta diperbaiki dengan cepat.

10. *Coding Standards*

    Diperlukan adanya kegiatan menyamakan keseluruhan *coding* program dengan kesepakatan tim kerja. Hal ini dapat mempermudah dalam proses akhir pengembangan sehingga hasil yang diperoleh dapat sesuai dengan tujuan.

11. *On Site Customer*

    Adanya anggota klien yang selalu terlibat dalam tahapan pengembangan menjadikan tujuan utama dari pengembangan perangkat lunak dapat terwujud melalui diskusi tanya jawab.

2.9. *Kuchiyose No Jutsu*

Kuchiyose No Jutsu merupakan suatu jurus ninja dalam komik fiksi karangan Masashi Kishimoto, Naruto Shippuden. Jurus ini merupakan jurus yang dapat memanggil berbagai mahluk (hewan/orang/mahluk fiksi lainnya) ke lokasi pemanggilan.

Gambar 2.10. Jurus Kuchiyose No Jutsu