



### **Hak cipta dan penggunaan kembali:**

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

### **Copyright and reuse:**

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Beras**

Beras menjadi pangan hampir seluruh penduduk Indonesia. Pangan beras menjadi pangan pokok favorit semua golongan, kaya dan miskin. Berdasarkan sisi gizi dan nutrisi, beras memang relatif unggul dibandingkan dengan pangan lain. Seluruh bagian beras bisa dimakan. Kandungan energinya mencapai 360 kalori per 100gram. Beras adalah sumber protein yang baik dengan kandungan protein 6,8 gram per 100gram. Itulah sebabnya, di Indonesia, dalam neraca makanan, sumbangan beras terhadap energi dan protein masih sangat tinggi: lebih dari 55 persen. Seseorang yang makan beras dalam jumlah cukup pasti tidak akan kekurangan protein (Suhartiningsih, 2004).

Beras bahan makanan yang dihasilkan oleh padi. Meskipun sebagai bahan makanan pokok, beras dapat disubsitusi oleh bahan makanan lainnya, namun padi memiliki nilai tersendiri bagi orang yang biasa makan nasi dan tidak dapat mudah digantikan oleh bahan makanan lainnya (Suparyono dan Agus, 1993).

##### **2.1.1. Mutu Beras**

Beras yang dijual di pasar bermacam-macam jenisnya dan berbeda-beda pula mutunya. Berikut dikemukakan secara umum

kriteria dan pengertian mutu beras yang meliputi mutu pasar, mutu rasa, mutu tanak (Haryadi, 2006).

Tinggi rendahnya mutu beras bergantung pada beberapa faktor, yaitu spesies dan varietas, kondisi lingkungan, waktu dan cara pemanenan, metode pengeringan, dan cara penyimpanan (Astawan, 2004).

Di Indonesia, tingkat mutu didasarkan antara lain pada kesepakatan oleh sebagian besar pedagang beras. Tingkatan mutu yang berlaku di masyarakat sangat beragam.

Berikut ini beberapa ciri yang sering menjadi dasar pengelompokan beras yaitu (Haryadi, 2006):

1. Asal daerah, seperti beras Cianjur, beras Solok, beras Delanggu dan beras Bayuwangi.
2. Varietas padi, misalnya beras Rojolele, beras Bulu dan beras IR.
3. Cara pengolahan, dikenal beras tumbuk dan beras giling.
4. Gabungan antara varietas dengan hasil penyosohan pada derajat yang berbeda, yang berlaku untuk suatu daerah. Misalnya di Jawa Tengah dikenal beras TP, SP dan BP; di Jawa Barat dikenal beras TA, BGA, dan TC.

Berikut dikemukakan secara umum kriteria dan pengertian mutu beras meliputi:

1. Mutu Pasar

Mutu beras dipasaran umumnya berkaitan langsung dengan harganya. Setidaknya, harga merupakan patokan yang dapat dipergunakan sebagai pedoman bagi penjual dan pembeli. Dalam kaitan ini, Badan Urusan Logistik (Bulog) telah menetapkan ciri-ciri untuk menetapkan mutu beras yang akan dibeli oleh badan tersebut. Ketentuan mutu tersebut hanya terbatas dalam hubungannya dengan Bulog dan tidak berlaku secara luas dalam perdagangan bebas.

Persyaratan mutu beras yang ditentukan oleh Bulog dapat dikelompokkan menjadi dua, yaitu persyaratan kualitatif dan persyaratan kuantitatif.

Persyaratan kualitatif ditentukan secara subjektif yang meliputi bau, suhu, hama penyakit dan bahan kimia. Persyaratan tersebut tidak dapat ditentukan dalam satu satuan, tetapi dinyatakan dengan membandingkan terhadap contoh. Bau beras yang tidak disenangi adalah bau apek dan bau alkoholik. Bau apek terutama disebabkan oleh hasil perusakan minyak, bau asam dan alkoholik disebabkan oleh hasil fermentasi gula. Pengujian bau dilakukan dengan

membandingkan terhadap contoh yang ditetapkan atau pembanding lainnya. Disyaratkan bahwa pada semua tingkatan mutu, sampel tidak boleh mengandung tanda-tanda keberadaan hama atau penyakit hidup, telur, kepompong, atau jamur baik dalam bentuk spora maupun miselia. Pengamatan dapat dilakukan secara langsung atau dengan kaca pembesar. Pada ketentuan mengenai mutu beras juga dipersyaratkan bahwa beras tidak boleh mengandung sisa-sisa obat antiserangga atau obat antijamur serta bahan kimia lainnya. Keberadaan bahan kimia ini dapat ditentukan dengan pembauan.

Persyaratan kuantitatif beras yang ditetapkan oleh Bulog, sebagian besar menyangkut akibat perlakuan-perlakuan lepas panen (Haryadi, 2006).

## 2. Mutu Tanak

Di Indonesia, mutu tanak belum disajikan syarat dalam menetapkan mutu beras. Lain halnya dengan di dunia internasional, khususnya di Amerika Serikat, mutu tanak merupakan salah satu persyaratan terutama dalam pengolahan

beras. Sifat tanak lebih banyak ditentukan oleh faktor genetik dari pada faktor perlakuan lepas panen, sehingga sifat ini dimasukkan kedalam ciri-ciri varietas.

Ciri-ciri umum yang mempengaruhi mutu tanak ialah perkembangan volume, kemampuan mengikat air, stabilitas pengalengan nasi parboiling, lama waktu penanakan, dan sifat viskositas padi. Namun demikian, pada penetapan ciri mutu tanak dan prosessing, digunakan sifat-sifat fisik dan kimia yang dapat diukur secara objektif dengan cepat, mudah, dan murah. Sifat beras yang digunakan sebagai ciri penentu mutu tanak dan prosessing adalah kadar amilosa, uji alkali untuk menduga suhu gelatinasi, kemampuan pengikatan air pada suhu 70° C, stabilitas pengalengan nasi parboiling dan sifat amilografi.

Sifat-sifat lain yang menentukan tingkat penerimaan kesukaan penduduk di Asia Tengah meliputi pemanjangan biji selama pemasakan. Varietas Basmati yang dikelompokkan sebagai beras bermutu tanak tinggi, mempunyai sifat

pemanjangan yang lebih besar dari pada jenis-jenis beras lainnya (Haryadi, 2006).

### 3. Mutu Rasa

Mutu giling dan mutu pasar tidak selalu berhubungan dengan mutu tanak dan rasa nasi. Oleh sebab itu, mutu pasar yang tinggi tidak memberi jaminan bahwa beras tersebut juga mempunyai harga yang tinggi. Mutu rasa lebih banyak ditentukan oleh faktor subjektif, yang dipengaruhi oleh daerah, suku bangsa, lingkungan, pendidikan, tingkat golongan dan jenis pekerjaan konsumen. Walaupun belum ada ketentuan pasti untuk menetapkan ciri-ciri mutu nasi, akan tetapi pada tingkat pasar, mutu rasa mempunyai kaitan langsung dengan selera dan tingkat kesukaan atau penerimaan konsumen dan dengan harga beras (Juliano, 1994). Dalam perdagangan karena rasa merupakan selera pribadi, rasa tidak dimasukkan kedalam ketentuan persyaratan mutu beras yang bersifat baku. Namun demikian mutu rasa secara tidak langsung sudah termasuk dalam pengelompokan jenis beras atau varietas padi (Haryadi, 2006).

Dalam penentuan mutu rasa nasi dikenal nasi pera dan nasi pulen. Nasi pera adalah nasi keras dan kering setelah dingin, tidak lekat satu sama lain, dan lebih mengembang dari nasi pulen. Nasi pulen ialah nasi yang cukup lunak walaupun sudah dingin, lengket tetapi kelengketannya tidak sampai seperti ketan, antar biji lebih berlekatan satu sama lain dan mengkilat (Haryadi, 2006).

Nasi pulen lebih disukai oleh sebagian besar penduduk Sulawesi, Jawa dan Kalimantan. Penduduk Sumatera lebih menyukai nasi yang agak pera. Pengujian mutu rasa nasi dapat dilakukan secara subjektif dengan uji indrawi dan secara objektif dengan menggunakan uji analisa seperti instron, teksturometer, dan viskoelastograf. Uji indrawi dilakukan dengan menyajikan nasi kepada 10-12 panelis. (Haryadi, 2006).

## 2.2 *Android*

Android merupakan sistem operasi yang digunakan untuk perangkat *mobile* berbasis Linux yang mencakup sistem operasi, *middleware* dan aplikasi utama *mobile*. Pada awalnya sistem operasi ini dikembangkan oleh



Android.Inc, yang kemudian dibeli oleh Google pada tahun 2005. Android mengembangkan usaha pada tahun 2007 dibentuklah *Open Handset Alliance* (OHA), sebuah konsorsium dari beberapa perusahaan, yaitu Texas Instrument, Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, T-Mobile, Packet Video, ARM Holdings, Atheros Communications, Asustek Computer INC, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan VodaFone Group Plc dengan tujuan untuk mengembangkan standar terbuka untuk perangkat *mobile Smartphone*. (Hermawan, 2010).

Android memiliki empat karakteristik sebagai berikut:

1. Terbuka

Android dibangun untuk benar-benar terbuka sehingga sebuah aplikasi dapat memanggil salah satu fungsi inti ponsel seperti membuat panggilan, mengirim pesan teks, menggunakan kamera, dan lain-lain. Android menggunakan sebuah mesin virtual yang dirancang khusus untuk mengoptimalkan sumber daya memori dan perangkat keras yang terdapat di dalam perangkat. Android merupakan *open source*, dapat secara bebas diperluas untuk memasukkan teknologi baru yang lebih maju pada saat teknologi tersebut muncul. *Platform* ini akan terus berkembang untuk membangun aplikasi *mobile* yang inovatif.

## 2. Semua aplikasi dibuat sama

Android tidak memberikan perbedaan terhadap aplikasi utama dari telepon dan aplikasi pihak ketiga (*third-party application*). Semua aplikasi dapat dibangun untuk memiliki akses yang sama terhadap kemampuan sebuah telepon dalam menyediakan layanan dan aplikasi yang luas terhadap para pengguna.

## 3. Memecahkan hambatan pada aplikasi

Android memecah hambatan untuk membangun aplikasi yang baru dan inovatif. Misalnya, pengembang dapat menggabungkan informasi yang diperoleh dari *web* dengan data pada ponsel seseorang seperti kontak pengguna, kalender, atau lokasi geografis.

## 4. Pengembangan aplikasi yang cepat dan mudah

Android menyediakan akses yang sangat luas kepada pengguna untuk menggunakan *library* yang diperlukan dan *tools* yang dapat digunakan untuk membangun aplikasi yang semakin baik. Android memiliki sekumpulan *tools* yang dapat digunakan sehingga membantu para pengembang dalam meningkatkan produktivitas pada saat membangun aplikasi yang dibuat (Nazruddin, 2012).

### 2.3 *Ontology*

Pada dasarnya ada dua bidang ilmu yang menggunakan istilah ontologi, yakni filsafat dan ilmu komputer. Berawal dari filsafat, kemudian istilah ontologi diadaptasi oleh ilmu komputer setelah melewati perdebatan mengenai apa yang dimaksud dengan ontologi hingga akhirnya ada definisi yang banyak dirujuk dalam literatur ilmu komputer (Fensel, Hendler, Lieberman, Wahlseet, 2003).

Untuk membedakan istilah ontologi pada dua bidang ini, digunakan istilah *philosophical ontology* yang mengarah pada filsafat, dan *computational ontology* yang mengacu pada ontologi di ilmu komputer atau ada pula menyebutkannya sebagai *information systems ontology*. Sebagian berpendapat bahwa *computational ontology* merupakan *applied philosophy*. Pembahasan ontologi yang dimaksud dalam tugas akhir ini ialah mengenai *computational ontology* (Gruber, 2007).

*Ontology* merepresentasikan dunia nyata secara terstruktur dan sistematis. *Ontology* menyediakan sebuah model referensi untuk domain-domain *ontology* tersebut. Model referensi merupakan sekumpulan istilah yang dapat digunakan untuk menyederhanakan komunikasi antar *domain experts* dan meningkatkan pemahaman dan *knowledge sharing*. (Halim, 2017)

Berikut merupakan komponen-komponen dari *Ontology*:

1. *Instances*

*Instances* adalah komponen dasar dari sebuah *ontology*. *Instance* dari sebuah *ontology* dapat berupa objek nyata seperti manusia, hewan, benda atau bisa juga berupa objek abstrak seperti angka dan huruf.

2. *Class*

*Class* menjelaskan konsep yang ada di dalam sebuah *domain*. Biasanya sebuah *class* merupakan kumpulan dari objek. Sebuah *class* juga memiliki turunan dari *class* itu yang mempresentasikan konsep yang lebih spesifik dari *class* atasnya.

3. *Attributes*

Objek-objek yang berada dalam *ontology* dapat dideskripsikan dengan memberikan tambahan atribut ke dalam objek tersebut. Setiap atribut memiliki setidaknya sebuah nama dan nilai dari nama tersebut. Hal tersebut digunakan untuk menyimpan informasi yang spesifik tentang objek yang memiliki atribut tersebut.

4. *Relationship*

Sebuah *relationship* menjadi penting dalam sebuah *ontology*. Hal tersebut dikarenakan dalam suatu *ontology*, relasi antar objek yang ada harus bisa mendeskripsikan keunggulan dari *ontology* yang berasal dari kemampuannya dalam mendeskripsikan sebuah relasi. (Halim, 2017)

Ontologi dapat dibedakan menjadi beberapa tipe sebagai berikut:

### 1. *Upper-level ontology*

*Upper-level Ontology* merupakan ontologi yang berupa suatu model umum untuk merepresentasikan apa yang ada di dunia, sangat serupa dengan apa yang diteliti dalam *philosophical theory*. Saat ini ada SUO (*Standard Upper Ontology*) yang dikembangkan oleh IEEE. Namun sangat sulit untuk mencapai kesepakatan dalam menetapkan ontologi yang demikian umum. Beberapa kandidat untuk SUO adalah SUMO (*Suggested Upper Merged Ontology*) dan *Cys upper Ontology* (*OpenCys*). NASA juga mengembangkan upper level ontology yang disebut SWEET (*Sementic Web for Earth and Environmental Terminology*).

### 2. *Domain ontology*

*Domain ontology* merupakan ontologi yang merepresentasikan suatu *domain* tertentu saja. Banyak penelitian yang mengembangkan ontologi di bidang kesehatan atau biologi, seperti *Gene Ontology*, *Cancer Ontology*, dan *Medical Ontology*.

### 3. *Application dan Task Ontology*

*Application dan Task Ontology* merupakan ontologi yang khusus menyatakan *application* dan *task* yang independen terhadap *domain*. Contoh ontologi tipe ini adalah PROTON yang digunakan untuk *knowledge management system* dan selanjutnya dikembangkan pula

untuk *automatic entity recognition* dan *information extraction* dari teks.

(Halim, 2017)

## 2.4 *Web Ontology Language (OWL)*

*Web Ontology Language (OWL)* merupakan suatu bahasa pemrograman web yang dapat digunakan oleh aplikasi yang memiliki fitur mencari, memproses, dan menampilkan informasi. OWL juga biasa digunakan untuk menggambarkan model data. Secara umum, OWL memiliki struktur yang sama dengan RDF. Hal-hal yang membedakan RDF dengan OWL terletak pada fitur *vocabulary* yang lebih kaya pada OWL. OWL dirancang untuk bisa dibaca oleh komputer, sehingga komputer dapat membaca dan mengerti hubungan antara data-data yang ada. (Halim, 2017)

OWL memiliki tiga buah subbahasa sebagai berikut.

### 1. *OWL Lite*

*OWL Lite* biasa digunakan oleh *user* yang membutuhkan suatu klasifikasi hierarki dan *constraint* yang sederhana.

### 2. *OWL DL*

*OWL DL* biasa digunakan oleh *user* yang membutuhkan tingkat ekspresi maksimal dan semua kesimpulan yang dihasilkan dapat dihitung dalam waktu terbatas.

### 3. *OWL Full*

*OWL Full* biasa digunakan oleh *user* yang membutuhkan tingkat ekspresi maksimal dan kebebasan dalam penggunaan *syntax* dari RDF tanpa mempertimbangkan komputasi yang dibutuhkan. (Halim, 2017)

```

<owl:Class rdf:ID="Agent">
  <rdfs:label>Agent</rdfs:label>
  <rdfs:subClassOf rdf:resource="&foaf;Agent"/>
  <rdfs:subClassOf rdf:resource="&loc;
    ThingHasLocationContext"/>
</owl:Class>

<owl:Class rdf:ID="Person">
  <rdfs:label>Person</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Agent"/>
</owl:Class>

```

Gambar 2.1 Contoh OWL

## 2.5 SPARQL

*The Simple Protocol and RDF Query Language* (SPARQL), merupakan bahasa pemrograman SQL, yang berfungsi untuk mengambil data dari RFD. SPARQL bekerja dengan mengambil dari dari RDF *triples* dan *resources* dalam mencocokkan bagian dari *query* dan mengembalikan hasil dari *query*. Bahasa ini khusus digunakan untuk pengembangan *Semantic Web* (Pollock, 2009).

Model data RDF berupa suatu *statement* dalam bentuk *triple* yang terdiri dari subyek, predikat, objek. Untuk mendapatkan informasi dari suatu *graph* RDF dibutuhkan suatu *query*. SPARQL adalah *query* untuk RDF/OWL, *query* ini digunakan untuk mengambil data yang ditulis dengan menggunakan RDF/OWL atau XML. Query ini menggunakan URI (*Universal Resource Identifier*) untuk *me-retrieve* struktur RDF/OWL. SPARQL memungkinkan untuk melakukan beberapa hal yaitu, mengambil nilai dari data yang terstruktur maupun data yang semi terstruktur,

mengembangkan data dengan melakukan *query* terhadap suatu relasi yang tidak diketahui, melakukan *query* operasi *join* yang kompleks pada *database* yang berlainan secara lebih sederhana, dan mengubah suatu data RDF menjadi *vocabulary* yang lain. Hasil dari *query* SPARQL dapat mengembalikan nilai dalam beberapa format data yang antara lain: XML, RDF, dan OWL (Awaludin, 2009).

*Query* ini menggunakan URI untuk mengambil struktur dari RDF/OWL. Bahasa *query* ini hampir sama dengan *query* SQL pada *database* umum lainnya, tetapi SPARQL lebih sederhana dibanding SQL biasa. Berikut merupakan bagian-bagian yang digunakan dalam *query* SPARQL.

#### 1. PREFIX

Pernyataan PREFIX merupakan sebuah metode yang digunakan sebagai petunjuk yang membawa informasi dalam sebuah halaman *web*. Pada dasarnya, PREFIX digunakan untuk menyingkat sebuah sumber data.

#### 2. SELECT

Pernyataan SELECT telah didefinisikan oleh sebuah daftar *variable* yang dikembalikan sebagai hasil dari eksekusi *query*. Setiap *variable* diawal dengan tanda tanya (?).

#### 3. WHERE

Pernyataan WHERE didefinisikan sebagai deretan yang dimiliki oleh setiap hasil *query* yang valid. Seluruh pola yang merepresentasikan suatu kalimat RDF harus sesuai dengan RDF *triples*, yaitu terdiri dari



subjek, predikat, dan objek. Ketiga hal tersebut dapat direpresentasikan oleh URI.

#### 4. OPTIONAL

Pernyataan OPTIONAL digunakan untuk mengatasi ketidakcocokan struktur pola *query* dengan pola yang ada pada RDF.

#### 5. FILTER

Pernyataan FILTER digunakan untuk menyaring data sesuai dengan kebutuhan dari *user*, sehingga data yang ditampilkan hanya data yang memenuhi syarat dari FILTER

```
PREFIX fb:<http://rdf.freebase.com/ns/>

SELECT ?film ?when
WHERE
{
    ?film fb:film.film.initial_release_date ?when .
}
```

Gambar 2.2 Contoh Filter

## 2.6 Protégé

Protégé merupakan sebuah perangkat lunak yang bersifat gratis dan *open source*, yang bertindak sebagai sebuah sistem manajemen pengetahuan dan membangun domain model dan aplikasi *knowledge based*. Protégé memberikan *interface* yang berguna untuk mendefinisikan *ontology*.



**Gambar 2.3 Logo Protégé**

(Sumber: <http://protege.stanford.edu/>)

*Plug-in architecture* dari *tools* Protégé dapat disesuaikan untuk membangun aplikasi berbasis *ontology* yang sederhana maupun kompleks. Tim pengembang dapat mengintegrasikan *output* dari Protégé dengan sistem untuk membangun sebuah *intelligence* (Protégé, 2016, <http://protege.stanford.edu>, diakses pada: 13 Maret 2017).

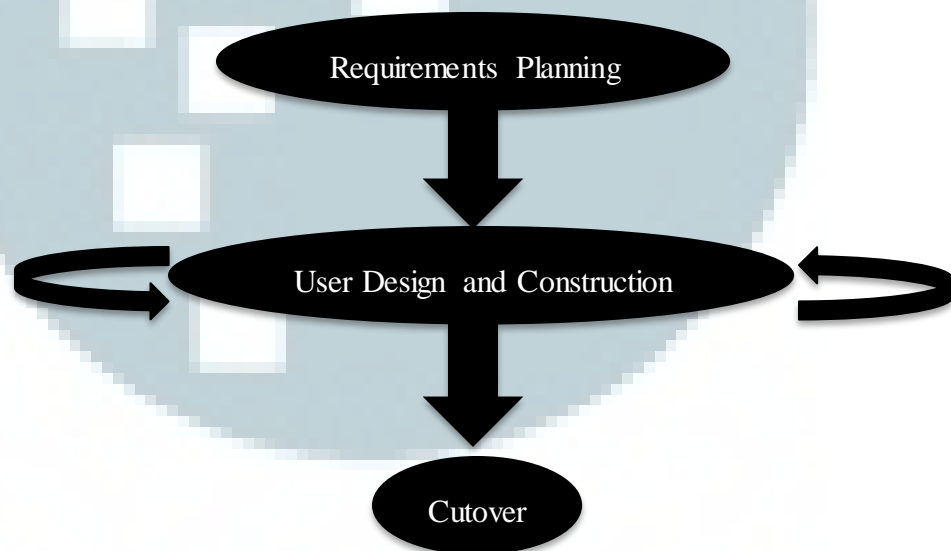
## 2.7 **Rapid Application Development (RAD)**

Menurut Kendall (2010), RAD adalah suatu pendekatan berorientasi obyek terhadap pengembangan sistem yang mencakup suatu metode pengembangan serta perangkat-perangkat lunak. RAD bertujuan mempersingkat waktu yang diperlukan dalam pengembangan sistem tradisional antara perancangan dan penerapan suatu sistem informasi.

Sedangkan Menurut Pressman (2012), RAD adalah proses model perangkat lunak inkremental yang menekankan siklus pengembangan yang singkat. Model RAD adalah sebuah adaptasi “kecepatan tinggi” dari model RAD, di mana perkembangan pesat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen. Jika setiap kebutuhan dan batasan ruang lingkup telah diketahui dengan baik, proses RAD

memungkinkan tim pengembang untuk menciptakan sebuah “sistem yang berfungsi penuh” dalam jangka waktu yang sangat singkat. Dari penjelasan Pressman (2012) ini, satu perhatian khusus mengenai metodologi RAD dapat diketahui, yaitu implementasi metode RAD akan berjalan maksimal jika pengembang aplikasi telah merumuskan kebutuhan dan ruang lingkup pengembangan aplikasi dengan baik.

Menurut Cashman, Shelly, dan Vermaat (2013), terdapat 4 (empat) fase yang terjadi dalam siklus *Rapid Application Development*:



Gambar 2.4 Fase RAD

(Sumber: Cashman, Shelly, Vermaat, 2013)

### 1. *Requirements Planning*

Pertama, pengguna dan analis bertemu untuk menentukan tujuan-tujuan aplikasi atau sistem serta untuk mengidentifikasi kondisi informasi yang ditimbulkan dari tujuan tersebut. Orientasi dalam fase ini adalah menyelesaikan masalah. Meskipun teknologi informasi dan sistem dapat

berubah sebagian dari sistem yang diterapkan, fokusnya akan selalu tetap pada upaya pencapaian tujuan-tujuan perusahaan.

## 2. *User Design*

Pada fase ini, penganalisis merancang dan memperbaiki rancangan sistem yang dibuat, dapat disebut pula sebagai *workshop*. Analis dan pemrogram dapat bekerja membangun dan menunjukkan representasi visual desain dan pola kerja kepada pengguna.

## 3. *Construction*

Lalu, analis dan pemrogram membuat sistem dari hasil desain yang telah disetujui oleh pengguna. Selama fase ini, pengguna akan merespon *prototype* yang ada dan penganalisis akan memperbaiki modul-modul yang dirancang berdasarkan respon pengguna.

## 4. *Cutover*

Pada fase ini, sistem akan disetujui lalu diimplementasikan.

Kemudian akan dilakukan uji coba kepada sistem.

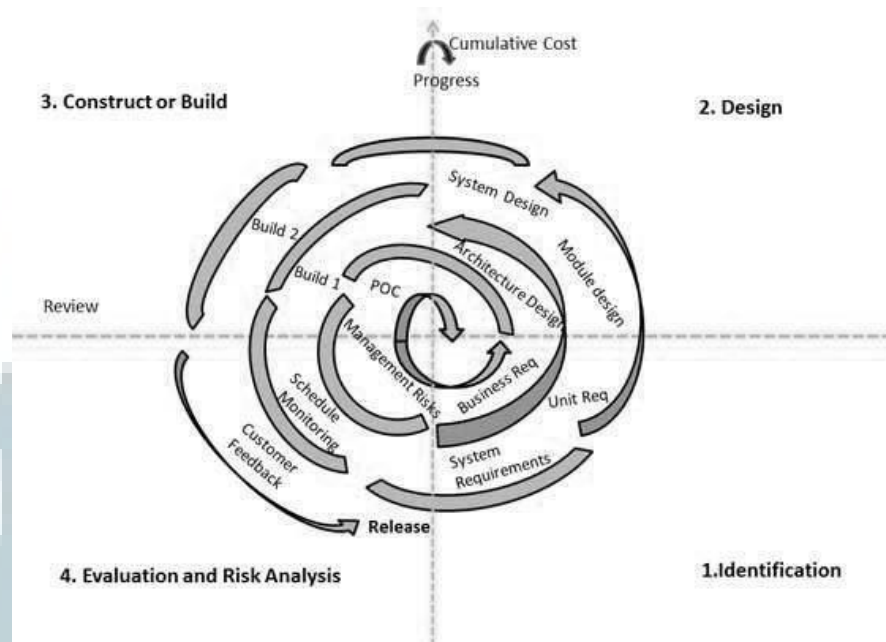
Menurut Pressman (2012), pendekatan RAD memiliki beberapa kelemahan, yaitu:

- a. Untuk proyek yang berskala besar, RAD membutuhkan sumber daya manusia yang cukup untuk membuat tim RAD dengan jumlah yang tepat.

- b. RAD membutuhkan pengembang dan pelanggan yang berkomitmen terhadap aktivitas gerak cepat yang dibutuhkan untuk membuat sistem selesai pada jangka waktu terbatas. Jika komitmen yang dimiliki terbatas, proyek RAD akan gagal.
- c. Tidak semua jenis aplikasi cocok dengan RAD. Jika sistem tidak dapat dimodularisasi, membangun komponen yang penting bagi RAD dapat merepotkan.
- d. RAD tidak layak digunakan ketika risiko teknis tinggi. Hal ini dapat terjadi ketika sebuah aplikasi baru menggunakan teknologi baru secara besar atau ketika *software* baru membutuhkan kerja sama yang tinggi dengan program komputer yang ada.

## 2.8 *Spiral Model*

Model *Spiral* merupakan kombinasi dari proses pengembangan *iterative* dengan model pengembangan secara *linear sequential* seperti model *RAD* dengan pelaksanaan yang sangat tinggi terhadap analisis risiko. Pada model ini, tahapan pengembangan *software* dilewati berulang kali dalam proses iterasi berbentuk *spiral*. Berikut adalah struktur dari model *Spiral* (Poppendieck, 2003).



**Gambar 2.5 Metode Spiral**

(Sumber: en.wikipedia.com)

Pada fase ini, dilakukan pengumpulan kebutuhan bisnis dalam sebuah spiral dasar. Pada spiral berikutnya dilakukan identifikasi kebutuhan sistem sesuai dengan kebutuhan bisnis yang ada. Dalam fase ini juga dilakukan persyaratan dari tiap unit sistem.

### 1. *Design*

Fase ini dimulai dengan membuat desain conceptual dalam spiral yang melibatkan *architectural design*, *logical design*, *physical design*, dan *final design* pada spiral berikutnya.

### 2. *Construct / Build*

Fase ini mengacu pada pembuatan software di setiap spiral.

### 3. *Evaluation and Risk Analysis*

Fase ini mencakup identifikasi, perkiraan, dan pemantauan kelayakan teknis dan manajemen risiko. Setelah

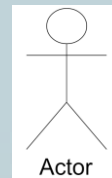
dilakukannya pengujian, pelanggan akan mengevaluasi *software* dan memberikan *feedback*.

## 2.9 Use Case Diagram

*Use Case Diagram* merupakan diagram yang digunakan untuk menampilkan seluruh kegiatan dari sebuah aktor. Berikut adalah komponen-komponen penting yang ada dalam *Use Case Diagram*

### 2.9.1 Aktor

Aktor melambangkan peran dari sebuah *user* dalam sebuah sistem informasi. Seorang aktor dapat melakukan satu atau lebih kegiatan dalam sebuah sistem informasi.



Gambar 2.6 Lambang Aktor

Aktor dilambangkan dengan gambar orang dan diberikan tulisan dibawah gambar orang tersebut. Tulisan itu menggambarkan jabatan yang dimilikinya dalam sebuah sistem informasi, seperti *admin*, *cashier*, *programmer*, dan sebagainya.

### 2.9.2 Use Case

*Use Case* melambangkan kegiatan yang dilakukan oleh aktor.

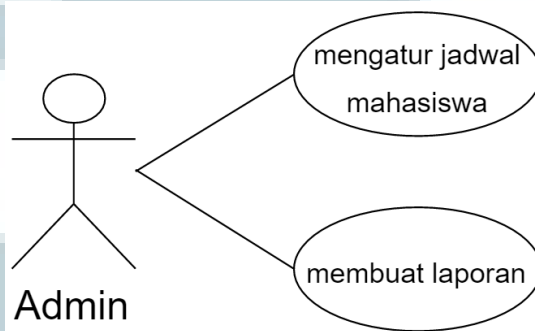


Gambar 2.7 Lambang Use Case

Use Case dilambangkan dengan gambar oval dan diberikan tulisan didalam bentuk oval tersebut. Tulisan itu memberitahu jenis pekerjaan yang dilakukan oleh seorang aktor dalam sebuah sistem informasi. Seperti contoh mengatur jadwal, melakukan pembayaran, menginput data, dan sebagainya. (Whitten & Bentley, 2007).

### 2.9.3 Assossiation

Assossiation melambangkan hubungan antara aktor dengan satu atau lebih use case, sesuai dengan pekerjaan-pekerjaan yang dilakukan oleh seorang aktor.



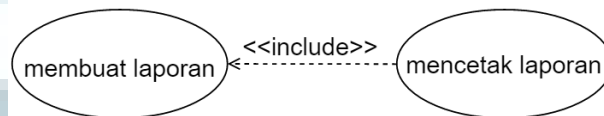
Gambar 2.8 Contoh Penggunaan Assossiation (Garis)

Assossiation dilambangkan dengan bentuk garis yang menghubungkan aktor dengan satu atau lebih use case. Seperti pada gambar 2.7, terdapat sebuah aktor bernama “admin”, yang memiliki tugas untuk mengatur jadwal mahasiswa dan membuat laporan.



#### 2.9.4 *Include Relationship*

*Include* merupakan relasi yang digunakan antara dua buah *use case* untuk melambangkan satu kesatuan. Berikut adalah contoh dari penggunaan *Include*:



**Gambar 2.9** Contoh Penggunaan *Include*

*Include* dilambangkan dengan panah putus-putus dengan tulisan `<<include>>` yang menghubungkan dua buah *use case*. Sesuai dengan gambar 2.8, dapat dilihat bahwa *use case* “mencetak laporan” merupakan kegiatan yang harus dilakukan jika ada kegiatan “membuat laporan” (Whitten & Bentley, 2007).

#### 2.9.5 *Extend Relationship*

*Extend* merupakan relasi yang digunakan antara dua buah *use case* untuk melambangkan pilihan atas *use case*. Berikut adalah contoh dari penggunaan *Extend*:






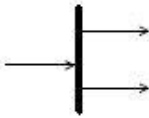
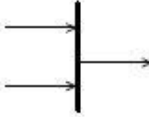

**Gambar 2.10** Contoh Penggunaan *Extend*

*Extend* dilambangkan dengan panah putus-putus dengan tulisan `<<extend>>` yang menghubungkan dua buah *use case*. Sesuai

dengan gambar 2.10, dapat dilihat bahwa dalam melakukan “mengatur jadwal mahasiswa”, seorang aktor bisa melakukan kegiatan (Whitten & Bentley, 2007).

## 2.10 Activity Diagram

Activity diagram adalah sebuah diagram yang menggambarkan tentang aktifitas yang terjadi pada sistem. Dari pertama sampai akhir, diagram ini menunjukkan langkah – langkah dalam proses kerja sistem yang kita buat. Berikut adalah *element – element* pada *activity diagram*:

Simbol	Keterangan
	Start Point
	End Point
	Activities
	Fork (Percabangan)
	Join (Penggabungan)
	Decision
Swimlane	Sebuah cara untuk mengelompokkan activity berdasarkan Actor (mengelompokkan activity dalam sebuah urutan yang sama)

Gambar 2.11 Elemen Activity Diagram

(Sumber: codepolitan.com)

## 2.11 *Class Diagram*

Menurut Dennis, (2012), *Class Diagram* adalah “a static model that supports the static view of the evolving system”. Berbeda dengan Satzinger, (2012), menurutnya *class diagram* adalah “a diagram consisting o classes (i.e., sets of objects) and associations among the classes”. Dapat disimpulkan dari kedua pendapat diatas *class diagram* adalah kumpulan dari kelas-kelas dan merupakan hubungan relasi terstruktur.

## 2.12 **Wawancara**

Menurut Lexy J Moleong, (1989), dengan metode wawancara peneliti dan responden/narasumber berhadapan langsung (tatap muka) untuk mendapatkan informasi secara lisan dengan mendapatkan data tujuan yang dapat menjelaskan masalah penelitian. Namun disini peneliti memilih melakukan wawancara mendalam, ini bertujuan untuk mengumpulkan informasi yang kompleks, yang sebagian besar berisi pendapat, sikap, dan pengalaman pribadi (Sulistyo-Basuki, 2006).

Wawancara adalah suatu percakapan yang diarahkan pada suatu masalah tertentu yang merupakan tanya jawab lisan, ketika dua orang atau lebih berhadap-hadapan secara fisik (*face to face*) untuk mengetahui tanggapan, pendapat, keyakinan, perasaan, dan motivasi seseorang (Gunadi, 1998).

### 2.13 Sistem Pakar

Sistem pakar (*expert system*) adalah sebuah sistem yang menggunakan pengetahuan manusia. pengetahuan tersebut dimasukan ke dalam sebuah komputer dan kemudian digunakan untuk menyelesaikan masalah-masalah yang biasanya membutuhkan kepakaran atau keahlian manusia (Sutojo, Mulyanto, dan Suhartono, 2010). Sistem pakar merupakan bidang yang dicirikan oleh sistem berbasis pengetahuan (*Knowledge Base System*), memungkinkan komputer dapat berfikir dan mengambil keputusan dari sekumpulan kaidah (Ignizio, 1991).

Sistem pakar memiliki beberapa komponen utama, yaitu antarmuka pengguna (*user interface*), basis data sistem pakar (*expert system database*), fasilitas akuisisi pengetahuan (*knowledge acquisition facility*), dan mekanisme inferensi (*inference mechanism*). Selain itu ada satu komponen yang hanya ada pada beberapa sistem pakar, yaitu fasilitas penjelasan (*explanation facility*) (Martin dan Oxman, 1988).

Ada 4 tipe penjelasan yang digunakan dalam sistem pakar, yaitu (Schnupp, 1989):

1. Penjelasan mengenai jejak aturan yang menunjukkan status konsultasi.
2. Penjelasan mengenai bagaimana sebuah keputusan diperoleh.
3. Penjelasan mengapa sistem menanyakan suatu pertanyaan.

4. Penjelasan mengapa sistem tidak memberikan keputusan seperti yang dikehendaki pengguna.

#### **2.14 *Smartphone***

Telepon pintar (*smartphone*) adalah telepon genggam yang mempunyai kemampuan tingkat tinggi, kadang-kadang dengan fungsi yang menyerupai computer (Elcom, 2011). *Smartphone* merupakan teknologi *mobile phone* yang terus berkembang sejak awal kemunculannya dan akan terus mengalami berbagai inovasi untuk memenuhi kebutuhan komunikasi (Chuzaimah, Mabruroh & Fereshti, 2010).

Tingkat penggunaan *smartphone* di zaman sekarang ini sangat tinggi, hal ini dikarenakan untuk memperlancar hubungan komunikasi bagi para kerabat atau keluarga yang jaraknya berjauhan. Fitur dan aplikasi tersebut sudah *built-in* namun dapat ditambahkan secara gratis maupun berbayar melalui situs toko aplikasi yang disediakan oleh masing-masing developer sistem operasi *smartphone*, seperti Google Play (*Android*), AppStore (*iOS*), Windows Market (*Windows Phone*) dan BlackBerry World (*BlackBerry*).

#### **2.15 *Prototype***

Menurut Rainer dan Turban (2009) dalam "*Introduction to Information Systems*", *prototype* adalah versi kecil yang dibuat secara cepat, berdasarkan sistem yang sedang dikembangkan, dimana pengguna

memberikan masukan untuk meningkatkan serta memperbaiki kinerja dari *prototype*.

Menurut O'Brien dan Marakas (2007) dalam "*Enterprise Information Systems*", *prototype* adalah sebuah model kerja, khususnya model kerja dari suatu sistem informasi yang mencakup versi tentatif dari masukan dan keluaran pengguna, *database* dan *file*, metode pengendalian, serta pengolahan rutinitas.

Berdasarkan pengertian dari para ahli diatas, dapat disimpulkan bahwa *prototype* adalah seluruh atau sebagian dari sistem yang sedang dirancang dimana pengguna dapat berperan aktif dalam memberikan kritik dan saran terhadap sistem tersebut.

UMMN