BAB II

LANDASAN TEORI

2.1 Teknologi informasi

Menurut Sawyer (2011,p4) Teknologi informasi (TI) adalah istilah umum yang menggambarkan setiap teknologi yang membantu untuk menghasilkan, memanipulasi, menyimpan, berkomunikasi, dan / atau menyebarkan informasi.

2.2 Internet

Menurut Sawyer (2011,p18) internet adalah jaringan komputer di seluruh dunia yang menghubungkan ratusan dari ribuan jaringan yang lebih kecil. Jaringan ini menghubungkan pendidikan, komersial, nirlaba, dan militer entitas, serta individu.

2.3 WWW

Menurut Sawyer (2011,p18) WWW adalah Web merupakan sistem komputer yang saling berhubungan dengan internet (disebut server) yang mendukung dokumen khusus diformat dalam bentuk multimedia.

2.4 HTML

Menurut Sawyer (2011,p68) Hypertext markup language (HTML) adalah set instruksi khusus (disebut "tag" atau "markup") yang digunakan untuk

menentukan struktur dokumen, format, dan link ke multimedia lainnya pada dokumen di web.

2.5 URL

Menurut Sawyer(2011,p65) URL adalah string karakter yang menunjuk ke sebuah bagian tertentu informasi di mana saja di web. Dengan kata lain, URL adalah alamat unik dari website.

2.6 HTTP

Menurut Sawyer (2011,p66) Hypertext Transfer Protocol (HTTP) adalah aturan-aturan komunikasi yang memungkinkan browser untuk terhubung dengan web server.

2.7 Web browser

Menurut Sawyer (2011,p64) Web browser adalah sebuah software yang memungkinkan anda untuk menemukan dan mengakses berbagai bagian web.

2.8 Web portal

Menurut Sawyer (2011,p72) web portal atau biasa disingkat portal, adalah jenis gateway website yang berfungsi sebagai "*anchor site*" yang menawarkan jasa atau layanan seperti online shopping malls, email support, komunitas forum, berita saat ini dan cuaca, harga saham. Informasi wisata, dan link ke subjek popular yang lain.

2.9 Web server

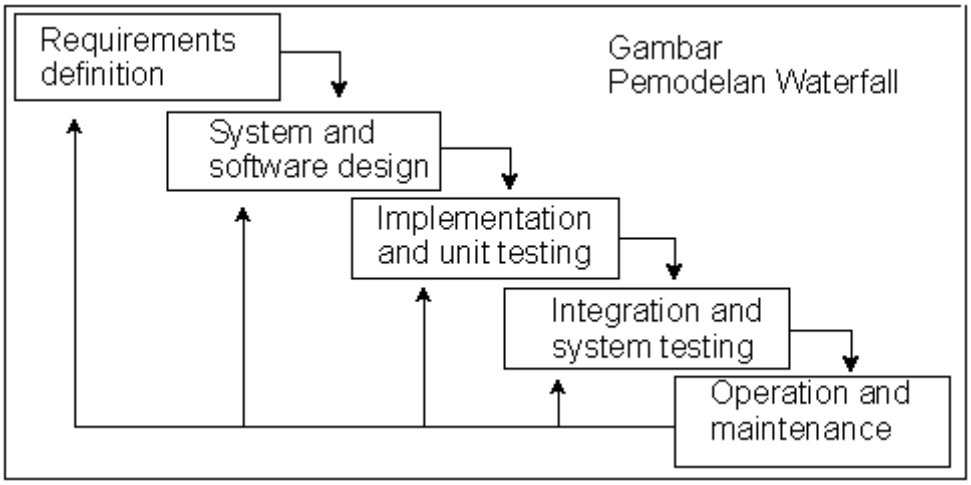
Menurut Sawyer (2011,p318) web server berisi web pages yang dapat kita lihat melalui browser.

2.10 Software Engineering

Menurut Sommerville (2007,p17) Software Engineering adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak. Software engineers harus mengadopsi pendekatan sistematik dan terorganisir untuk pekerjaan yang akan mereka lakukan dengan menggunakan alat dan teknik yang tepat, tergantung dari masalah yang akan di pecahkan, kendala pengembangan, dan sumber daya yang tersedia.

2.11 Waterfall Process Model

Menurut Sommerville (2007,p65) Model ini sering disebut dengan "*Software Life Cycle*" atau model *waterfall*. Model ini melakukan pendekatan secara sistematis dan urut mulai dari *requirement* lalu menuju ke tahap *design*, *coding & unit testing*, *integration & system testing* dan *maintenance*. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Sebagai contoh tahap *design* harus menunggu selesainya tahap sebelumnya yaitu tahap *requirement*. Secara umum tahapan pada model *waterfall* dapat dilihat pada gambar berikut :



Gambar 2.1 Waterfall Model

(Software Engineering 8th edition, p66)

Gambar 2.1 adalah tahapan umum dari model proses ini. Berikut adalah penjelasan dari tahap-tahap yang dilakukan menurut Sommerville (2007,p67):

- **Requirement Analisys and Definition**

  Permodelan ini diawali dengan mencari kebutuhan dari keseluruhan sistem yang akan diaplikasikan ke dalam bentuk *software*. Hal ini sangat penting, mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware*, *database*, dsb.

- **System and Software Design**

  Proses pencarian kebutuhan diintensifkan dan difokuskan pada *software*. Untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang domain informasi dari software, misalnya fungsi yang dibutuhkan, user interface, dsb. Desain *software* melibatkan

identifikasi dan menggambarkan dasar abstraksi *software* sistem dan hubungan mereka.

- **Implementation and Unit Testing**

  Dalam tahap ini,*software design* di realisasikan menjadi satu set program atau program unit. Unit testing bertugas memverifikasi setiap unit mencapai sepsifikasi yang diinginkan.

- **Integration and system testing**

  Mengintegrasikan program yang satu dengan yang lain dan memastikan sistem yang sudah jadi agar sama dengan kebutuhan yang diinginkan oleh *customer*. Setelah testing selesai dilakukan, maka *software* sistem diberikan kepada *customer*.

- **Operation and Maintenance**

  Pemeliharaan suatu software diperlukan, termasuk di dalamnya adalah pengembangan, karena software yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada errors kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada software tersebut. Pengembangan diperlukan ketika adanya perubahan dari eksternal perusahaan seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

2.12 Interaksi Manusia dan Komputer

Menurut ACM SIGHCI (1992,p5) Interaksi Manusia dan Komputer atau *Human-Computer Interaction* (HCI) adalah disiplin ilmu yang berhubungan

dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang berhubungan dengannya.

2.13 Delapan Aturan Emas (Eight Golden Rules)

Menurut Shneiderman (2005,p74) Delapan aturan ini disebut dengan *Eight Golden Rules of Interface Design*, yaitu:

a. Konsistensi

   Konsistensi dilakukan pada urutan tindakan, perintah, dan istilah yang digunakan pada prompt, menu, serta layar bantuan.

b. Memungkinkan pengguna untuk menggunakan *shortcut*

   Ada kebutuhan dari pengguna yang sudah ahli untuk meningkatkan kecepatan interaksi, sehingga diperlukan singkatan, tombol fungsi, perintah tersembunyi, dan fasilitas makro.

c. Memberikan umpan balik yang informative

   Untuk setiap tindakan operator, sebaiknya disertakan suatu sistem umpan balik. Untuk tindakan yang sering dilakukan dan tidak terlalu penting, dapat diberikan umpan balik yang sederhana. Tetapi ketika tindakan merupakan hal yang penting, maka umpan balik sebaiknya lebih substansial. Misalnya

muncul suatu suara ketika salah menekan tombol pada waktu input data atau muncul pesan kesalahannya.

d. Merancang dialog untuk menghasilkan suatu penutupan

Urutan tindakan sebaiknya diorganisir dalam suatu kelompok dengan bagian awal, tengah, dan akhir. Umpan balik yang informatif akan meberikan indikasi bahwa cara yang dilakukan sudah benar dan dapat mempersiapkan kelompok tindakan berikutnya.

e. Memberikan penanganan kesalahan yang sederhana

Sedapat mungkin sistem dirancang sehingga pengguna tidak dapat melakukan kesalahan fatal. Jika kesalahan terjadi, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan mekanisme yang sedehana dan mudah dipahami untuk penanganan kesalahan.

f. Mudah kembali ke tindakan sebelumnya

Hal ini dapat mengurangi kekuatiran pengguna karena pengguna mengetahui kesalahan yang dilakukan dapat dibatalkan; sehingga pengguna tidak takut untuk mengekplorasi pilihan-pilihan lain yang belum biasa digunakan.

g. Mendukung tempat pengendali internal (*internal locus of control*)

Pengguna ingin menjadi pengontrol sistem dan sistem akan merespon tindakan yang dilakukan pengguna daripada pengguna merasa bahwa sistem

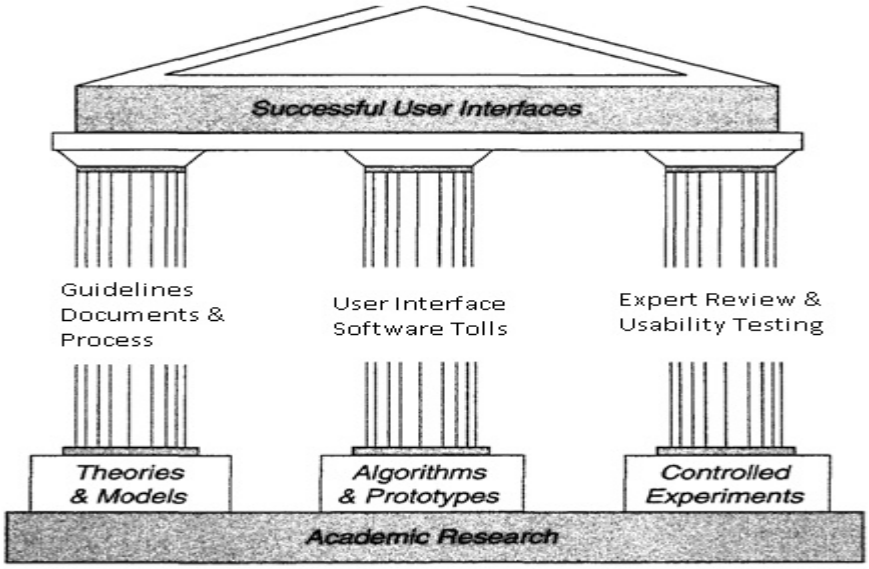
mengontrol pengguna. Sebaiknya sistem dirancang sedemikan rupa sehingga pengguna menjadi inisiator daripada responden.

h. Mengurangi beban ingatan jangka pendek

Keterbatasan ingatan manusia membutuhkan tampilan yang sederhana atau banyak tampilan halaman yang sebaiknya disatukan, serta diberikan cukup waktu pelatihan untuk kode, mnemonic, dan urutan tindakan.

2.14 3 Pilar Perancangan Interface

Menurut Shneiderman (2005,p114) 3 Pilar Perancangan Interface terbagi sebagai berikut:



Gambar 2.2 3 Pilar Perancangan Interface

(Designing the User Interface,p114)

a. Dokumen Pedoman dan Proses

Setiap proyek mempunyai kebutuhan berbeda, tetapi pedoman harus dibuat untuk hal-hal berikut ini:

- Kata-kata, Ikon, dan Grafik

  Terminologi, singkatan, fonts, ikon, grafik, warna, dll.
- Layout layar

  Menu, formulir, dialog box, umpan balik, pesan kesalahan, format pemasukan dan tampilan data dll.
- Perangkat input dan output

  Keyboard, piranti penunjuk, voice, dll.
- Urutan aksi

  Manipulasi Langsung, sintaks bahasa perintah, shortcut, dll
- Pelatihan

  Online help, Tutorial, manual, dll

b. User Interface Software Tools

Pada dasarnya ada suatu kesulitan saat membuat sistem desain interaktif *user* dan *customer* tidak memiliki bayangan seperti apakah jadinya sistem yang telah dibuatnya. Oleh karena itu *prototype* dapat digunakan untuk mengetahui akan jadi seperti apa sistem yang dibuat. *Prototype* dapat dibuat dengan gambar sederhana atau word processing tools, dan design gambar dapat dibuat dengan menggunakan flash dll.

c. Ulasan Pakar dan Uji Usability

- Ulasan Pakar (*Expert Review*)

  Ulasan pakar yang cukup formal telah terbukti efektif. Ulasan pakar dapat dilakukan di awal atau di akhir fase perancangan, dan keluarannya berupa laporan formal dengan masalah yang ditemui atau rekomendasi perubahan.

  Metode ulasan pakar:

  - Evaluasi heuristik
  - Ulasan kesesuaian dengan pedoman (*guidelines review*)
  - Pemeriksaan konsistensi
  - Penelusuran kognitif
  - Pemeriksaan *usability* formal

  Pakar yang berbeda cenderung menemukan masalah yang berbeda, maka 3-5 pakar dapat sangat produktif sebagai uji *usability* pelengkap.

- Uji dan Laboratorium Usability

  Uji *usability* (*usability test*) memberikan konfirmasi kemajuan yang mendukung dan rekomendasi perubahan yang spesifik. Uji *usability* tidak hanya mempercepat proses, tetapi juga menghasilkan penghematan biaya yang dramatik.

2.15 Object Oriented Programming(OOP)

Menurut Grady (2007,p41) *Object Oriented Programming* adalah metode implementasi dimana program diatur sebagai koleksi sebuah objek kooperatif, yang masing – masingnya mewakili *instance* suatu kelas.

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama :

- Encapsulation
- Inheritance
- Polymorphism

a. Encapsulation

Encapsulation merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu 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.

b. Inheritance

Inheritance adalah teknik yang menyatakan bahwa anak(*subclass*) dari objek akan mewarisi data/atribut dan metode dari induknya(*superclass*) langsung. Atribut dan metode dari objek induk diturunkan kepada anak objek, demikian seterusnya.

Inheritance mempunyai arti bahwa atribut dan operasi yang dimiliki bersama di antara kelas yang mempunyai hubungan secara hirarki.

Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi subkelas. Setiap subkelas mempunyai hubungan atau mewarisi semua sifat yang dimiliki oleh kelas induknya, dan ditambah dengan sifat unik yang dimilikinya. Kelas objek dapat didefinisikan atribut dan service dari kelas objek lainnya.

Inheritance menggambarkan generalisasi sebuah kelas.

c. Polymorphism

- Polymorphism yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.
- Polymorphism mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.

2.16 Unified Model Language

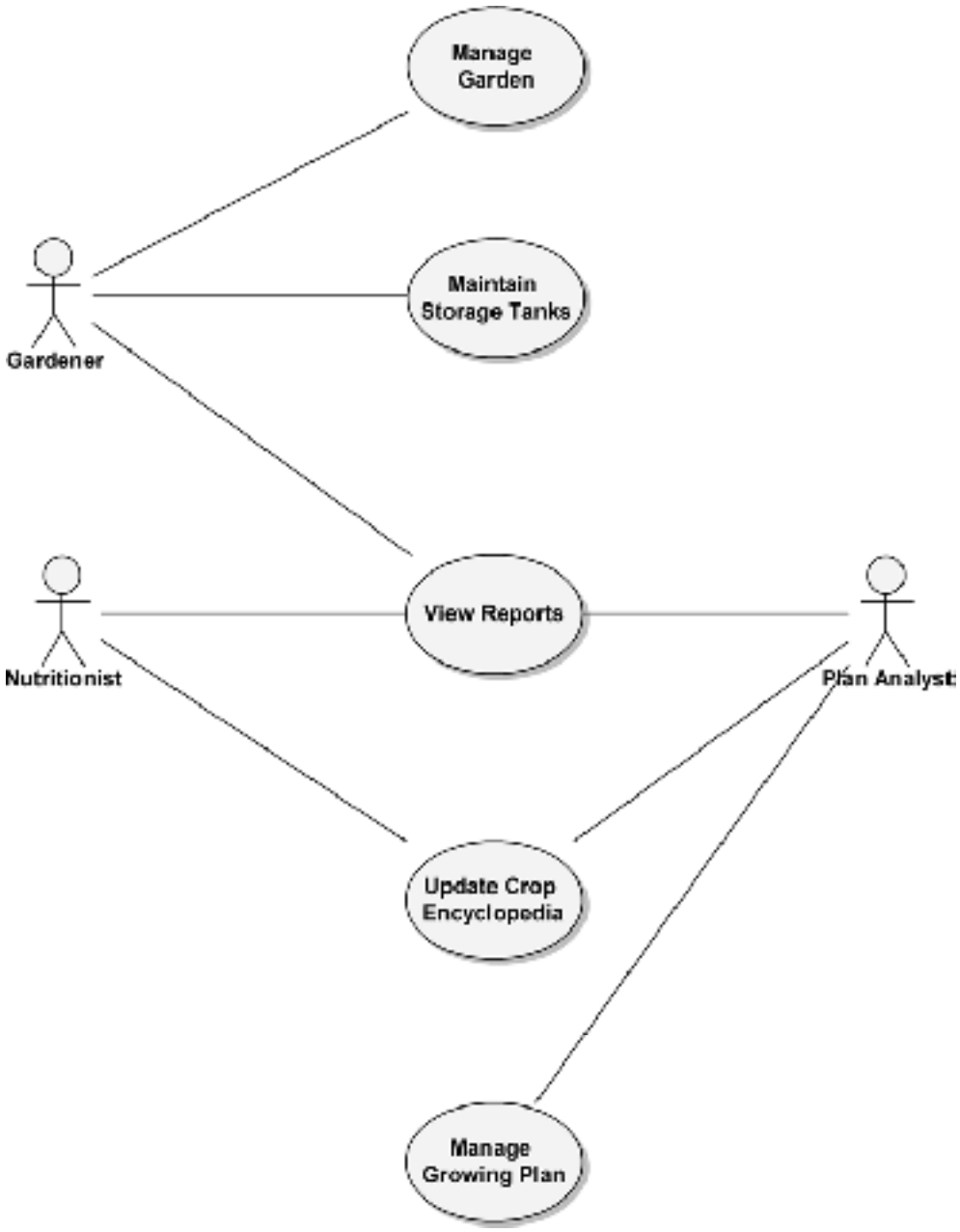
Menurut Grady (2007,p148) *Unified Modeling Language* (UML) adalah sebuah bahasa utama pemodelan yang digunakan untuk menganalisa, menentukan, mendesain sistem suatu perangkat lunak.

Contoh – contoh UML seperti :

- Use Case Diagram
- Class Diagram
- Activity Diagram

a. Use Case Diagram

Menurut Grady (2007,p177) Use Case diagram adalah cara spesifik menggunakan sistem dengan menggunakan beberapa bagian dari fungsi tersebut. Use Case adalah suatu pola atau gambaran yang menunjukan kelakukan atau kebiasaan sistem. Setiap Use Case adalah suatu urut-urutan (*sequence*) transaksi yang saling berhubungan dan dilakukan oleh sebuah actor dan sistem dalam bentuk sebuah dialog. Use Case Diagram dibuat untuk memvisualisasikan/ menggambarkan hubungan antara Actor dan Use Case. Use Case diagram mempresentasikan kegunaan atau fungsi-fungsi sistem dari perspektif pengguna.



Gambar 2.3 Use Case Diagram

(Object Oriented Analysis & Design With Applications 3rd Edition,

p178)

Bagian – bagian use case diagram :

- Use Case

  Gambar use cases menggunakan lingkaran berbentuk bulat telur (oval)

  Beri nama oval tersebut dengan kata kerja (*verb*) yang menggambarkan fungsi-fungsi sistem

Gambar 2.4 Use Case

(Object Oriented Analysis & Design With Applications 3rd Edition, p177)

- Actors

  Actors adalah para pengguna (users) dari sebuah sistem. Kadangkala sebuah sistem adalah merupakan actors bagi sistem yang lain, beri nama *actors* sistem tersebut dengan streotipe (bentuk klise/tiruan) *actor*. *Actor* adalah seseorang atau sesuatu yang harus berinteraksi dengan sistem atau sistem yang dibangun/dikembangkan.

Gambar 2.5 Actor

(Object Oriented Analysis & Design With Applications 3rd Edition, p176)

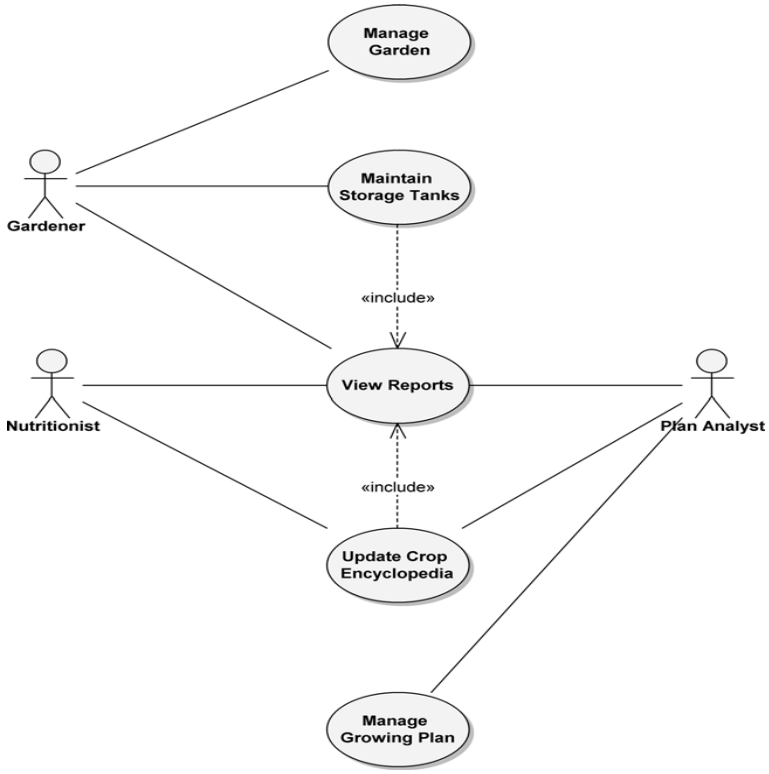
- Relationship

  Relationship adalah sebuah tanda yang berbentuk garis yang menghubungkan actor dan use case. Ataupun use case dengan use case lainnya.

  Relationship dibagi menjadi tiga yaitu:

  - Include

    Include Relationship adalah hubungan antara use case yang dimana jika use case yang satu dijalankan, yang satunya lagi juga harus dijalankan.

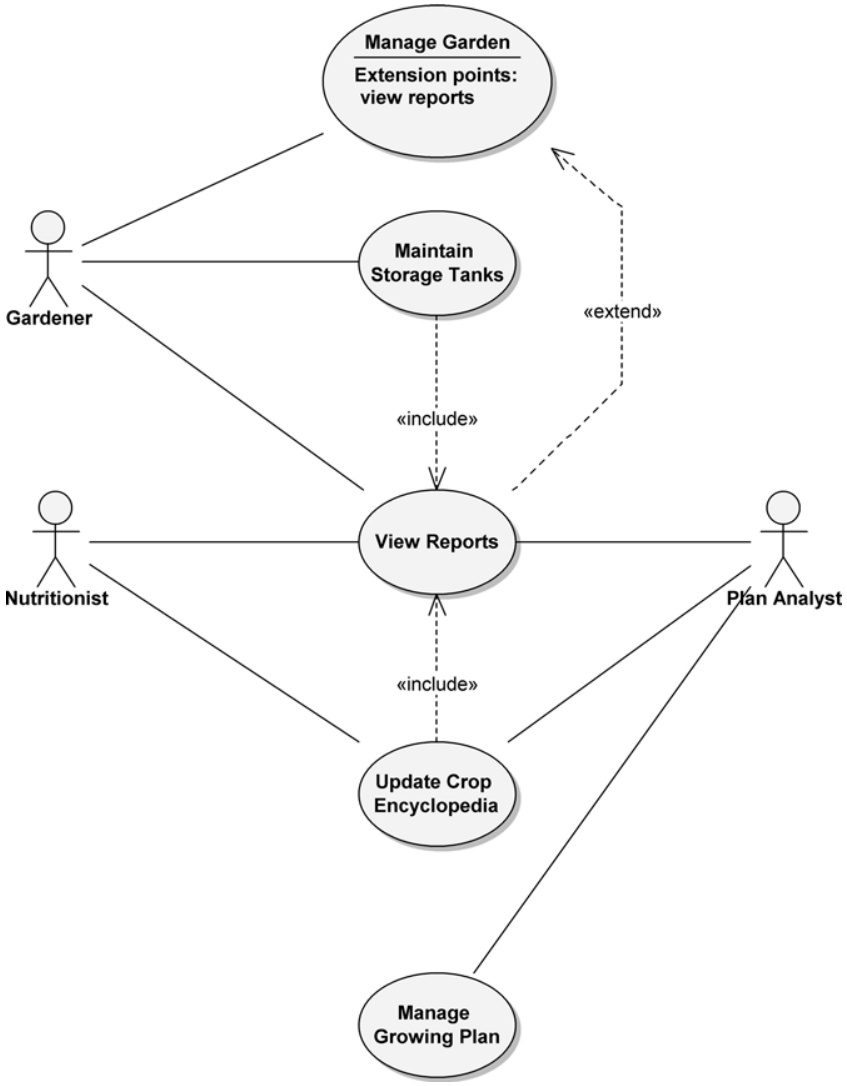


Gambar 2.6 Include Relationship

(Object Oriented Analysis & Design With Applications 3rd Edition, p181)

- Extend

  Extend Relationship adalah hubungan antara use case dimana use case tersebut memerlukan use case lain sebagai syarat agar dapat melakukan use case selanjutnya.

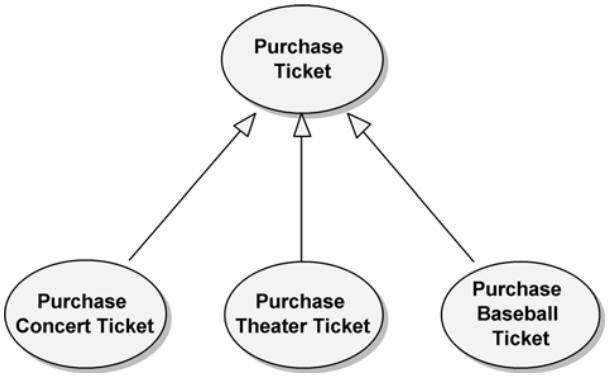


Gambar 2.7 Extend Relationship

(Object Oriented Analysis & Design With Applications 3rd Edition, p183)

- Generalization

  Generalization Relationship adalah hubungan antara use case yang bisa mempunyai sifat yang sama dengan use casee lainnya (seperti *inheritance superclass* dan *subclass*).



Gambar 2.8 Generalization Relationship

(Object Oriented Analysis & Design With Applications 3rd Edition, p185)

b. Class Diagram

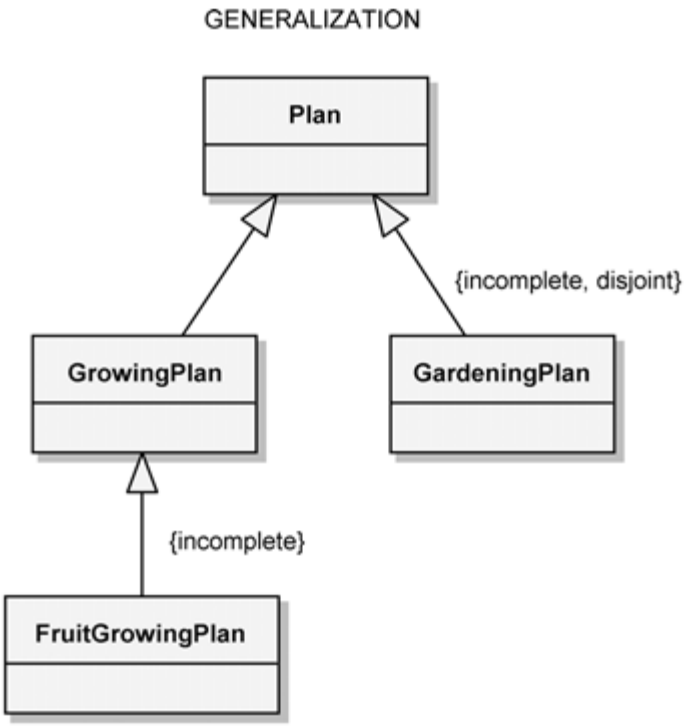
Menurut Grady (2007,p192) Class diagram digunakan untuk melihat eksistensi suatu *class* dan relasinya dari sudut pandang logika suatu sistem.

Relasi di dalam class diagram :

- Generalisasi dan Inheritance

  Diperlukan untuk memperlihatkan hubungan pewarisan (inheritance) antar unsur dalam diagram kelas. Pewarisan memungkinkan suatu

kelas mewarisi semua atribut, operasi ,relasi, dari kelas yang berada dalam hirarki pewarisannya.

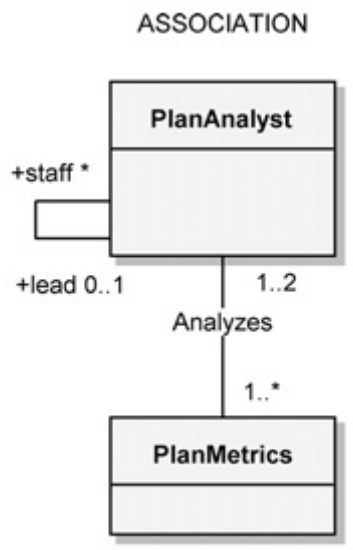


Gambar 2.9 Generalisasi

(Object Oriented Analysis & Design With Applications 3rd Edition, p195)

- Asosiasi

  Hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui ekstensi class lain.

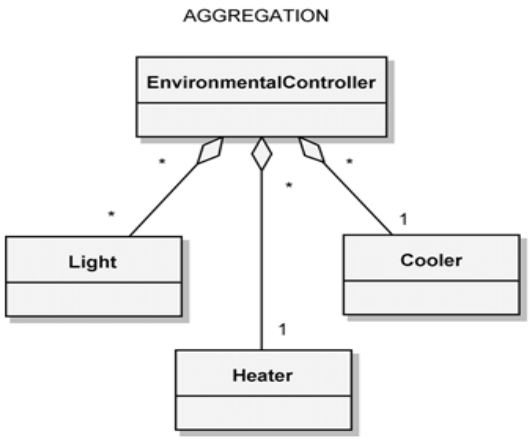


Gambar 2.10 Asosiasi

(Object Oriented Analysis & Design With Applications 3rd Edition, p195)

- Agregat

  hubungan antar-class di mana class yang satu (part class) adalah bagian dari class lainnya (whole class).

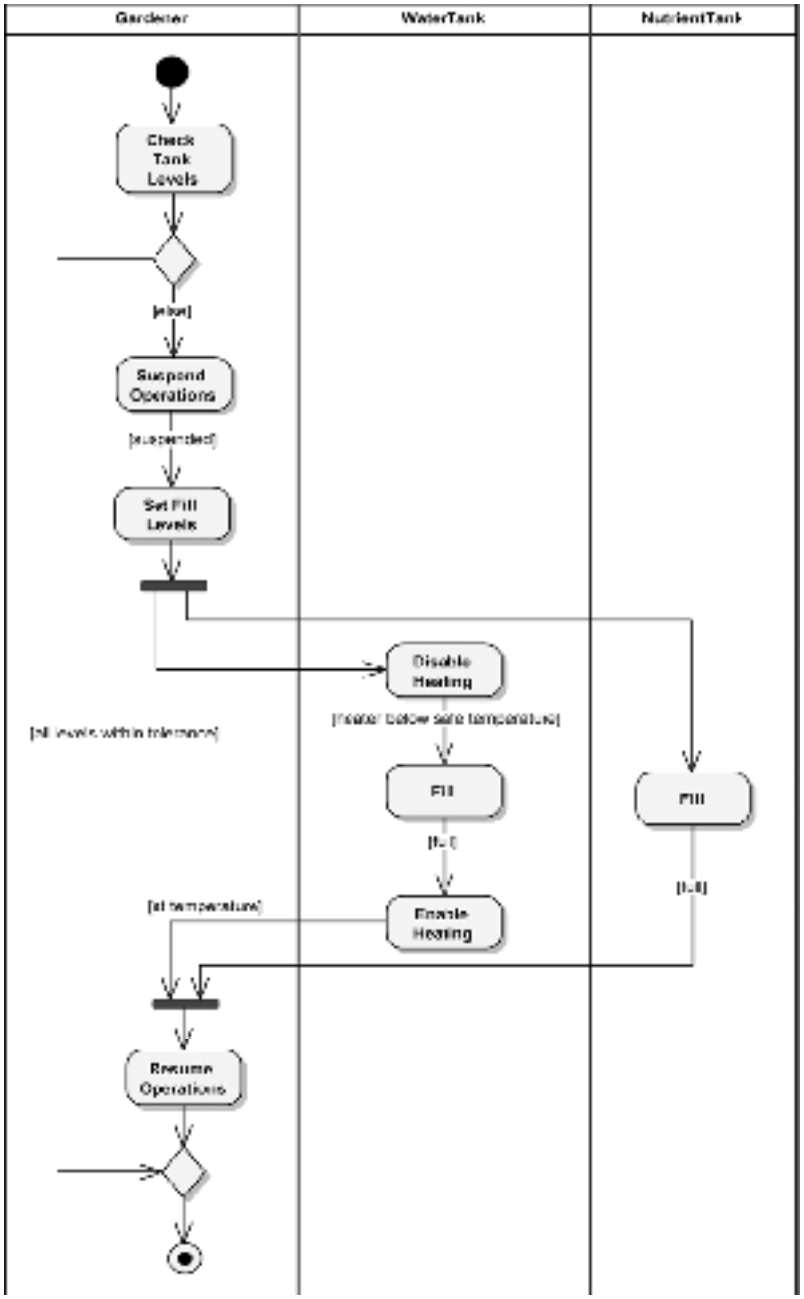


Gambar 2.11 Agregat

(Object Oriented Analysis & Design With Applications 3rd Edition, p195)

c. Activity Diagram

Menurut Grady (2007,p186) Activity Diagram mendukung aliran aktifitas dari suatu system, aliran kerja, atau proses lain.



Gambar 2.12 Activity Diagram

(Object Oriented Analysis & Design With Applications 3rd Edition, p189)

- Oval merepresentasikan aktifitas.
- Diamond merepresentasikan pilihan.
- Fork merepresentasikan awal dan akhir dari kerja yang dilakukan bersamaan.
- Black Round merepresentasikan awal dari aliran kerja.
- Black Round with Circle merepresentasikan akhir dari aliran kerja.

2.17 ASP.NET

Menurut Matthew MacDonald (2006,p7) ASP.NET adalah model pemrograman berorientasi objek yang membiarkan kita menaruh semua web page semudah kita ingin membuat aplikasi windows. ASP.NET didesain dengan teknologi *server side,* karena semua code di ASP.NET dieksekusi di server.

2.18 Database

Menurut Connolly (2005,p15) Database adalah koleksi yang berbagi antara data logika yang berhubungan, dan deskripsi dari data, yang dirancang untuk memenuhi kebutuhan informasi suatu organisasi.

2.19 Database Management System (DBMS)

Menurut Connolly (2005,p16) Database Management System adalah system software yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan mengontrol akses ke database.

2.20 Standart Query Language (SQL)

Menurut Connolly (2005,p113) SQL adalah sebuah contoh dari transform oriented language, atau bahasa yang didesain untuk menggunakan *relations* untuk mengubah input menjadi outpu yang diperlukan.