BAB II

LANDASAN TEORI

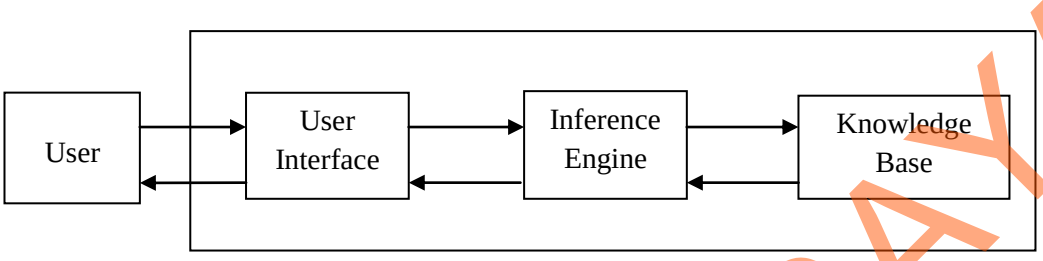
Dalam menyelesaikan permasalahan pada pembuatan tugas akhir ini, terdapat beberapa landasan teori yang mendukung penerapan dari aplikasi sistem pakar untuk mendiagnosis penyakit pada tanaman kopi dengan metode *forward chaining*. Berikut ini adalah penjelasan secara detail tentang teori-teori yang menunjang dalam pembuatan aplikasi sistem pakar ini.

2.1 Konsep Dasar Sistem Pakar

Menurut Irawan (2007), sistem pakar adalah sebuah program komputer yang mencoba meniru atau mensimulasikan pengetahuan (*knowledge*) dan ketrampilan (*skill*) dari seorang pakar pada area tertentu. Pada umumnya pengetahuan sistem pakar berusaha menirukan metodologi dan kinerja dari seorang manusia yang pakar dalam domainnya. Tujuan dari sistem pakar sebenarnya bukan untuk menggantikan peran manusia, tetapi untuk mensubstitusikan pengetahuan manusia ke dalam bentuk sistem sehingga dapat digunakan oleh orang banyak.

Menurut Irawan (2007), keuntungan yang didapat dari sistem pakar adalah tidak terbatas karena dapat digunakan kapan pun juga. Pengetahuannya bersifat konsisten, kecepatan untuk memberikan solusi lebih cepat daripada manusia dan biaya yang dikeluarkan sedikit. Berbeda dengan manusia yang membutuhkan istirahat, pengetahuannya bersifat variabel dan dapat berubah-ubah tergantung situasi. Kecepatan untuk menemukan solusi sifatnya bervariasi dan biaya yang harus dikeluarkan untuk konsultasi biasanya mahal.

Menurut Gonzales (1993), sistem pakar mempunyai 3 bagian utama, yaitu *User Interface*, *Inference Engine* dan *Knowledge Base*. Hubungan ketiga bagian tersebut dapat dinyatakan seperti Gambar 2.1.



Gambar 2.1 Bagian Utama Sistem Pakar

1. User Interface

   User interface adalah perangkat lunak yang menyediakan media komunikasi antara *User* dengan sistem. *User interface* memberikan berbagai fasilitas informasi dan berbagai keterangan yang bertujuan untuk membantu mengarahkan alur penelusuran masalah sampai ditemukan sebuah solusi (Andi, 2003).

2. Inference Engine

   Menurut Andi (2003), *inference engine* adalah bagian dari sistem pakar yang melakukan penalaran dengan menggunakan isi *rules* berdasarkan urutan dan pola tertentu. Selama proses konsultasi antara sistem dengan *user, inference engine* menguji *rules* satu demi satu sampai kondisi *rules* itu benar. Secara umum ada dua metode *inference engine* yang penting dalam sistem pakar, yaitu runut maju (*forward chaining*) dan runut balik (*backward chaining*).

3. Knowledge Base

   Knowledge base merupakan inti program sistem pakar. Pengetahuan ini merupakan representasi pengetahuan dari seorang pakar. Menurut Irawan

(2007), *knowledge base* bisa direpresentasikan dalam berbagai macam bentuk, salah satunya adalah bentuk sistem berbasis aturan (*ruled-based system*). *Knowledge base* tersusun atas fakta yang berupa informasi tentang obyek dan *rules* yang merupakan informasi tentang cara bagaimana membangkitkan fakta baru dari fakta yang telah diketahui.

2.1.1 Ciri-Ciri Sistem Pakar

Ciri-ciri sistem pakar adalah sebagai berikut (Kusrini, 2006):

1. Terbatas pada bidang yang spesifk.
2. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti.
3. Dapat mengemukakan rangkaian alasan yang diberikannya dengan cara yang dapat dipahami.
4. Berdasarkan pada *rules* atau aturan-aturan tertentu.
5. Dirancang untuk dikembangkan secara bertahap.
6. Outputnya bersifat nasihat atau anjuran.
7. Output tergantung dari dialog dengan user.
8. *Knowledge base* dan *inference engine* terpisah.

2.1.2 Keuntungan Dan Kelemahan Sistem Pakar

Menurut Kusrini (2006), ada beberapa keuntungan yang dapat diperoleh dengan mengembangkan sistem pakar antara lain:

1. Membuat seorang yang awam dapat bekerja seperti layaknya seorang pakar.
2. Dapat bekerja dengan informasi yang tidak lengkap atau tidak pasti.
3. Meningkatkan output dan produktivitas.

4. Meningkatkan kualitas.
5. Menyediakan nasihat atau solusi yang konsisten dan dapat mengurangi tingkat kesalahan.
6. Membuat peralatan yang kompleks dan mudah dioperasionalkan karena sistem pakar dapat melatih pekerja yang tidak berpengalaman.
7. Sistem tidak dapat lelah atau bosan.
8. Memungkinkan pemindahan pengetahuan ke lokasi yang jauh serta memperluas jangkauan seorang pakar, dapat diperoleh dan dipakai di mana saja.

Menurut Kusrini (2006), ada beberapa kelemahan yang dapat diperoleh dengan mengembangkan sistem pakar, antara lain:

1. Daya kerja dan produktivitas manusia menjadi berkurang karena semuanya dilakukan secara otomatis oleh sistem.
2. Pengembangan perangkat lunak sistem pakar lebih sulit dibandingkan dengan perangkat lunak konvensional.
3. Biaya pembuatannya mahal, karena seorang pakar membutuhkan pembuat aplikasi untuk membuat sistem pakar yang diinginkannya.

2.1.3 Orang Yang Terlibat Dalam Sistem Pakar

Menurut Kusrini (2006), untuk memahami perancangan sistem pakar, perlu dipahami mengenai siapa saja yang berinteraksi dengan sistem. Orang yang terlibat dalam sistem pakar adalah:

1. Pakar (*Domain Expert*).

   Pakar adalah seseorang yang dapat menyelesaikan masalah yang sedang diusahakan untuk dipecahkan oleh sistem.

2. Pembangun pengetahuan (*Knowledge Engineer*).

   Pembangun pengetahuan adalah seseorang yang menterjemahkan pengetahuan seorang pakar dalam bentuk deklaratif sehingga dapat digunakan oleh sistem pakar.

3. Pemakai (*User*).

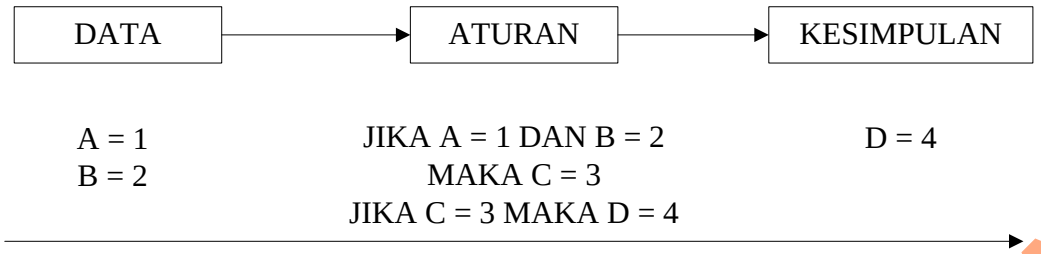
   Pemakai adalah seseorang yang berkonsultasi dengan sistem untuk mendapatkan saran yang disediakan oleh pakar.

4. Pembangun sistem (*System Engineer*)

   Pembangun sistem adalah seseorang yang dapat membuat antarmuka pengguna (*user interface*), merancang bentuk basis pengetahuan (*knowledge base*) secara deklaratif dan mengimplementasikan mesin inferensi (*inference engine*).

2.1.4 Runut Maju (*Forward Chaining*)

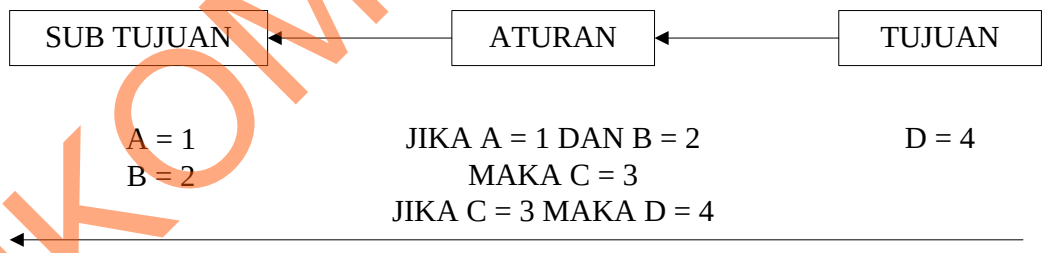
Runut maju (*forward chaining*) berarti menggunakan himpunan aturan kondisi-aksi. Dalam metode ini, data digunakan untuk menentukan aturan mana yang akan dijalankan, kemudian aturan tersebut dijalankan. Mungkin proses menambahkan data ke memori kerja. Proses diulang sampai ditemukan suatu hasil (Kusrini, 2006). Gambar 2.2 menunjukkan bagaimana cara kerja metode inferensi runut maju (*forward chaining*).



Gambar 2.2 Cara Kerja Metode *Forward Chaining* (Kusrini, 2006:36)

2.1.5 Runut Balik (*Backward Chaining*).

Runut balik (*backward chaining*) merupakan metode penalaran kebalikan dari runut maju (*forward chaining*). Dalam runut balik, penalaran di mulai dengan tujuan kemudian merunut balik ke jalur yang mengarah ke tujuan tersebut. Runut balik disebut juga sebagai *goal-drive reasoning* yang merupakan cara yang efisien untuk memecahkan masalah yang dimodelkan sebagai masalah pemilihan terstruktur. Tujuan dari metode ini adalah mengambil pilihan terbaik dari banyak kemungkinan (Kusrini, 2006). Gambar 2.3 menunjukkan bagaimana cara kerja metode *backward chaining*.



Gambar 2.3 Cara Kerja Metode *Backward Chaining* (Kusrini, 2006:36)

2.1.6 Verifikasi

Verifikasi merupakan sekumpulan aktifitas yang memastikan suatu sistem telah berlaku dalam kondisi yang ditetapkan. Verifikasi itu sendiri terdiri dari dua proses, yaitu pertama memeriksa keadaan sistem, kedua memeriksa konsistensi dan kelengkapan dari basis pengetahuan (*knowledge base*). Verifikasi

dijalankan ketika ada perubahan pada *rules*, karena *rules* tersebut sudah ada pada sistem. Tujuan verifikasi adalah untuk memastikan adanya kecocokan antara sistem dengan apa yang sistem kerjakan dan juga memastikan apakah sistem itu terbebas dari *error*. Berikut ini adalah beberapa metode pemeriksaan *rules* dalam suatu basis pengetahuan (Gonzales, 1993).

1. Redundant Rules

   Redundant rules terjadi jika dua *rules* atau lebih mempunyai *premise* dan *conclusion* yang sama.

   Contoh:

   Rule 1: *if the humidity is high and the temperature is hot*
   Then there will be thunderstorms

   Rule 2: *if the temperature is hot and the humidity is high*
   Then there will be thunderstorms

2. Conflicting Rules

   Conflicting rules terjadi jika dua *rules* atau lebih mempunyai *premise* yang sama, tetapi mempunyai *conclusion* yang berlawanan.

   Contoh:

   Rule 1: *if the temperature is hot and the humidity is high*
   Then there will be sunshine

   Rule 2: *if the temperature is hot and the humidity is high*
   Then there will be no sunshine

3. Subsumed Rules

   Subsumed rules terjadi jika *rules* tersebut mempunyai *constraint* yang lebih atau kurang tetapi mempunyai *conclusion* yang sama.

Contoh:

Rule 1: *If the temperature is hot and the humidity is high*

Then there will be thunderstorms

Rule 2: *If the temperature is hot*

Then there will be thunderstorms

4. Circular Rules

Circular rules adalah suatu keadaan dimana terjadinya proses perulangan dari suatu *rule*. Ini dikarenakan suatu *premise* dari salah satu *rule* merupakan *conclusion* dari *rule* yang lain, atau kebalikannya.

Contoh:

Rule 1: *If X and Y are brothers*

Then X and Y have the same parents

Rule 2: *If X and Y have the same parents*

Then X and Y are brothers

5. Unnecessary if Condition

Unnecessary if Condition terjadi jika dua *rules* atau lebih mempunyai *conclusion* yang sama, tetapi salah satu dari *rule* tersebut mempunyai *premise* yang tidak perlu dikondisikan dalam *rule* karena tidak mempunyai pengaruh apapun.

Contoh:

Rule 1: *If the patient has the pink spots and the patient has a fever*

Then the patient has measles

Rule 2: *If the patient has the pink spots and the patient does not have fever*

Then the patient has measles

6. Dead-end Rules

   Dead-end rules adalah suatu *rule* yang *conclusion*-nya tidak diperlukan oleh *rule* lainnya.

   Contoh:

   Rule 1*:* *If the gauge reads empty*

   Then the gas tank

7. Missing Rules

   Missing rules merupakan suatu aturan yang ditandai dengan fakta yang tidak pernah digunakan dalam proses *inference engine*.

8. Unreachable Rules

   Unreachable rules merupakan suatu atauran yang gejalanya tidak akan pernah benar.

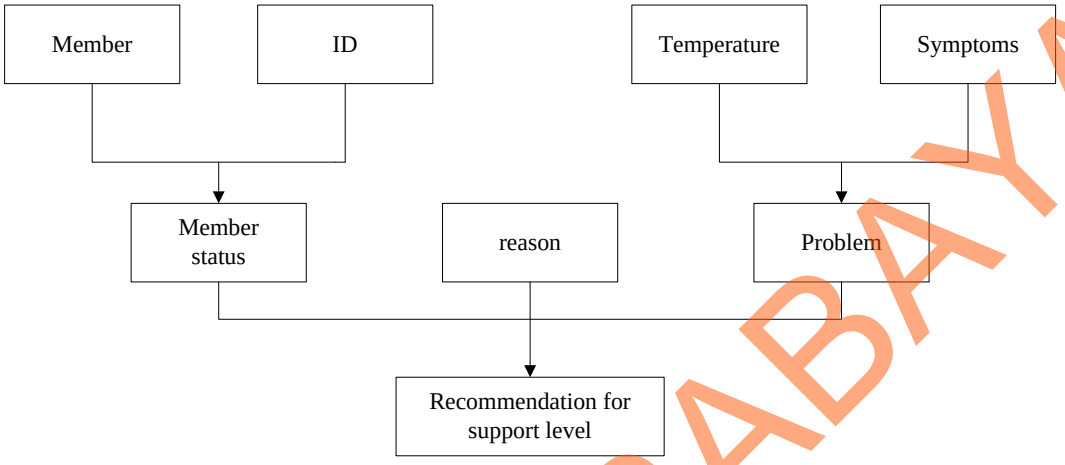
2.1.7 Block Diagram

Langkah awal yang dilakukan dalam menerjemahkan suatu bidang ilmu ke dalam sistem berbasis aturan adalah melalui *block diagram* (diagram blok). *Block diagram* merupakan susunan dari *rules* yang terdapat di dalam sebuah bidang ilmu (Dologite, 1993). Dengan membuat *block diagram* di dalam sistem pakar, maka dapat diketahui urutan kerja sistem dalam mencari keputusan. Contoh dari *block diagram* dapat dilihat pada Gambar 2.4.

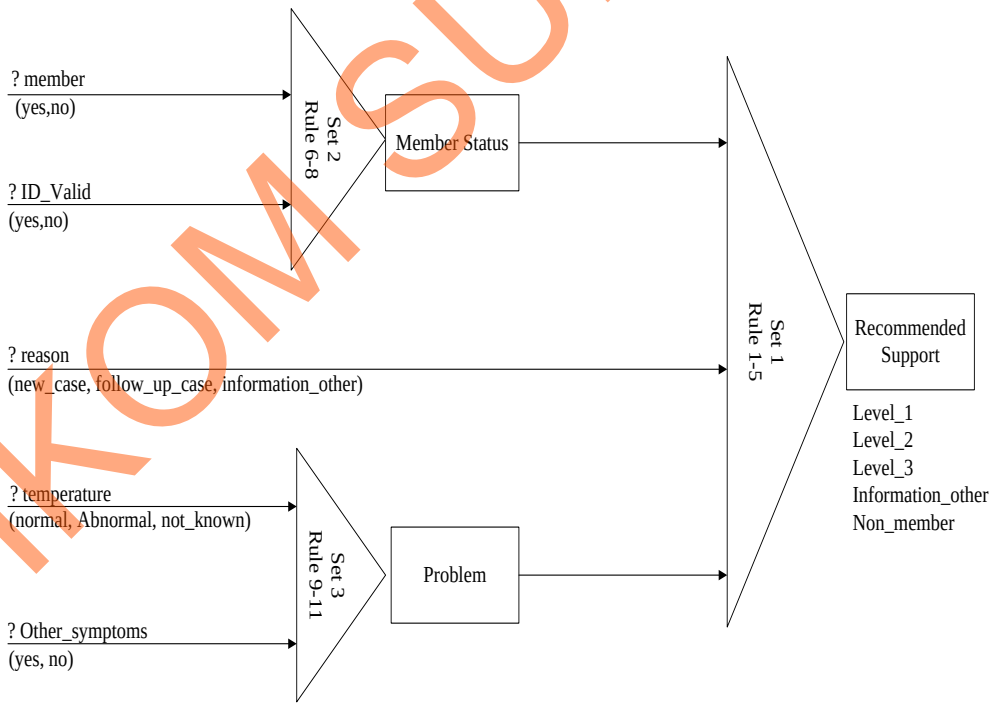
2.1.8 Dependency Diagram

Menurut Dologite (1993), *dependency diagram* adalah suatu relasi yang menunjukan hubungan atau ketergantungan antara inputan jawaban, aturan-aturan

(*rules*), nilai dan rekomendasi yang dibuat oleh *prototype* sistem berbasis pengetahuan. Contoh dari *dependency diagram* dapat dilihat pada Gambar 2.5.



Gambar 2.4 *Block Diagram* (Irawan, 2007:56)



Gambar 2.5 *Dependency Diagram* (Irawan, 2007:57)

2.1.9 Decision Table

Menurut Dologite (1993), *decision table* diperlukan untuk menunjukan hubungan timbal balik antara nilai-nilai pada hasil fase antara atau rekomendasi akhir *knowledge based system* (KBS). Contoh dari pembuatan *decision table* dapat dilihat pada tabel 2.1.

Tabel 2.1 *Decision Table* (Irawan, 2007:57)

Step 1: *Plan*

Kondisi	Member_status (Ok, Not_ok)	2
	Reason (new_case, follow_up, information_other)	3
	Problem (serious, non_serious)	2
Baris	2 x 3 x 2 = 12	

Step 2: *Completed Decision Table*

Rule	Member Status	Reason	Problem	Concluding Recommendation for support level
A1	Ok	New_case	Serious	Level_1
A2	Ok	New_case	Non_serious	Level_2
A3	Ok	Follow_up_case	Serious	Level_1
A4	Ok	Follow_up_case	Non_serious	Level_3
A5	Ok	Information_other	Serious	Information_other
A6	Ok	Information_other	Non_serious	Information_other
A7	Not_ok	New_case	Serious	Non_member
A8	Not_ok	New_case	Non_serious	Non_member
A9	Not_ok	Follow_up_case	Serious	Non_member
A10	Not_ok	Follow_up_case	Non_serious	Non_member
A11	Not_ok	Information_other	Serious	Non_member
A12	Not_ok	Information_other	Non_serious	Non_member

2.1.10 Reduced Decision Table

Menurut Dologite (1993), *reduced decision table* adalah pembuatan tabel yang nilai-nilainya didapat dari mereduksi *decision table*. Setelah didapatkan nilai dari *decision table*, nilai tersebut direduksi untuk mendapatkan nilai dari kondisi terakhir. Contoh dari *reduced decision table* dapat dilihat pada tabel 2.2.

Tabel 2.2 *Reduced Decision Table* (Irawan, 2007:58)

Rule	Member Status	Reason	Problem	Concluding Recommendation for support level
A1	Ok	New_case	Serious	Level_1
A2	Ok	New_case	Non_serious	Level_2
A3	Ok	Follow_up_case	Serious	Level_1
A4	Ok	Follow_up_case	Non_serious	Level_3
A5	Ok	Information_other	-	Information_other
A6	Not_ok	-	-	Non_member

2.2 Kopi (*Coffea spp. L.*)

Tanaman yang termasuk *Genus Coffea* dari *Family Rubiaceae* ini adalah salah satu dari tiga bahan minuman non-alkoholik (kopi, teh dan coklat). Produksi kopi sebagian besar berasal dari Benua Amerika, yaitu Amerika Selatan dan Amerika Tengah. Sejak tahun 1990-an produksi kopi di Benua Afrika Nampak semakin merosot, sedangkan di Benua Asia semakin meningkat. Perkembangan produksi kopi dunia dapat dilihat pada tabel 2.3 (Yahmadi, 2007).

Saat ini produksi kopi dunia masih tetap didominasi oleh Negara Brasil, walaupun selama beberapa tahun pernah mengalami kemunduran akibat terserang penyakit karat daun (*Hemileia vastatrix*) (Yahmadi, 2007).

Tabel 2.3 Perkembangan Produksi Kopi Dunia (Yahmadi, 2007:3)

Periode Tahun	Persentase produksi kopi			
	Amerika selatan	Amerika Tengah	Afrika	Asia
1967-1968	48	16	29	7
1997-1998	43	19	15	23
1999-2000	41	18	16	25
2001-2002	50	15	12	23
2003-2004	47	14	13	26

2.2.1 Perkembangan Kopi di Indonesia

Jenis-jenis kopi komersial yang sekarang diusahakan di Indonesia, yaitu Robusta dan Arabika. Asal kedua jenis kopi ini adalah dari Benua Afrika. Dulu, di Indonesia pada abad 18 dan 19 pernah ditanam jenis kopi Liberika. Namun, semenjak abad 20 kebanyakan kopi yang ditanam di Indonesia adalah jenis Robusta dan Arabika (Yahmadi, 2007).

1. Kopi Arabika

   Jenis kopi yang pertama kali dimasukkan ke Indonesia adalah kopi Arabika (*Coffea arabica*), yaitu pada tahun 1696. Satu abad lebih jenis kopi ini telah membudidaya menjadi tanaman rakyat. Tanaman kopi perkebunan pertama-tama diusahakan di Jawa Tengah (Semarang dan Kedu) pada abad ke-19. Perkebunan kopi di Jawa Timur (Kediri dan Malang) baru dibuka pada akhir abad ke-19 dan di Besuki baru dibuka pada tahun 1890-1990. Selama satu setengah abad kopi arabika merupakan satu-satunya jenis kopi komersial yang ditanam di Indonesia. Perkembangan budidaya kopi jenis ini kemudian mengalami kemunduran hebat, akibat terserang penyakit karat daun (*Hemileia vastatrix*) yang masuk ke Indonesia sejak tahun 1876, sehingga kopi jenis ini

hanya bisa bertahan di daerah-daerah dataran tinggi (1000m dari permukaan laut).

2. Kopi Liberika.

   Kopi Liberika (*Coffea liberica*) dimasukkan ke Indonesia pada tahun 1875, sebagai usaha untuk mengatasi penyakit karat daun. Dengan harapan jenis kopi ini akan lebih kuat dibandingkan dengan jenis kopi Arabika dalam serangan penyakit karat daun. Namun ternyata jenis kopi ini juga mudah diserang penyakit karat daun. Selain itu, jenis ini pada umumnya kurang disukai, karena rasanya terlalu asam. Sampai saat ini jenis ini tidak ditanam lagi dan tinggal sisa-sisanya yang masih ada di beberapa tempat.

3. Kopi Robusta

   Kopi Robusta (*Coffea robusta*) dimasukkan ke Indonesia pada tahun 1900. Kopi jenis ini ternyata tahan terhadap serangan penyakit karat daun. Syarat tumbuh dan pemeliharaanya juga ringan serta produksinya jauh lebih tinggi dibandingkan dengan kopi Arabika. Saat ini kira-kira 90% dari area kopi di Indonesia terdiri atas kopi ini.

2.2.2 Mengenal Tanaman Kopi

Untuk mengenal tanaman kopi, bisa dilihat dari struktur atau bagian-bagian dari tanaman kopi tersebut (Yahmadi, 2007).

1. Akar

   Tanaman kopi berakar tunggang dan perakaran kopi relatif dangkal. Lebih dari 90% akar kopi beradah di dalam tanah yang dalamnya hanya antara 0-30cm. Oleh karena itu, kopi sangat peka terhadap kandungan bahan organik, perlakuan tanah dan juga terhadap saingan rumpai. Bila pertumbuhan akar

tanaman kopi ini terhambat, maka akan mengakibatkan tanaman kopi terlihat kerdil, karena kekurangan air atau kekurangan udara.

2. Batang dan cabang

   Batang dari tanaman kopi ini mulai kelihatan saat tanaman kopi tumbuh dari bijinya dan tumbuh terus sampai menjadi besar. Bentuknya beruas-ruas dan pada tiap ruasnya tumbuh sepasang daun yang berhadapan, yang selanjutnya tumbuh pula cabang yang berbeda-beda. Pada batang tumbuh 2 macam cabang, yaitu:

   a. Cabang yang tubuh tegak lurus atau vertikal. Cabang ini disebut cabang orthotrop atau tunai air (wiwilan).
   b. Cabang yang tumbuhnya ke samping atau horisontal. Cabang ini merupakan tempat tumbuh bunga atau buah. Cabang ini disebut cabang plagiotrop atau cabang buah.

3. Kuncup

   Di daerah ketiak daun yang terletak di atas buku, tumbuh dua macam kuncup atau titik tumbuh, yaitu:

   a. Kuncup primer

      Di setiap ketiak daun terdapat satu mata kuncup yang disebut kuncup legitium. Kuncup inilah yang akan tumbuh menjadi cabang horisontal.

   b. Kuncup reproduksi

      Pada batang ketiak daun tersusun 4–5 kuncup reproduksi. Kuncup ini terletak di bawah kuncup primer dan nantinya akan tumbuh menjadi cabang vertikal.

4. Daun

   Kopi mempunyai bentuk daun bulat telur, ujungnya agak meruncing sampai bulat. Daun kopi tumbuhnya berhadapan dan berpasangan, baik yang tumbuh pada cabang maupun pada batang. Pada cabang, pasangan daun tersebut terletak pada satu bidang. Akan tetapi pada batang dan wiwilan, pasangan daun tersebut tidak terletak pada satu bidang melainkan pada bidang yang berlainan.

5. Bunga dan buah

   a. Bunga

      Bunga kopi terletak pada ketiak daun dari cabang. Pada tiap ketiak akan terdapat 4–6 kuntum bunga yang bertangkai pendek, masing-masing terdiri dari 3–5 bunga. Mahkota bunga berwarna putih dengan jumlah daun mahkota yang berbeda-beda tergantung jenisnya. Kopi umumnya berbunga pada umur 3 tahun. Dari bunga inilah yang akan menjadi bakal buah.

   b. Buah

      Dari bunga sampai menjadi buah yang masak membutuhkan waktu 9 – 12 bulan tergantung jenisnya. Buah kopi yang muda berwarna hijau, setelah tua menjadi kuning dan kalau masak menjadi warna merah. Pada umumnya buah kopi mempunyai dua butir biji, biji tersebut mempunyai dua bidang, bidang yang datar atau perut dan bidang yang cembung atau punggung.

2.2.3 Penyakit Tanaman Kopi

Pertumbuhan dan perkembangan tanaman dari sejak benih, pembibitan, perkembangan, hingga pada gudang penyimpanan selalu tidak luput dari gangguan hama, penyakit, gulma dan faktor-faktor lingkungan. Gangguan atau penyakit tanaman terhadap kehidupan manusia begitu besar, namun masih bayak orang yang belum sadar akan kerugian tersebut. Pengetahuan tentang hama, penyakit, gulma dan cara pengendaliannya masih belum banyak diketahui oleh para petani dan pekebun. Oleh karena itu, pengetahuan tentang hal-hal tersebut sangat penting untuk diketahui oleh para petani dan pekebun (Tim Karya Tani Mandiri, 2010).

Adapun penyakit-penyakit yang sering menyerang tanaman kopi di Indonesia adalah:

1. Penyakit cendawan akar coklat

   Penyebabnya: Jamur *Phellinus noxius*

   Gejalanya:

   a. Daunnya menguning, layu dan gugur.
   b. Pada akar tertutup kerak yang terdiri atas butir-butir tanah yang melekat sangat kuat, sehingga tidak dapat terlepas.
   c. Di antara butir-butir tanah tersebut tampak jaringan jamur yang berwarna coklat tua sampai coklat kehitaman.
   d. Akar menjadi busuk, kering dan lunak.

   Pengendaliannya:

   a. Dilakukan pembongkaran pada tanaman yang sakit, sisa-sisa akar diambil dan dibakar.

b. Membuat saluran isolasi di tempat yang terinfeksi.

c. Melakukan peremajaan, dengan membongkar tanaman yang sudah tua hingga tidak dijumpai tunggul pohon-pohon tua.

2. Penyakit akar putih

Penyebabnya: Cendawan *Rigidoporus microporus*

Gejalanya:

a. Daun menguning kemudian gugur.

b. Munculnya bunga dan buah meskipun umurnya belum cukup.

c. Akar membusuk dan diselubungi selaput *miselium* jamur mirip jala putih.

d. Infeksi berat membuat pohon roboh karena akar membusuk.

Pengendaliannya:

a. Lakukan sanitasi lahan sampai benar-benar bersih dari serasah dan tonggak ataupun akar yang tersisa dalam tanah.

b. Tebang dan cabut akar pohon yang terserang.

c. Sebelum ditanami, lahan disanitasi dengan fungisida berbahan aktif *triadimefon* seperti, *Bayleton* 250 EC, *Anvil* 50 SC yang mengandung *heksakonazol* dan *Calixin* 750 EC dengan bahan aktif *tridemorf*.

d. Buat parit antar pohon untuk mengisolasi penyakit.

3. Penyakit busuk akar

Penyebabnya: Cendawan *Armillaria tabascens*

Gejalanya:

a. Daun menguning dan layu.

b. Muncul retakan pada batang yang menandakan kekurangan air.

c. Akar tampak mengeluarkan cairan kental coklat kekuningan.

d. Kalau kelembapan tinggi tampak tudung jamur bergerombol di dasar tanaman.

Pengendaliannya:

a. Lakukan sanitasi lahan dengan menyingkirkan tonggak dan sisa kayu tebangan yang bisa menjadi media tumbuh spora.
b. Kalau ada tanaman yang baru mati akibat terserang penyakit ini, segera singkirkan dan ganti tanah di sekitarnya dengan tanah dari tempat lain.
c. Pencegahan fisik dengan menyingkirkan cendawan yang tampak dan memotong stolon yang menjalar di tanah.
d. Pencegahan penjalaran secara fisik bisa dilakukan dengan menanam pelat sampai kedalaman 0,5m dalam tanah antara pohon yang terkena dengan pohon yang sehat.

4. Penyakit jamur upas

Penyebabnya: Jamur *Upasia salmonicolor*

Gejalanya:

a. Muncul bercak putih pada batang yang tidak terkena sinar matahari.
b. Pada tahap awal atau tahap sarang laba-laba, muncul bercak berupa lapisan miselia tipis, berbentuk jala berwarna putih perak.
c. Pada tahap bongkol, bercaknya berupa gambaran miselia berwarna putih biasanya dibentuk pada celah–celah batang.
d. Pada tahap kortisium, bercaknya berupa lapisan kerak berwarna merah jambu biasanya dibentuk di bawah cabang yang agak ternaung.
e. Pada tahap nekator, bercaknya berupa bintil–bintil kecil berwarna oranye kemerahan, biasanya dibentuk di bawah cabang yang tidak terlindungi.

f. Kulit batang menjadi mati.

Pengendaliannya:

a. Mengurangi kelembapan kebun dengan memangkas pohon pelindung atau ranting-ranting kopi yang tidak produktif.
b. Membersihkan sumber infeksi yang ada di sekitar tanaman, misalnya tanaman pupuk hijau yang sakit.
c. Penggunaan fungisida, dengan cara mengoles fungisida pada batang atau cabang yang terserang jamur.

5. Penyakit karat daun

Penyebabnya: Jamur *Hemileia vastatrix*

Gejalanya:

a. Pada sisi bawah daun terdapat bercak-bercak berwarna kuning muda kemudian berubah menjadi kuning tua.
b. Pada bercak-bercak tersebut terdapat tepung berwarna jingga cerah yang terdiri atas jamur karat.
c. Bercak yang tua berwarna coklat tua sampai hitam mengering.
d. Daun gugur.
e. Pohonnya menjadi gundul.

Pengendaliannya:

a. Menggunakan varietas kopi yang tahan.
b. Menggunakan mikrobia yang bersifat berlawanan, yaitu bakteri *Bacillus thuringiensis* dan jamur *Verticillium hemileiae*.

c. Penggunaan fungisida, misalnya *oksicholorida* tembaga dengan giliran penyemprotan 3 minggu sekali. Penyemprotan dimulai menjelang datangnya masa hujan lebat.

6. Penyakit bercak daun

   Penyebabnya: Jamur *Cercospora coffeicola*

   Gejalanya:

   a. Pada daun yang sakit timbul bercak berwarna kuning.
   b. Pada buahnya timbul bercak berwarna coklat, biasanya pada sisi yang banyak terkena sinar matahari.
   c. Pembusukan pada bagian buah yang terkena bercak dapat sampai ke biji kopi, sehingga dapat menurunkan kualitas.

   Pengendaliannya:

   a. Potong dan musnakan daun yang terserang.
   b. Menyemprotkan fungisida, misalnya *Bavistin* 50 WP, *Dithane* M 45 80 WP, *Delsene MX* 2000 atau *Amistartop* 325 SC.
   c. Pengurangan kelembapan kebun melalui pemangkasan dan pengendalian gulma.

7. Penyakit bercak daun alga

   Penyebabnya: Alga *Cephaleuros virescens*

   Gejalanya:

   a. Pada permukaan atas daun muncul bintik merah kecoklatan berbentuk cakram yang lama-kelamaan melebar dan tampak hangus.
   b. Bintik-bintik ini juga bisa terlihat di ranting dan batang pohon.

Pengendaliannya:

a. Batang atau cabang yang terinfeksi biasanya ternaungi, makanya pangkas atau hilangkan naungannya.
b. Perbaiki drainase agar kelembapan berkurang.
c. Cegah tanaman menjadi stress dengan mencukupi kebutuhan air dan pupuk.
d. Pangkas secara teratur agar sinar matahari mengenai semua bagian tanaman.
e. Menyemprotkan pestisida bebahan aktif tembaga dan senyawanya, misalnya *Cuprative OB* 21, *Kasumin* 575 WP atau *Shell Copper* yang berbahan aktif tembaga *oksiklorida*.

2.3 Visual Studio .NET 2005

Visual Studio .NET 2005 merupakan salah satu produk pengembangan aplikasi yang diproduksi oleh Microsoft. Visual Studio .NET 2005 dapat digunakan untuk pengembangan aplikasi web ASP .NET, XML *Web Service*, aplikasi *desktop* dan juga aplikasi *mobile*. Dalam Visual Studio .NET 2005 terdapat beberapa bahasa pemrograman yang dapat dipilih untuk pengembangan aplikasi, yaitu Visual Basic, Visual C#, Visual C++ dan Visual J# (Mangkulo, 2005).

Pada bahasa pemrograman Visual Basic .NET 2005, banyak sekali fasilitas w*izard* yang disediakan oleh Visual Studio .NET 2005 untuk memudahkan para pengembang aplikasi. Dengan fasilitas ini, pengembangan aplikasi dapat dilakukan dengan cepat. Fasilitas yang disediakan Visual Studio .NET 2005 bagi setiap bahasa pemrogramannya adalah IDE (*Interface*

Development Environment). Fasilitas ini menyediakan *tool* untuk mendesain, menjalankan dan mencari kesalahan program dari aplikasi yang dibuat (Mangkulo, 2005).

2.4 SQL Server 2005

SQL Server merupakan salah satu *database engine* terpopular dan terbaik saat ini yang dikeluarkan oleh Microsoft. Kompatibilitas SQL Server telah diakui berbagai pihak dan mampu mendukung berbagai macam bahasa pemrograman berbasis *windows* seperti Visual Basic .NET. SQL server 2005 mempunyai berbagai fasilitas yang memudahkan seorang *database administrator* dalam membuat dan mendesain sebuah basis data atau *database* (Mangkulo, 2005)

Manfaat dari pembuatan basis data adalah untuk mempermudah penciptaan struktur data. Selain itu, basis data dapat digunakan untuk sejumlah program aplikasi yang berlainan sehingga dapat meningkatkan produktivitas *programmer*.

2.5 Test Case

Variasi metode desain *test case* untuk *software* saat ini telah berkembang. Metode-metode ini menyediakan pendekatan sematik terhadap *testing*, menyediakan mekanisme yang dapat membantu memastikan kelengkapan dari *testing* dan juga menyediakan kemungkinan tertinggi untuk mendapatkan *error* pada *software* (Romeo, 2003).

Test case merupakan suatu tes yang dilakukan berdasar pada suatu inisialisasi, masukan, kondisi ataupun hasil yang telah ditentukan sebelumnya. Adapun kegunaan dari *test case* ini adalah sebagai berikut (Romeo, 2003):

1. Untuk melakukan *testing* kesesuaian suatu komponen terhadap spesifikasi produk. *Test case* yang digunakan untuk *testing* ini adalah *black box testing*.
2. Untuk melakukan *testing* kesesuaian suatu komponen terhadap desain. *Test case* yang digunakan untuk *testing* ini adalah *white box testing*.

2.5.1 *White Box Testing*

White box testing yang disebut juga *glass box testing* atau *clear box testing*, merupakan suatu metode desain *test case* yang menggunakan strukur kendali dari desain prosedural. *White box testing* diasosiasikan dengan pengukuran cakupan tes (*test coverage metrics*), yang mengukur persentase jalur dari tipe yang dipilih untuk dieksekusi oleh *test cases* (Romeo, 2003).

Kesalahan-kesalahan yang dapat ditemukan dengan menggunakan *white box testing* adalah (Romeo, 2003):

1. Kesalahan logika dan asumsi yang tidak benar kebanyakan dilakukan ketika *coding* untuk "kasus tertentu". Dibutuhkan kepastian bahwa eksekusi jalur ini telah dites.
2. Asumsi bahwa adanya kemungkinan terhadap eksekusi jalur yang tidak benar.
3. Kesalahan penulisan yang acak, seperti berada pada jalur logika yang membingungkan pada jalur normal.

2.5.2 *Black Box Testing*

Black box testing merupakan *testing* yang dilakukan tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites. *Black box testing* juga disebut sebagai *behavioral testing, specification-based testing, input/output*

testing atau *functional testing. Black box testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*. Kategori *error* yang akan diketahui dengan menggunakan *black box testing* ini adalah (Romeo, 2003):

1. Fungsi yang hilang atau tidak benar.
2. *Error* dari *interface.*
3. *Error* dari struktur data atau akses *external database*.
4. *Error* dari kinerja atau tingkah laku sistem.
5. *Error* dari inisialisasi dan terminasi.