BAB II
LANDASAN TEORI

Pada bab ini penulis akan menguraikan tentang teori-teori yang melandasi penulisan Laporan Tugas Akhir ini.

2.1. Pengantar *File sharing*

Dewasa ini dengan seiringnya perkembangan teknologi *Internet* yang makin pesat, pengguna tidak lagi di batasi oleh ruang dan waktu bahkan batas negara, sekarang mereka dapat melakukan tukar-menukar *file*. Misalnya seorang ilmuwan yang sedang melakukan penelitian di pelosok afrika yang bahkan sulit terjangkau oleh manusia dapat mengirimkan laporan kepada atasannya yang sedang berlibur di sebuah kepulauan Bahama di Amerika Tengah, ilmuwan tersebut cukup melakukan dengan satu buah klik pada mouse untuk menunggungah hasil temuannya tersebut hingga sampai ke tangan seorang atasannya.

2.1.1. Pengertian *File sharing*

File sharing adalah kegiatan untuk mendistribusikan atau menyediakan akses kepada informasi yang disimpan secara *digital* seperti aplikasi komputer, *multimedia* (*audio* ataupun *video*) dokumen elektronik ataupun buku elektronik. *File sharing* dapat di implementasikan melalui berbagai macam media penyimpanan, cara memindahkannya (transmisi) dan model distribusi dan metode umum yang biasanya digunakan untuk *file sharing* adalah menggunakan media penyimpanan yang bisa dipindahkan seperti flashdisk, dvd dan penggunaan penggunaan jaringan *peer-to-peer*(p2p).[2]

2.1.2. *File Hosting*

File Hosting berkaitan erat dengan *web hosting*, yang membedakannya adalah *file hosting* menyimpan data-data seperti dokumen ataupun audio video bukan *web*site ataupun aplikasi *web*. Beberapa keuntungan dari *file hosting* adalah penyimpanan aman dari informasi dan dapat di akses

darimanapun selama pengguna terhubung dengan internet. *File hosting* kebanyakan dipakai untuk menyimpan dan mengarsipkan data sehingga dapat diakses dari bagian manapun didunia ini. Perusahaan besar yang mempunyai banyak cabang biasanya membutuhkan jasa ini.

File layanan hosting, penyedia penyimpanan *file* secara online atau cyberlocker adalah layanan *hosting* internet yang dirancang khusus untuk *host* konten statis, biasanya *file* besar yang bukan halaman *web*. Biasanya mereka mengijinkan *web* dan akses *FTP*. Mereka dapat dioptimalkan untuk melayani banyak pengguna (seperti yang disiratkan oleh istilah "*hosting*") atau dioptimalkan untuk penyimpanan *single-user* (seperti yang tersirat oleh istilah "penyimpanan"). [3]

2.1.3. *File hosting services*

File hosting services adalah alternatif lain dari aplikasi *peer-to-peer*. *File Hosting Service* biasanya digunakan dengan menggabungkan perangkat internet lain seperti *email*, *forum* dan *blog* yang membutuhkan pengunduh langsung dari *file hosting service* yang dapat di-*embedded*-kan. Situs-situs ini biasanya menyimpan data yang dapat diunduh oleh para pengguna.

2.1.4. *One-click hosting*

One-click hosting umumnya menggambarkan layanan hosting *web* yang memungkinkan pengguna internet untuk dengan mudah meng-*upload* satu atau lebih *file* dari hard drive mereka (atau dari lokasi yang jauh) ke *server host* satu klik yang gratis.

Kebanyakan layanan seperti hanya memberikan sebuah URL yang dapat diberikan kepada orang lain, yang kemudian dapat mengambil *file* di kemudian hari. Pada tahun 2005 situs ini telah meningkat drastis popularitas, dan kemudian, banyak, situs-situs yang lebih kecil kurang efisien telah gagal. Banyak internet ada forum untuk berbagi link tersebut; jenis *file sharing* ini, untuk gelar, diambil alih dari jasa berbagi pakai *file* P2P [4].

Situs yang menghasilkan uang melalui iklan atau pengisian untuk layanan *premium* seperti peningkatan kapasitas *download*, menghapus pembatasan menunggu situs mungkin memiliki atau memperpanjang berapa lama *file* upload tetap di situs. Banyak situs tersebut menerapkan CAPTCHA untuk mencegah *download* secara otomatis.

2.2. Sistem Rekomendasi

Bayangkan anda menjadi seorang pegawai perusahaan yang sibuk sehingga anda tidak mempunyai waktu untuk mengetahui dunia luar seperti buku apa yang sedang populer saat ini ataupun film apa yang sedang menjadi *box office*. Anda hanya tinggal menggunakan sistem rekomendasi untuk memberikan anda sebuah rekomendasi buku apa yang cocok dengan anda dan film apa yang mungkin anda sukai berdasarkan buku atau film yang pernah anda beli sebelumnya.

2.2.1. Definisi

Sistem Rekomendasi merupakan model aplikasi dari hasil observasi terhadap keadaan dan keinginan pelanggan. Sistem Rekomendasi memanfaatkan opini seseorang terhadap suatu barang dalam domain atau kategori tertentu, untuk membantu seseorang dalam memilih produk. Karena itu Sistem Rekomendasi memerlukan model rekomendasi yang tepat agar apa yang direkomendasikan sesuai dengan keinginan pelanggan, serta mempermudah pelanggan mengambil keputusan yang tepat dalam menentukan produk yang akan dibelinya [5]

2.2.2. *User-based Collaborative filtering*

User-based nearest neighbour algorithm menggunakan teknik statistika untukmenemukan sekumpulan pengguna, dikenal sebagai tetangga (neighbour), yangmemiliki sejarah setuju dengan pengguna yang menjadi sasaran. Setelahsekumpulan tetangga terbentuk, sistem menggunakan

algoritma yang berbeda untuk menggabungkan kesukaan neighbours untuk menghasilkan prediksi atau rekomendasi N-teratas untuk active user

Algoritma k-nearest neighbor (k-NN atau KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut.

Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data pembelajaran. Sebuah titik pada ruang ini ditandai kelas c jika kelas c merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat titk tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak Euclidean.

Pada fase pembelajaran, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi dari data pembelajaran. Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk data test (yang klasifikasinya tidak diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor data pembelajaran dihitung, dan sejumlah k buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik-titik tersebut.

Nilai k yang terbaik untuk algoritma ini tergantung pada data; secara umumnya, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai k yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, k = 1) disebut algoritma nearest neighbor.

Ketepatan algoritma k-NN ini sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan, atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik.

2.2.3. *Item–based collaborative filtering*

Item–based collaborative filtering merupakan metode rekomendasi yang didasari atas adanya kesamaan antara pemberian rating terhadap suatu produk dengan produk yang dibeli. Tingkat kesamaan produk dihitung, kemudian dibagi dengan parameter kebutuhan pelanggan (yang membutuhkan rekomendasi) untuk memperoleh nilai kegunaan produk. Produk yang memiliki nilai kegunaan tertinggilah yang kemudian dijadikan masukan rekomendasi kepada pengguna.

2.2.4. *Slope One Algorithm*

Slope one termasuk dalam keluarga algoritma *rating-based collaborative* yang memprediksi seorang *user* akan memberikan rating ke sebuah *item* dari perhitungan rating dari *user* lainnya. *Slope one* bekerja menurut perbedaan kepopuleran antara beberapa *item*.

Slope one memperkirakan kesukaan *user* terhadap sebuah *item* untuk beberapa *item* baru berdasarkan perbedaan rata-rata (differensial) antara *item* baru dengan *item* lainnya yang sudah di *rating* oleh pengguna tersebut.

sebagai contoh:

Ilustrasi mari kita asumsi bahwa kita tahu bahwa rata-rata menyukai *video* briptu norman (untuk selanjutnya di singkat norman) lebih besar 1.0 daripada *video* shinta dan jojo (sinjo). Dan mari juga berasumsi seolah-olah kita mengetahui bahwa kebanyakan orang memberikan *rating* sinjo sama dengan norman secara rata-rata. Dan sekarang kita wakilkan dengan seorang *user* yang memberikan rata-rata sinjo 2.0 dan norman 4.0 bagaimana cara menentukan kesukaan *user* tersebut untuk sinjo? Berdasarkan video sinjo kita dapat menebak 2.0 + 1.0 = 3.0, berdasarkan norman kita dapat menebak 4.0+0.0 = 4.0 dengan cara sederhana dari dua *video* tersebut kita dapat menebak 3.5, inilah inti dari *algoritma slope-one recommender*

2.2.5. Penjelasan Algoritma *Slope One*

Dari namanya, kita dapat menyimpulkan bahwa algoritma sistem rekomendasi dimulai dari asumsi adanya hubungan linear antara nilai kesukaan (*preference value*) dari suatu *item* dengan *item* lainnya yang dapat menghasilkan estimasi umum dari nilai kesukaan tersebut. Seperti untuk *item* y didapatkan nilai *preference* dari sebuah *item* x.

Maka dari itu algoritma *slope one* terdiri dari fase pra-proses, dimana didapatkan nilai perbedaan preference value yang di komputasi:[6]

```
for every item i
  for every other item j
    for every user u expressing preference for both i and j
      add the difference in u's preference for i and j to an
      average
```

Algoritma untuk menentukan *item* yang direkomendasikan

```
for every item i the user u expresses no preference for
  for every item j that user u expresses a preference for
    find the average preference difference between j and i
    add this diff to u's preference value for j
    add this to a running average
return the top items, ranked by these averages
```

Algoritma *slope one* sangat atraktif karena pada porsi *media online*, algoritma ini dapat menghitung secara cepat. Seperti *item*-based recommender, performa *slope one* tidak bergantung pada jumlah *user*, tetapi tergantung dari rata-rata perbedaan preference dari setiap pasangan dari *item* tersebut yang di pra-komputasikan. Lebih lanjut struktur data *slope one* dapat terupdate secara langsung, sewaktu *preference* berubah, dengan mudah dapat langsung memperbaharui perbedaan data.

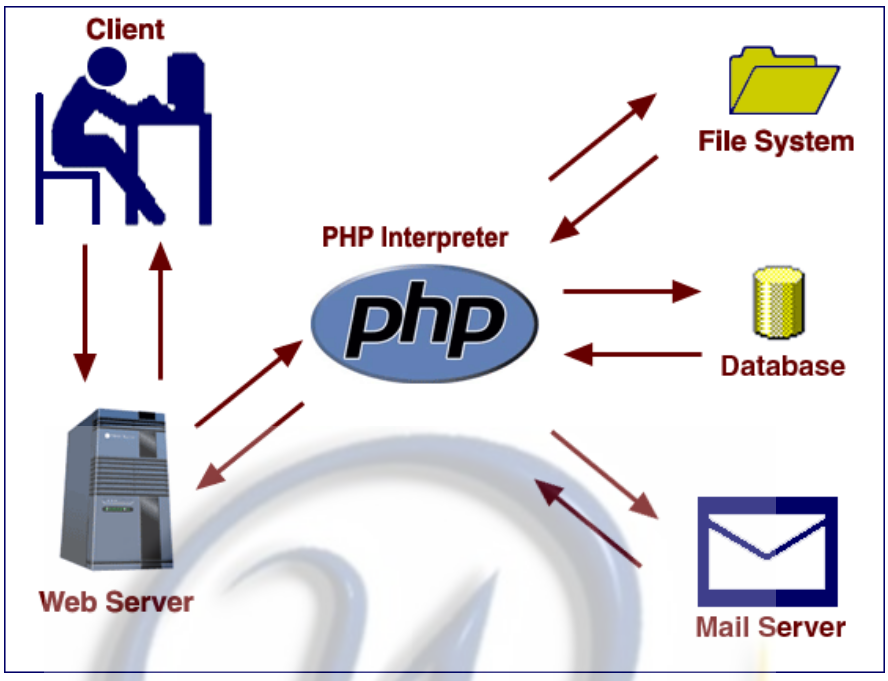
Algoritma *slope one* dianggap paling cocok untuk diterapkan pada aplikasi *file hosting* yang saya bangun karena tidak membutuhkan banyak data untuk melakukan perhitungan, dan juga dapat langsung memberikan rekomendasi ketika *user* memberikan nilai *rating*.

2.3. PHP

PHP merupakan singkatan rekursif (akronim berulang) dari PHP *Hypertext Preprocessor*. PHP adalah bahasa pemrograman script yang paling banyak dipakai saat ini atau dalam kata lain bisa diartikan sebuah bahasa pemrograman *web* yang bekerja di sisi server (*server side scripting*) yang dapat melakukan konektifitas pada *database* yang di mana hal itu tidak dapat dilakukan hanya dengan menggunakan sintaks-sintaks HTML biasa[7]. PHP banyak dipakai untuk memrogram situs *web* dinamis, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain. Contoh terkenal dari aplikasi PHP adalah phpBB dan MediaWiki (*software* di belakang Wikipedia). PHP juga dapat dilihat sebagai pilihan lain dari *ASP.NET/C#/VB.NET Microsoft, ColdFusion Macromedia, JSP/Java Sun Microsystems,* dan *CGI/Perl*. Contoh aplikasi lain yang lebih kompleks berupa CMS yang dibangun menggunakan PHP adalah *Mambo, Joomla!, Postnuke, Xaraya*, dan lain-lain.

Kelebihan PHP dari bahasa pemrograman lain:

1. Bahasa pemrograman PHP adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaanya.
2. *Web* Server yang mendukung PHP dapat d*item*ukan dimana - mana dari mulai apache, IIS, Lighttpd, nginx, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan developer yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa *scripting* yang paling mudah karena memiliki referensi yang banyak.
5. PHP adalah bahasa open source yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah system.



Gambar 2.1 Cara Kerja PHP

Seperti pada gambar 2.1 di atas dijelaskan bahwa PHP adalah aplikasi di sisi server atau dengan kata lain beban kerja ada di server bukan di client. Pada saat browser meminta dokumen PHP, *web* server langsung menggunakan modul PHP untuk mengolah dokumen tersebut. Jika pada dokumen terkandung fungsi yang mengakses database maka modul PHP menghubungi *database server* yang bersangkutan. Dokumen yang berformat PHP dikembalikan *web server* dalam format HTML, sehingga *source code* PHP tidak tampak di sisi *browser*.

2.4. MySQL

MySQL (*My Structure Query Language*) adalah salah satu *database* dari sekian banyak *database* lain seperti Oracle, MS-SQL , PostrgreSQL dan banyak lagi kesemuanya itu mempunyai fungsi dan manfaat yang hampir sama namun dalam pengerjaannya sedikit berbeda tetapi MySQL penggunaannya paling mudah.

2.4.1. Kenapa MySQL?

MySQL adalah *Database Management System* (DBMS), merupakan salah satu system dalam mengakses database yang menggunakan bahasa SQL. MySQL menggunakan bahasa SQL dan dapat dikatakan sebagai DBMS[8]

1. ***MYSQL Software Open Source***

Open source artinya memungkinan untuk semua orang yang menggunakan dan memodifikasi software. Setiap orang dapat men-*download* MySQL dari internet dan menggunakannya tanpa membayar apapun. Jika mau, anda bisa mempelajari kode sumber dan menukar apa yang anda inginkan.

2. **Kenapa menggunakan MySQL?**

Database MySQL sangat cepat, *reliable* dan mudah untuk digunakan, selain itu MySQL telah banyak digunakan dalam pembuatan *software* besar. MySQL merupakan *Relational Database Management Sistem* (RDBMS) yang ddistribusikan secara gratis di bawah lisensi GPL (*General Public License*). Di mana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat closed source atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structure Query Language*). SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan/seleksi dan pemasukan data, yang memungkinkan pengoperasian data dkerjakan dengan mudah secara otomatis. Keandalan suatu *system database* (DBMS) dapat diketahui dari cara kerja *optimizer*-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh *user* maupun program-program aplikasinya. Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibandingkan dengan *database server* yang lainnya dalam *query data.*

3. **Keistimewaan MySQL**

Sebagai *database* yang memiliki konsep database modern, MySQL memiliki banyak sekali keistimewaan. Berikut ini beberapa keistimewaan yang dimiliki oleh MySQL :

1) *Portability*. MySQL dapat berjalan stabil pada berbagai sistem operasi di antaranya adalah seperti *Windows, Linux, FreeBSD, Mac OS X server, Solaris, Amiga, HP-UX* dan masih banyak lagi.
2) *Open Source*. MySQL didistribusikan secara open source (gratis), di bawah lisensi GPL.
3) *Multiuser*. MySQL dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik. Hal ini memungkinkan sebuah database server MySQL dapat diakses *client* secara bersamaan.
4) *Performance Tuning*. MySQL memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5) *Column Types*. MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed/unsigned integer, float, double, char, varchar, text, blob, date, time, datetime, year, set* serta *enum*.
6) *Command dan Function*. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *SELECT* dan *WHERE* dalam *query*.
7) *Security*. MySQL memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, dan *user* dengan system perizinan yang mendetail serta password terencripsi.
8) Stability dan Limits. MySQL mampu menangani *database* dalam skala besar, dengan jumlah records lebih dari 50 juta dan 60 ribu table serta 5 miliar baris. Selain itu, batas indeks yang dapat di tampung mencapai 32 indeks pada tiap tabelnya.
9) *Connectivity*. MySQL dapat melakukan koneksi dengan client menggunakan protocol TCP/IP, Unix soket (Unix), atau Named Pipes (NT).
10) *Localisation*. MySQL dapat mendeteksi pesan kesalahan (error code) pada *clent* dengan menggunakan lebih dari dua puluh bahasa. Meski demikian, bahasa Indonesia belum termasuk di dalamnya.

11) *Interface*. MySQL memiliki *interface* (antar muka) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
12) *Client* dan *Tools*. MySQL dilengkapi dengan berbagai tool yang dapat digunakan untuk administrasi *database*, dan pada setiap *tool* yang ada disertai petunjuk online.
13) Struktur Tabel. MySQL memiliki struktur table yang lebih fleksibel dalam menangani *ALTER TABLE*, dibandingkan *databse* lainnya semacam *PostgreSQL* ataupun *Oracle*.

4. SQL Query

Structured Query Language (SQL) adalah bahasa yang khusus digunakan untuk mengoperasikan database. Untuk memudahkan pelajaran, *SQL query* akan dikelompokkan menjadi tiga: (1) *Query* untuk mengelola *database*, (2) *Query* untuk mengakses data dalam satu table, dan (3) *Query* yang melibatkan lebih dari satu table

a. *Query* pengelolaan *database*

Hal-hal yang termasuk ke dalam kelompok ini adalah query yang bertujuan untuk:

(1)Membuat *database*, (2) Menghapus *database*, (3) Membuat *table*, (4)Memodifikasi *table*, (5) Menghapus *table*, (6) Menambah *user*, (7) Mengatur *permission*, dan (8) Menghapus *user*.

b. *Query* satu tabel

Query satu tabel digunakan untuk mengelola data dalam satu tabel. Beberapa hal yang dapat dilakukan pada satu table adalah:

Tujuan	*Query*
Memasukkan data	*Insert*
Memodifikasi data	*Update*
Mengambil data	*Select*
Menhitung banyaknya data	*Count*
Menghitung penjumlahan data	*Sum*
Menghitung nilai minimal	*Min*
Menghitung nilai rata-rata	*Avg*

Tabel 2. 1 *Query* tabel

2.5. Pemrograman Berorientasi Objek Menggunakan *Unified Modeling Language* (UML)

Pemrograman berorientasi objek (Inggris: *object-oriented programming* disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

2.5.1. Pemrograman Berorientasi Objek (*Object Oriented Programming*)

Dalam program berbasiskan pada objek, sebuah program dibagi menjadi bagian-bagian kecil yang disebut dengan objek. Setiap objek memiliki entiti yang terpisah dengan entiti objek-objek lain dalam lingkungannya. Objek-objek yang terpisah ini dapat diolah sendiri-sendiri dan setiap objek memiliki sekumpulan sifat dan metode yang melakukan fungsi tertentu sesuai dengan yang telah kita programkan kepadanya.

Setiap objek mengandung tiga hal utama seperti dibawah ini:

1. Properti atau Atribut, properti atau atribut adalah karakteristik atau sifat dari sebuah objek.
2. Metode, metode adalah serangkaian prosedur yang dimiliki oleh suatu objek yang akan dijalankan sesuai dengan respon yang diberikan oleh suatu perintah atau kejadian.
3. *Event, event* adalah kejadian atau segala sesuatu yang dapat dialami oleh sebuah objek.

Objek-objek dibuat secara terpisah dan masing-masing memiliki properti serta metode sendiri-sendiri. Setiap objek bisa memiliki metode dan properti yang berbeda satu dengan yang lainnya. Tetapi ada pula dua atau lebih objek yang memiliki metode yang sama. Objek-objek seperti ini harus dibuat dari satu kelas yang sama. Dengan memakai fasilitas ini, pembuatan objek yang bersifat sama tidak perlu dilakukan berulang-ulang.

Pemrograman berorientasi objek memiliki beberapa kelebihan dari pemrograman linier, diantaranya seperti:

1. Lebih cepat dari pemrograman linier karena tidak perlu mengetikkan kode program untuk setiap objek.
2. Resiko kesalahan kecil karena melakukan pengetikan lebih sedikit dan beberapa objek sudah disediakan di dalam menu pilihan yang tinggal dipilih sesuai kebutuhan.
3. Bisa dilakukan daur ulang. Setiap objek dapat digunakan secara berulang-ulang dalam program yang sama maupun program yang berlainan.

1. Object

Object merupakan entitas yang mampu menyimpan state (informasi) dan menyediakan sejumlah operasi (behavior) yang mempengaruhi state. Model berorintasi objek berisi sejumlah objek, objek biasanya berhubungan dengan entitas objek pada kehidupan nyata seperti orang. Setiap objek berisi informasi individu, contohnya orang memiliki nama.

Objek memiliki relasi dengan objek lainnya. Terdapat dua jenis relasi, yang pertama relasi statik dimana kedua objek mengetahui keberadaan satu dan lainnya. Kedua relasi dinamik dimana kedua objek berkomunikasi satu dan lainnya.

2. *Class* dan *Instance*

Class mewakili *template* untuk beberapa objek dan menggambarkan bagaimana objek terstruktur dari dalam. Objek dari *class* yang sama memiliki definisi yang sama, baik oprasinya maupun struktur informasinya.

Instance adalah objek yang terbuat dari *class*. *Class* menggambarkan sruktur dari *instance*, yang mana *current state* dari *instance* bergantung pada operasi yang dilakukan oleh *instance*.

2.5.2. Karakteristik Metodologi Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek memiliki tiga karakteristik utama[9], yaitu:

1. *Encapsulation*

Encapsulation (pengkapsulan) merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas bersama-sama dalam satu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada dalam objek itu sendiri.

2. *Polymorphism*

Polymorphism yaitu konsep yang menyatakan bahwa suatu yang sama dapat mempunyai bentuk dan prilaku berbeda. Polymorphism mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda. Suatu implementasi yang spesifik pada suatu oprasi dari *class* tertentu disebut metoda. Karena operator berorientasi objek adalah bersifat pilymorphism, mungkin dapat mempunyai lebih dari satu metoda

3. *Inheritance*

Inheritance adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metoda dari induknya langsung. Atribut dan metoda dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pendefinisian objek dipergunakan untuk membangun suatu hirarki dari objek turunannya, sehingga tidak perlu membuat atribut dan metoda lagi pada anaknya, karena telah mewarisi sifat induknya.

Inheritance merupakan pewarisan srtuktur dari parent calss ke child *class*. Jika *class* B inherit *class* A, maka oprasi dan struktur informasi dalam *class* A akan menjadi bagian dari *class* B

2.6. Metode Rekayasa Perangkat Lunak dengan *Scrum*

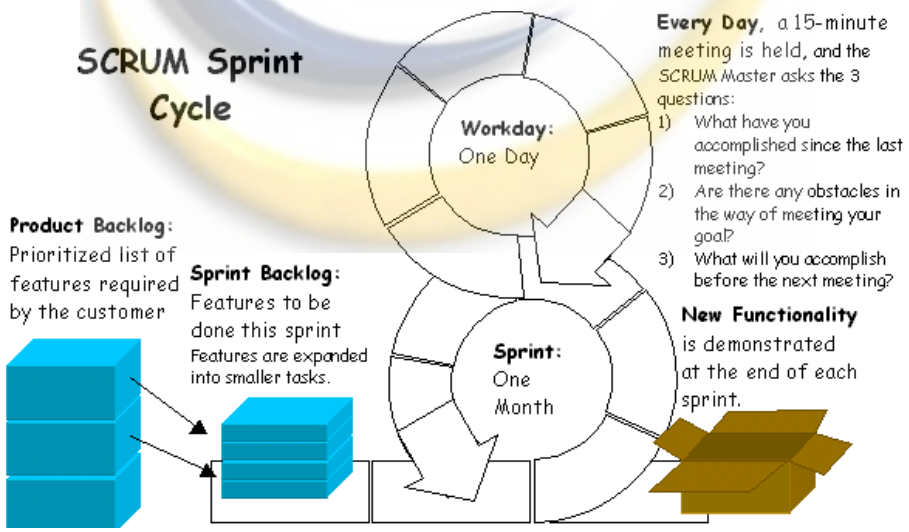
Scrum telah digunakan untuk mengembangkan produk yang kompleks semenjak awal tahun 1990. Scrum bukanlah sebuah proses atau teknik untuk membuat sebuah produk, melainkan Scrum adalah sebuah kerangka kerja dimana didalamnya anda dapat memasukkan berbagai macam proses dan teknik. Peran Scrum adalah untuk memunculkan praktek pengembangan yang sedang kita gunakan ke permukaan sehingga kita dapat membuatnya lebih baik yang pada saat bersamaan praktek tersebut akan menjadi sebuah kerangka kerja untuk mengembangkan produk yang rumit.

Metode scrum termasuk ke dalam metode agile. Karena metode Agile menggunakan prinsip perulangan. maka selama waktu pengerjaannya akan selalu dijumpai proses pengembangan yang dilakukan berulang. Setiap perulangan (iterasi) meliputi berbagai kegiatan yang wajib dilakukan dalam proyek pengembangan sofware itu sendiri, yaitu perencanaan, requirements *analysis, desain, coding, testing*, dan dokumentasi. Maka jelas bahwa satu kali iterasi bertujuan menghasilkan modul atau fungsionalitas yang deliverable pada client. Setiap iterasi umumnya berlangsung dalam jangka waktu 1 sampai 4 minggu. Sedangkan pada scrum dikenal istilah sprint duration yaitu rentang waktu satu kali iterasi, jadi jika suatu tugas atau masalah tidak di temukan solusi sampai sprint selesai maka tugas tersebut akan di kembalikan dalam proses backlog dan akan di

sertakan pada proses sprint berikutnya. Prinsip-Prinsip Agile Software Development adalah:

1. Prioritas tertinggi adalah memuaskan keinginan customer melalui proses deliver product secara dini dan berkesinambungan.
2. Menerima adanya perubahan requirement, meskipun akan membuat proses development menjadi terlambat. Agile process memaklumi perubahan tersebut demi kepentingan customer agar lebih kompetitif.
3. Berulang kali melakukan deliver software, yang merupakan hasil dari beberapa minggu atau bulan proses development, dengan preferensi pada jadwal yang lebih singkat.
4. Analis dan developer harus bekerja sama setiap hari sepanjang pengerjaan project.
5. Produk dibangun oleh individu-individu yang memiliki motivasi tinggi. Memberikan mereka environment dan dukungan yang dibutuhkan, serta memberikan kepercayaan atas tercapainya target yang diberikan.

2.6.1. Penjelasan Metodologi *Scrum*



Gambar 2.2 ***Scrum lifecycle***

Gambar 2.2 menggambarkan *lifecycle* dari proses scrum, Langkah-langkah dalam proses Scrum antara lain :

1. Sprint Planning Meeting (Product Owner, Customer, Scrum Master), hasil : Product Backlog berikut skala prioritasnya.
2. Scrum Master dan anggota tim memecah Product Backlog menjadi task-task yang lebih kecil (Sprint Backlog), dan kemudian memilih task-task dari Sprint Backlog yang akan dikerjakan beserta estimasi waktunya untuk 1 sprint duration.
3. Tim mengerjakan Sprint Backlog yang sudah dipilih.
4. Scrum Master memfasilitasi Daily Scrum.
5. Lakukan Sprint Review dan Sprint Restrospective setiap selesai satu Sprint.
6. Iterative Development. Product dibangun melalui beberapa proses iterasi, yang di setiap iterasinya akan menambahkan fitur tertentu, dan dapat diperlihatkan : demo program, dokumentasi, desain.
7. Anggota project dibagi menjadi beberapa tim yang disebut Sprint Teams yang terdiri hingga 7 atau 8 orang. Masing-masing tim diberikan keleluasaan untuk menggunakan metode development atau tools yang dianggap terbaik untuk memenuhi target.
8. Sprint Planning Meeting. Rapat ini dipimpin oleh Product Owner. Dilakukan untuk mendeskripsikan fitur-fitur yang diinginkan berikut skala prioritasnya. Setelah itu tim akan menentukan fitur mana saja yang akan dipenuhi dan fitur mana yang tidak. Disinilah ditentukan goal dari system , untuk kemudian tim akan mempersiapkan daftar Product Backlog yang akan dijadikan acuan.
9. The Daily Scrum Daily scrum biasanya berdurasi sekitar 15 menit setiap harinya. Dilakukan di waktu dan tempat yang sama untuk menjaga konsistensi. Idealnya Scrum Team membentuk sebuah lingkaran, dan duduk saling berhadapan dengan anggota tim yang lain. Setiap tim diharuskan menjawab pertanyaan-pertanyaan di bawah ini :

1. Apa yang sudah dikerjakan/dicapai semenjak Daily Scrum terakhir?
2. Kendala apa yang ditemukan untuk mencapai target?

3. Apa saja yang akan dikerjakan/dicapai sebelum Daily Scrum berikutnya?

10. Sprint Review

Rapat ini dilakukan setiap akhir dari sebuah Sprint. Selama rapat, masing-masing tim mendemonstrasikan hasil yang dicapai sesuai target pada sprint tersebut. Sprint Review ini sangat direkomendasikan untuk dilakukan secara informal, sehingga diharapkan tidak membebani tim secara psikologis. Sprint Review biasanya dihadiri oleh Product Owner, Scrum Master, Customer, Management, dan Scrum Team sendiri. Dalam rapat ini akan dibandingkan hasil Sprint dengan Sprint Goal yang telah disepakati saat Sprint Planning Meeting berdasarkan backlog yang terikat dengan Sprint yang bersangkutan.

11. Sprint Restrospective

Rapat ini difasilitasi oleh Scrum Master, untuk mendiskusikan dan menyimpulkan hasil dari Sprint yang baru saja dilakukan. Selain itu rapat ini juga dilakukan untuk membicarakan hal-hal yang dapat membuat Sprint berikutnya lebih efektif dan menyenangkan bagi seluruh anggota tim. Sprint Review digunakan untuk melihat perkembangan system yang dibangun, sedangkan Spring restrospective digunakan untuk melihat perkembangan tim yang membangun system.

Semua hal yang mempengaruhi bagaimana tim mengembangkan system dapat dibicarakan, seperti : factor komunikasi, environtment, tools, dll.

Sprint Restrospective merupakan satu tool yang sangat penting untuk membuat tim semakin berkembang seiring pengembangan system.

12. Product Backlog

Merupakan sebuah daftar requirement, beserta skala prioritas, dan estimasi waktu yang diperlukan untuk penyelesaian/implementasi. Estimasi waktu biasanya dalam satuan hari, dan biasanya tersusun mulai dari prioritas tertinggi. Dimana prioritas tertinggi ditentukan berdasarkan hal yang dinilai krusial atau penting dengan sistem yang akan dibangun. Daftar ini dapat secara dinamis berubah, bergantung pada bisnis rule dan teknologi yang digunakan.

13. Sprint Backlog

Sprint Backlog adalah sekumpulan task yang akan dikerjakan oleh scrum team pada setiap sprint. Sprint Backlog dibuat berdasarkan Product Backlog. Dan biasanya lebih detil daripada Product Backlog. Sprint Backlog ini lah yang sebenarnya lebih dicek pada Daily Srum untuk dikonfontir dengan Product Backlog yang bersangkutan.

2.6.2. *Role-role* yang terdapat pada Scrum

Ada beberapa peran penting yang berada pada pengembangan scrum, setiap orang memiliki perannya masing-masing

1. The Product Owner

Product ownerlah yang memiliki definisi terhadap berhasil atau tidaknya pembangunan sebuah product. Dia akan memimpin dan mengarahkan pengembangan product, sprint by sprint (iterasi demi iterasi) demi tercapainya target. Dia jugalah yang menentukan skala prioritas, release plans, dan pemilik(assingment) dari setiap product backlog yang ada. Selain itu dia juga membuat development timeschedule berdasarkan skala prioritas backlog.

Hanya satu orang yang ditempatkan pada role ini dengan harapan hanya ada satu orang yang menentukan requirement yang akan dipenuhi. Requirement ini dapat diperoleh berdasarkan pertemuan dengan client, sales, dll. Tetapi yang menjadi poin penting adalah bahwa hanya Product Owner lah yang menentukan skala prioritas dan validitas requirement. Mekanisme ini akan meng-eliminasi adanya kebingungan anggota tim terhadap perbedaan asumsi requirement, opini yang berkembang, dan dari gangguan-gangguan lainnya.

2. The Scrum Master

Scrum Master adalah seorang yang memfasilitasi / menangani tim pada process development harian. Scrum Master tidak mempunyai tugas lain karena pada dasarnya pekerjaan tersebut akan memakan keseluruhan

waktu kerja yang dimiliki. Scrum master bertanggung jawab dalam memastikan bahwa setiap tim menerapkan prinsip dan nilai-nilai dalam Scrum. Scrum master bertanggung jawab memecahkan setiap masalah yang timbul selama proses development yang ditemukan pada project management meeting. Role ini umumnya diisi oleh project manager atau technical team leader.

3. The Sprint

Merupakan rentang waktu yang telah ditetapkan untuk menghasilkan suatu incremental product. Sprint meliputi : desain, koding, testing, dan dokumentasi.

Special Release Sprint merupakan sprint khusus yang dilakukan untuk memperiapkan product yang akan dinaikkan ke level production. Tidak seperti sprint pada umumnya, sprint ini hanya mempersiapkan hal-hal yang berhubungan dengan persiapan production, seperti mem-*package* aplikasi, mempersiapkan environtment yang dibutuhkan, dll.

2.7. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem yang terkait dengan objek bahasa pemodelan UML lebih cocok untuk pembuatan perangkat lunak dalam bahasa pemrograman berorientasi objek (C++, Java, VB.NET), namun demikian tetap dapat digunakan pada bahasa pemrograman prosedural [11], UML biasa digunakan untuk:

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum, dibuat dengan *use case* dan aktor.
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan interaction diagrams.
3. Menggambarkan representasi struktur statik sebuah sistem dalam bentuk *class* diagrams.
4. Membuat model behavior ”yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan state transition diagrams.

5. Menyatakan arsitektur implementasi fisik menggunakan component and development diagrams.
6. Menyampaikan/memperluas fungsionality dengan stereotypes

Pemodelan menggunakan UML merupakan metode pemodelan berorientasi objek dan berbasis visual. Karenanya pemodelan menggunakan UML merupakan pemodelan objek yang fokus pada pendefinisian struktur statis dan model sistem informasi yang dinamis daripada mendefinisikan data dan model proses yang tujuannya adalah pengembangan tradisional. UML menawarkan diagram yang dikelompokan menjadi lima perspektif berbeda untuk memodelkan suatu sistem. Seperti satu set blue print yang digunakan untuk membangun sebuah rumah.

Penjelasan mengenai berbagai diagram UML serta tujuannya dapat dijelaskan sebagai berikut:

1. Model *Use Case Diagram*

Use case diagram secara grafis menggambarkan interaksi antara sistem, sistem eksternal, dan pengguna. Dengan kata lain use case diagram secara grafis mendeskripsikan siapa yang akan menggunakan sistem dan dalam cara apa pengguna (*user*) mengharapkan interaksi dengan sistem itu. Use case secara naratif digunakan untuk secara tekstual menggambarkan sekuensi langkah-langkah dari setiap interaksi. Use case diagram dibuat untuk memvisualisasikan/menggambarkan hubungan antara aktor dan use case. Aktor adalah para pengguna (*user*s) dari sebuah sistem/seseorang atau sesuatu yang harus berinteraksi dengan sistem atau sistem yang dibangun/dikembangkan.. Kadangkala sebuah sistem merupakan aktor bagi sistem yang lain, beri nama aktor sistem tersebut dengan stereotipe (bentuk klise/tiruan) aktor.

2. Diagram Struktur Statis

UML menawarkan dua diagram untuk memodelkan struktur statis sistem informasi, yaitu:

a. *Class* Diagram: menggambarkan struktur object sistem. Diagram ini menunjukan *class* object yang menyusun sistem dan juga hubungan antara *class* object tersebut. *Class* merepresentasikan sebuah abstraksi dari entitas-entitas dengan sifat-sifat atau karakteristik yang bersifat umum.
b. Object Diagram: serupa dengan *class* diagram, tetapi object diagram memodelkan isntance object actual dengan menunjukan nilai-nilai saat ini dari atribut instance. Object diagram menyajikan "snapshot/potret" tentang objek sistem pada point waktu tertentu. Diagram ini tidak digunakan sesering *class* diagram, tetapi saat digunakan dapat membantu seorang developer memahami struktur sistem secara lebih baik.

3. Diagram Interaksi

Diagram interaksi memodelkan sebuah interaksi, terdiri dari satu set objek, hubungan-hubungannya, dan pesan yang terkirim di antara objek. Model diagram ini memodelkan behavior (kelakuan) sistem yang dinamis dan UML memiliki dua diagram untuk tujuan ini, yaitu:

a. Diagram rangkaian/Sequence Diagram: secara grafis menggambarkan bagaimana objek berinteraksi dengan satu sama lain melalui pesan pada sekuensi sebuah use case atau operasi. Diagram ini mengilustrasikan bagaimana pesan terkirim dan diterima di antara objek dan dalam sekuensi atau timing apa.
b. Diagram kolaborasi/Collaboration Diagram: serupa dengan diagram rangkaian/sekuensi, tetapi tidak fokus pada timing atau sekuensi pesan. Diagram ini justru menggambarkan interaksi (atau kolaborasi) antara objek dalam sebuah format jaringan. Diagram rangkaian maupun diagram kolaborasi merupakan isomorphic artinya kita dapat mengubah dari satu diagram ke diagram lain.

4. Diagram State/State Diagram

a. UML memiliki sebuah diagram untuk memodelkan behavior objek khusus yang kompleks (statechart) dan sebuah diagram untuk memodelkan behavior dari sebuah use case atau sebuah metode, yaitu:

b. Diagram statechart: digunakan untuk memodelkan behavior objek khusus yang dinamis. Diagram ini mengilustrasikan siklus hidup objek berbagai keadaan yang dapat diasumsikan oleh objek dan event-event (kejadian) yang menyebabkan objek beralih dari satu state ke state lain.

c. Diagram aktivitas/Activity Diagram: secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun use case. Activity diagram dapat juga digunakan untuk memodelkan action yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari action tersebut.