MODUL 2
KONSEP PEMODELAN
UNIFIED MODELLING LANGUAGE (UML)

Daftar Isi

2.1 Konsep Pemodelan

Pada pengembangan sistem informasi model biasanya digambarkan dalam bentuk fisik dan abstrak. Pengembangan software umumnnya dilakukan oleh sebuah tim dimana masing-masing orang dalam tim tersebut memerlukan model untuk mendapatkan gambaran dari sistem tersebut. Namun demikian walaupun software tersebut dibangun oleh satu orang tetap saja sebuah model diperlukan karena pengembangan software merupakan satu kegiatan yang kompleks.

2.2 Model dan Diagram

2.2.1 Apakah Model itu ?

Seperti halnya sebuah peta, model digunakan untuk menggambarkan sesuatu. Ada beberapa pemahaman yang berkaitan dengan model,

- Sebuah model harus cepat dan mudah untuk dibangun
- Sebuah model bisa digunakan untuk simulasi, mempelajari mengenai sesuatu yang akan direpresentasikan
- Sebuah model mampu mempelajari perkembangan dari suatu kegiatan atau masalah
- Kita bisa memilih secara rinci sebuah model
- Model bisa merepresentasikan sesuatu secara real atau tidak sebuah domain.

Ada beberapa jenis model. Pada pengembangan sistem informasi model biasanya digunakan untuk menggambarkan situasi yang komplek atau frekuensi aktivitas sistem dengan pengguna. Kita juga bisa menggunakan model untuk menggambarkan situasi yang di inginkan oleh stakeholder untuk mendefinikan kebutuhan fungsional atau non fungsional. Model secara keseluruhan harus akurat, lengkap dan tidak ambigu. Beberapa model pengembangan sistem pada saat ini bentuknya diagram dengan deskripsi tektual dan logikal atau spesifikasi matematika untuk proses dan data.

2.2.2 Apakah Diagram

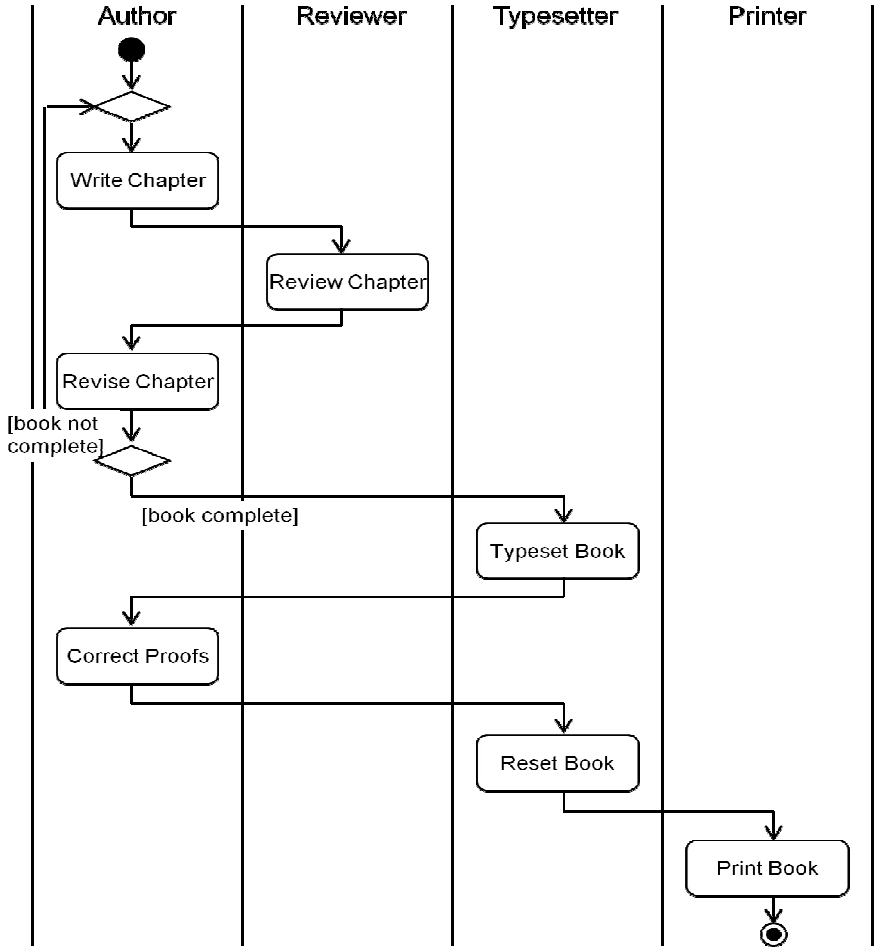
Analis dan perancang sistem menggunakan diagram untuk membuat model sebuah sistem mirip seperti seorang arsitek membuat diagram untuk membangun sebuah gedung. Fungsi diagram biasanya digunakan oleh analis dan designer untuk :

- Mengkomunikasikan ide-ide
- Mengenerate ide baru serta segala kemungkinan
- Melakukan tes terhadap ide serta membuat prediksi
- Memperlajari struktur dan hubungan suatu sistem

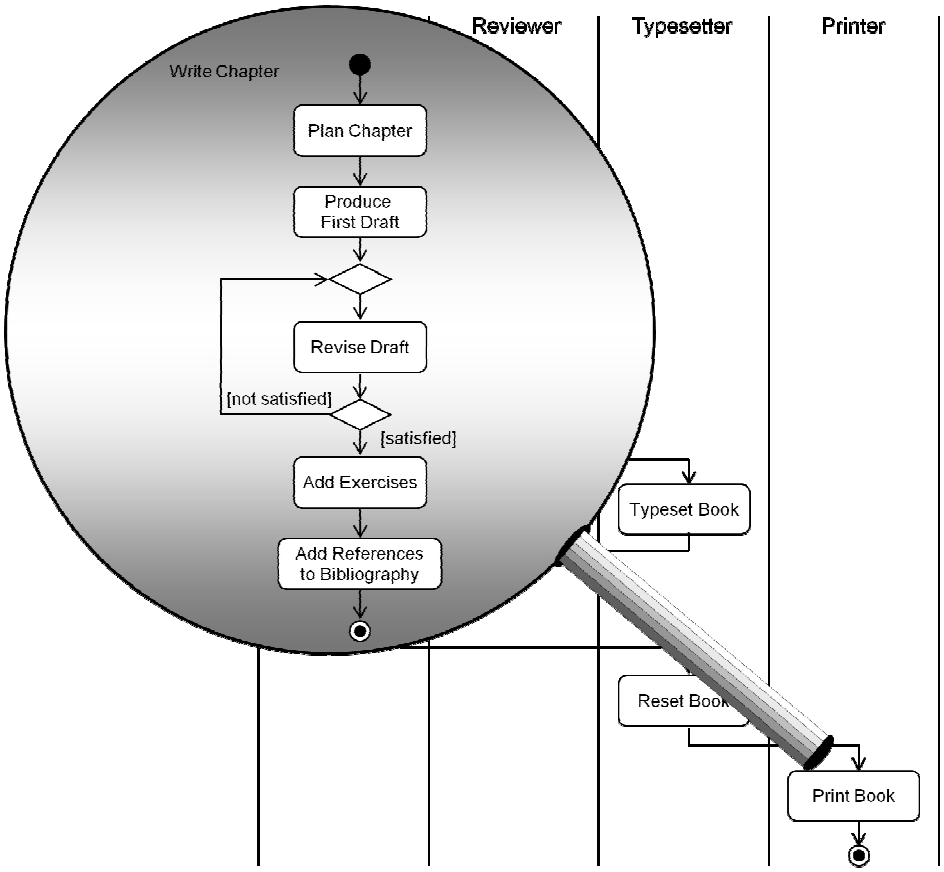
Diagram merupakan bentuk nyata yang digunakan untuk merepresentasikan sesuatu atau aksi dari dunia nyata. Diagram mengikuti aturan atau standar. Standar memastikan bahwa orang yang berbeda akan menafsirkan diagram dengan cara yang sama.

Contoh Diagram sederhana.

Contoh activity Diagram pada UML untuk membuat buku



Activity Diagram dengan detail yang tersembunyi



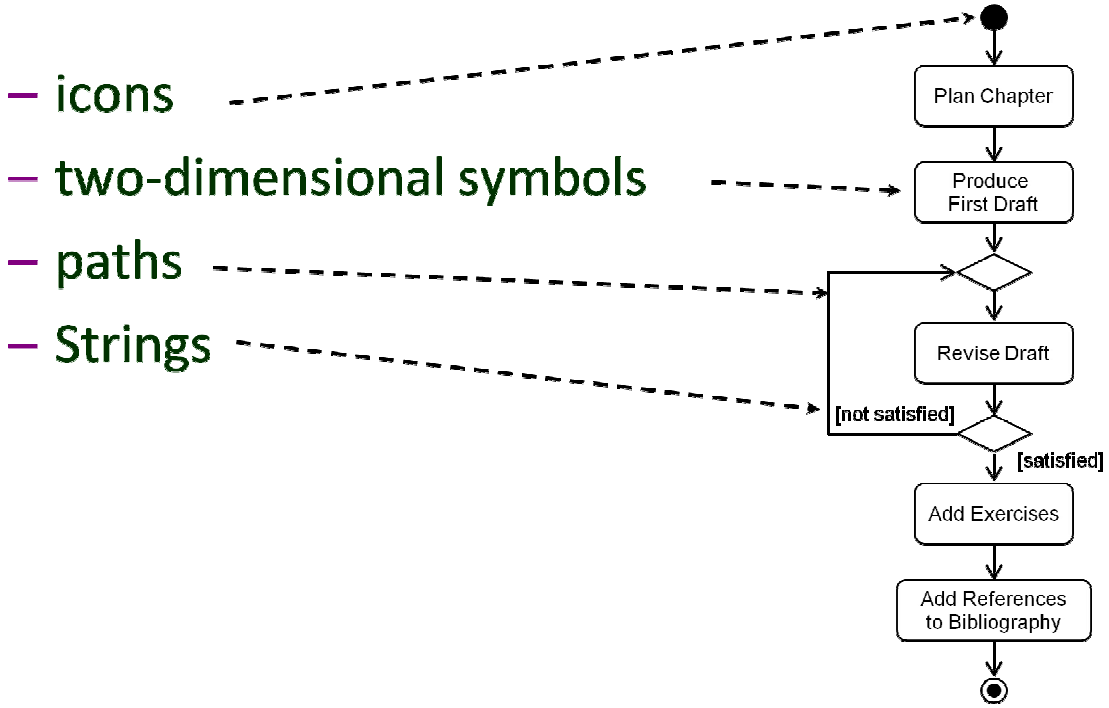
Ada beberapa hal yang digunakan sebagai acuan dalam merancang sebuah model, antara lain:

- Simplicity representation – hanya menggambarkan apa yang harus ditampilkan
- Internal consistency – pada sekumpulan diagram
- Complentness – menampilkan semua yang dibutuhkan
- Hierarchical representation – dapat diturunkan untuk melihat lebih detail pada level yang lebih rendah.

UML (Unified Modelling Language) banyak digunakan sebagai model standar dalam mengembangkan sebuah sistem informasi. UML terdiri atas beberapa gambar untuk mendefisinikan konsep atas apa yang dibutuhkan dalam membangun sistem. UML memilik 4 elemen utama, yaitu :

- Icons
- Simbol dua dimensi (Two Dimensional)

- Paths
- String



2.2.3 Perbedaan antara Model dan Diagram

Sebuah diagram bisa digunakan sebagai ilustrasi atau dokumen untuk melihat beberapa aspek dari sistem. Sedangkan model menyediakan sudut pandang yang lebih lengkap untuk setiap tahapan tertentu. Sebagai contoh model requirment sebuah sistem memberikan gambaran lengkap mengenai semua kebutuhan sistem.

2.2.4 Model dalam UML

(OMG, 2004b) mendefiniskan model sebagai berikut:

"sebuah model menangkap kebutuhan sistem secara fisik. Merupakan abstraksi dari sistem secara fisik. Tujuannya adalah memasukkan apa yang harus dimasukkan dalam sistem dan seperti apa hubungannya. Model yang lengkap menggambarkan segala aspek sistem secara fisik pada tahapan yang lebih rinci."

Pada UML ada sekumpulan konsep yang digunakan untuk menggambarkan sistem serta bagaimana cara untuk membuat modelnya . Sebuah sistem adalah segala sesuatu yang akan

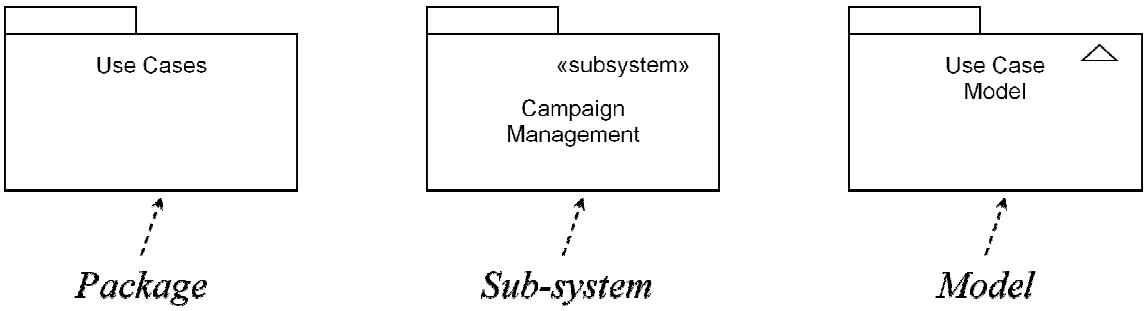
dibuat modelnya. Sebuah subsistem adalah bagian dari sistem yang terdiri atas elemen-elemen pembentuk subsistem dan sistem tersebut. Sebuah model adalah abstraksi dari sistem dan subsistem pada sudut pandang tertentu. Diagram adalah representasi grafis dari sekumpulan elemen dalam model sebuah sistem. Model yang berbeda menggambarkan sudut pandang yang berbeda dari suatu sistem. Ada 5 sudut pandang yang digunakan dalam membuat model dengan UML yaitu : usecase view, design view, process view, implementation view, deployment view.

UML menyediakan notasi untuk menggambarkan subsitem dalam bentuk *packages.*

2.2.5 Membangun Model

Model yang kita hasilkan dalam mengembangkan sistem selalu mengalami perubahan sejalan dengan perkembangan proyek. Perubahan tersebut meliputi tiga dimensi utama, antara lain :

- Abstraction(Abstraksi)
- Formality(Formalitas)
- Level of detail



Notasi UML untuk Package, SubSystem dan Model

Dalam proyek pengembangan sistem yang menggunakan siklus hidup iteratif, model yang berbeda mewakili pandangan yang sama dapat dikembangkan pada tingkat detail berbeda. Sebagai contoh model pertama dari sistem mungkin hanya menampilkan usecase dari iterasi pertama yaitu mendapatkan persyaratan (requirment). Setelah iterasi kedua, model use case dapat dijabarkan dengan lebih detail yang muncul dari pembahasan persyaratan. Beberapa prototipe dapat ditambahkan untuk mencoba ide-ide tentang bagaimana pengguna akan berinteraksi dengan sistem. Setelah iterasi ketiga, model akan diperluas untuk menyertakan deskripsi yang lebih terstruktur tentang bagaimana pengguna akan berinteraksi dengan use case . Setiap fase dalam proyek akan terdiri dari sejumlah iterasi, dan jumlah itu akan tergantung pada kompleksitas sistem yang dikembangkan.

2.3 Mengambar activity Diagram

2.3.1 Kegunaan Activity Diagram

Activity diagram bisa digunakan untuk memodelkan beberapa aspek dari sistem. Pada level yang lebih tinggi, activity diagram digunakan untuk memodelkan aktivitas bisnis yang ada atau potensial pada sistem. Umumnya activity diagram digunakan untuk beberapa tujuan, antara lain :

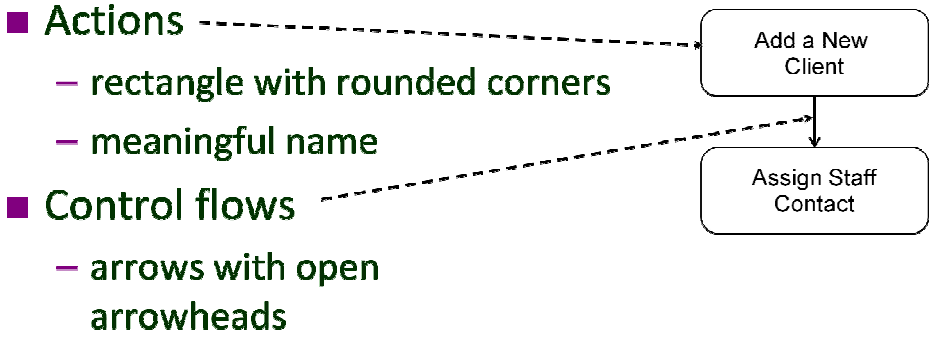
- Memodelkan proses atau task
- Mengambarkan fungsi sistem yang direpresentasikan oleh usecase
- Pada spesifikasi operasional digunakan untuk menggambarkan logika operasi

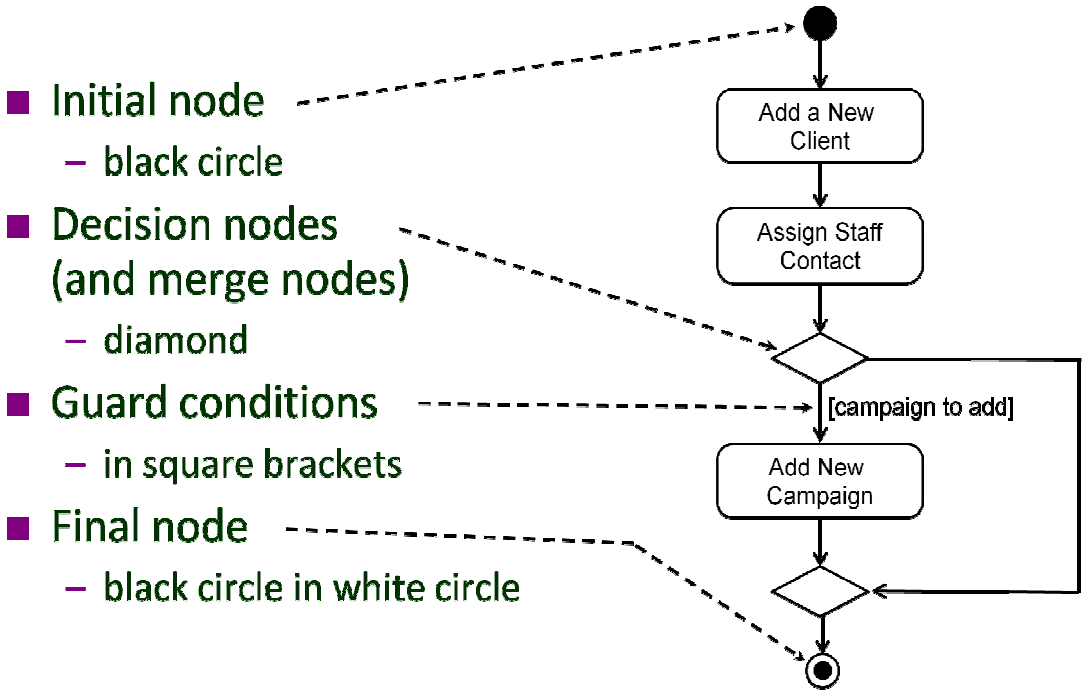
- Pada USDP (Unified Software Development Process) untuk memodelkan aktifitas yang membentuk siklus hidup (lifecycle)

2.3.2 Notasi pada Activity Diagram

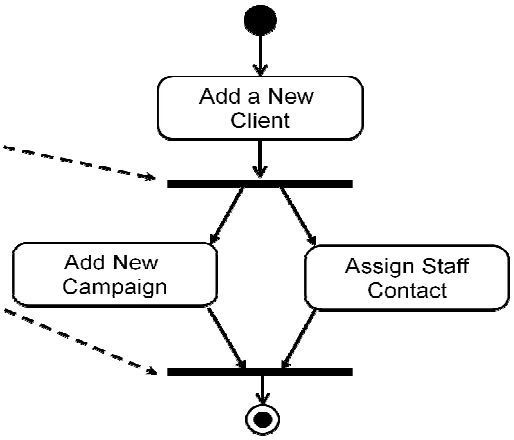
Activity diagram sederhana terdiri atas sekumpulan aksi yang dihubungkan dengan aliran (flow) dari suatu aktifitas ke aktifitas berikutnya yang dikenal dengan *activityedges.* Setiap aktivitas digambarkan dengan kotak dengan sudut tumpul. Namanya aktivitas ditulis di dalam kotak tesebut. Nama harus memiliki arti dan merupakan gambaran dari aktivitas tersebut.

Contoh

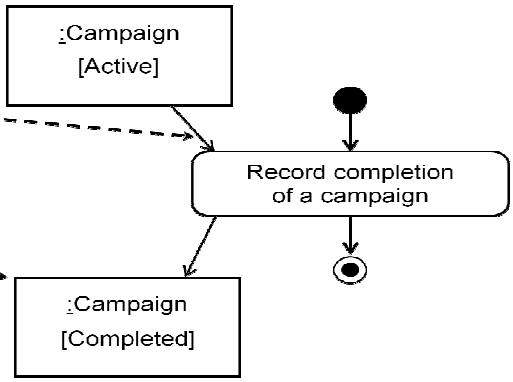




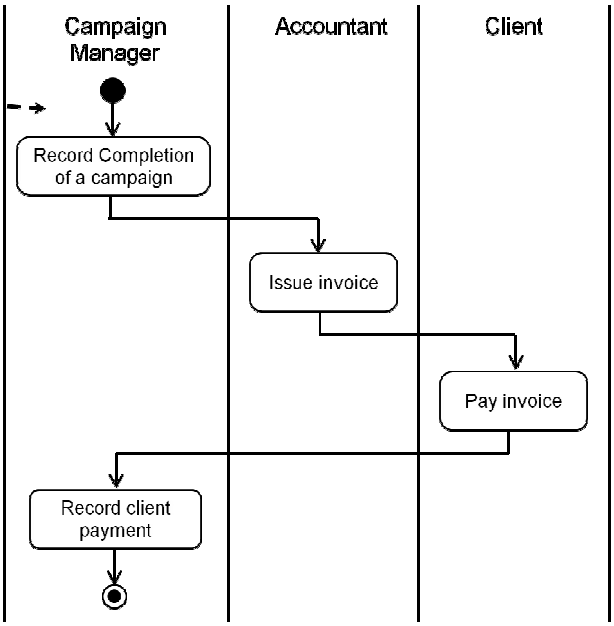
- Fork nodes and join nodes
  - thick bar
- Actions carried out in parallel



- Object flows
  - open arrow
- Objects
  - rectangle
  - optionally shows the state of the object in square brackets



- **Activity Partitions (Swimlanes)**
  - vertical columns
  - labelled with the person, organisation, department or system responsible for the activities in that column



Contoh Kasus.

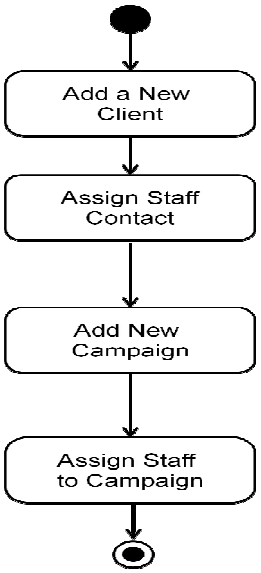
System Agate

Langkah-langkah :

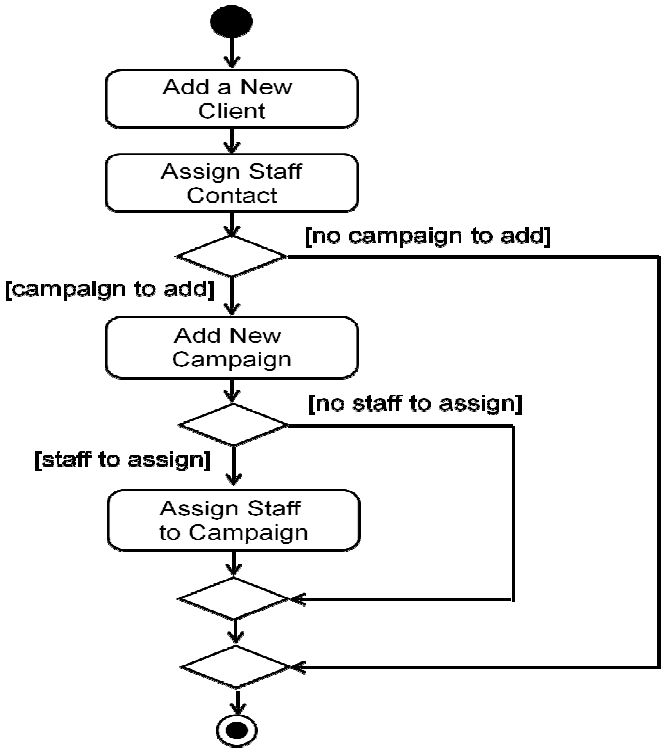
1. Tentukan Tujuan

   Pada sistem Agate, tujuannya adalah untuk menampilkan beberapa aktifitas yang ada pada sistem tersebut.
2. Apa saja yang akan ditampilkan pada diagram, dalam hal ini nama dari proses bisnis, usecase atau operasinya
3. Sampai level detail mana proses tersebut dibutuhkan apakah hanya global saja atau lebih rinci
4. Identifikasi setiap action/aksi.

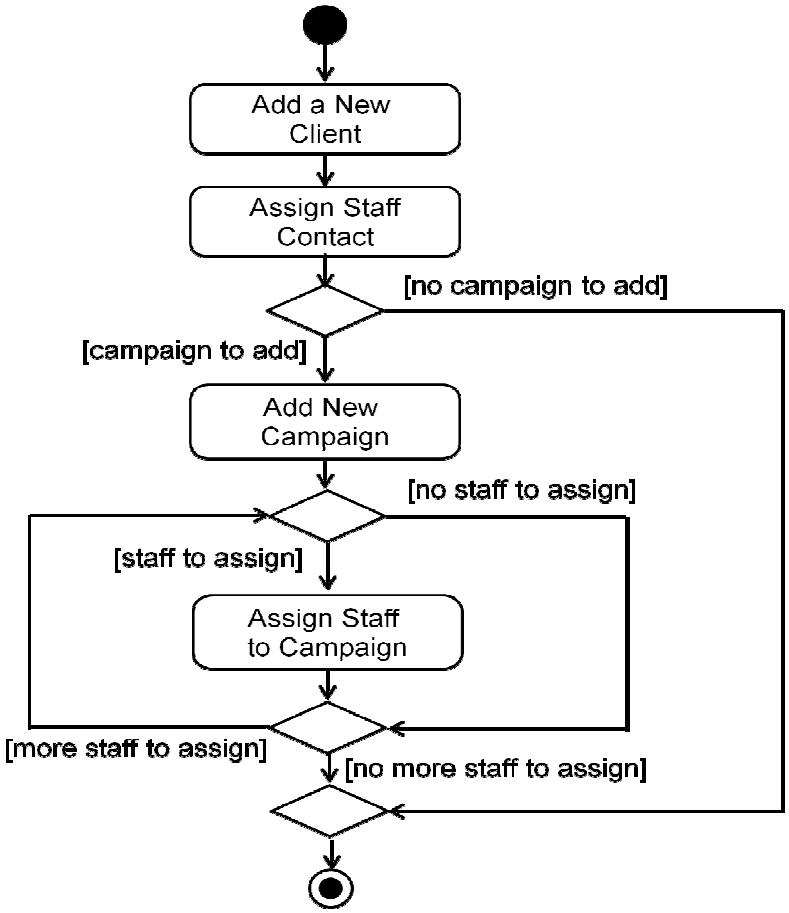
   Pada sistem Agate aksi yang dikerjakan adalah
   - Add a New Client
   - Assign Staff Contact
   - Add New Campaign
   - Assign Staff to Campaign
5. Organisasikan setiap aksi dalam bentuk aliran data



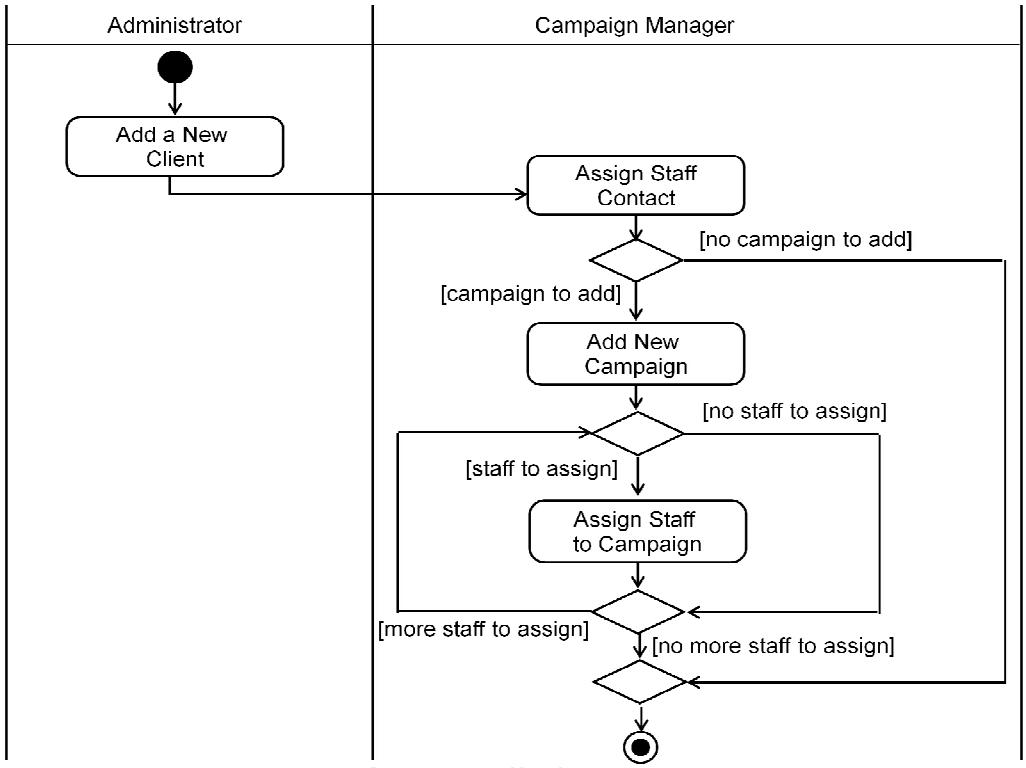
6. Identifikasi altenatif aliran data untuk setiap kondisi
7. Tambahkan node decision jika diperlukan



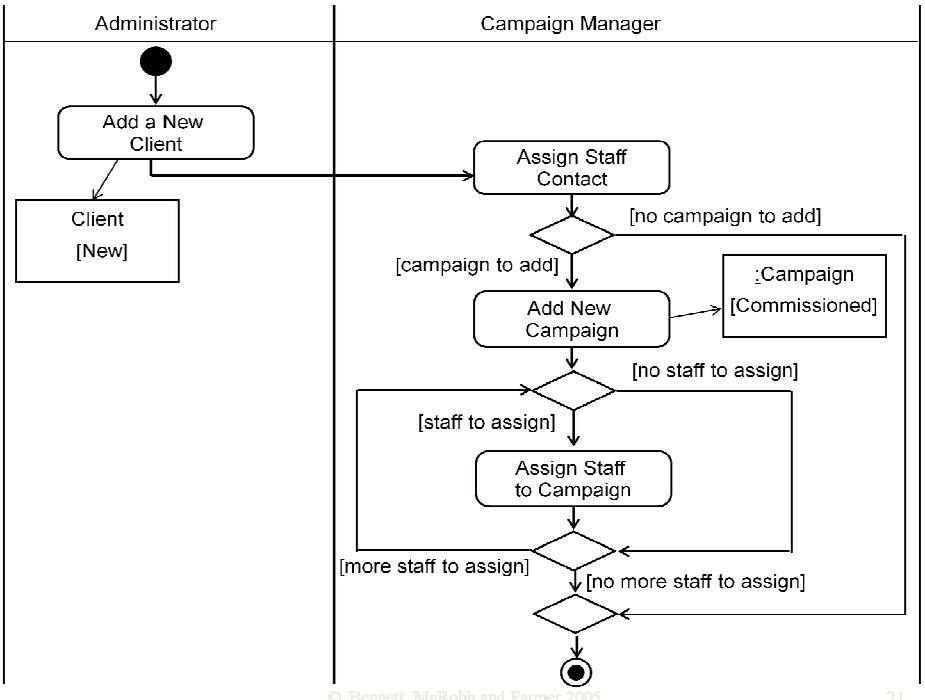
8. Identifikasi aksi yang berjalan secara pararel
9. Tambahkan notasi fork atau Join jika diperlukan
10. Identifikasi proses yang berulang



11. Tambahkan swimline untuk mengidentifikasi siapa yang melakukan aktifitas.



12. Masukkan object Flow dan object yang diperlukan



2.4 Tinjauan UML

2.4.1 Sejarah UML (Unified Modelling Language)

Pemodelan berorientasi objek mulai muncul antara pertengahan 1970 dan 1980-an. Berbagai metodologi digunakan untuk melakukan analisa dan perancangan. Banyaknya pemodelan diidentifikasi meningkat kurang dari 10 sampai lebih dari 50 selama periode antara 1989-1994. Banyak pengguna metode berorientasi object mengalami kesulitan menemukan satu pemodelan yang akan digunakan sehingga memicu "perang metode." Pada pertengahan 1990-an, iterasi baru dari metode ini mulai muncul dan metode ini mulai menggabungkan teknik masing-masing dari beberapa tokoh.

Pengembangan UML dimulai pada akhir 1994 ketika Grady Booch dan Jim Rumbaugh dari Rational Software Corporation mempersatukan model Booch dan OMT (Object Modeling Technique) . Dalam 1995, IvarJacobson dan perusahaan Objectory nya menggabungkan Rasional dalam upaya unifikasi, penggabungan ini membentuk metode OOSE (Object-Oriented Software Engineering) .

Sebagai penulis utama dari Booch, OMT, dan metode OOSE, Grady Booch, Jim Rumbaugh , dan Ivar Jacobson termotivasi untuk menciptakan sebuah bahasa pemodelan terpadu yang pada akhirnya pada oktober 1996 mumcullah UML versi 0.9
Selama tahun 1996 UML mulai d lirik sebagai bagian dari OMG (Object Management Group) dan mulai january 1997 mulai dimasukkan dalam RFP (A Request for Proposal) sebagai bagian dari OMG.

2.4.2 Artifak UML

UML menyediakan beberapa notasi dan artifak standar yang dapat digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain sistem. Artifak dalam UML didefinisikan sebagai informasi dalam berbagai bentuk yang digunakan atau dihasilkan dalam proses pengembangan software. Terdapat beberapa artifak utama dalam UML, yaitu :

a. Use Case Diagram
   Diagram yang menggambarkan actor, use case dan relasinya
b. Class Diagram
   Diagram untuk menggambarkan kelas dan relasi diantara kelas-kelas tersebut

c. Behaviour Diagram, yang terdiri dari :

i. Activity Diagram

Menggambarkan aktifitas-aktifitas, objek, state, transisi state dan event

ii. Collaboration Diagram

Menggambarkan objek dan relasinya, termasuk struktur perubahannya yang disebabkan oleh adanya suatu message

iii. Sequence Diagram

Menggambarkan objek dan relasinya termasuk kronologi (urutan) perubahan secara logis setelah menerima sebuah message

iv. Statechart Diagram

Menggambarkan state, transisi state dan event

d. Implementation Diagram, terdiri dari :

i. Component Diagram

Menggambarkan komponen dan relasi antara komponen tersebut

ii. Deployment DIagram

Menggambarkan komponen, titik awal dan relasi antara komponen tersebut

2.4.3 Notasi dalam UML

UML memiliki notasi untuk menjelaskan secara visual mengenai elemen-elemen pemodelan. Pada diagram Use-Case terdapat notasi untuk use-case, actor dan System. Sedangkan notasi untuk menggambarkan Class diagram, terdiri dari notasi Class, asosiasi, agregasi, generalisasi dan spesialisasi dan seterusnya.

Beberapa notasi dalam UML :

1. Actor
2. Association
3. Class
4. Generalization
5. Use Case dan use case specification
6. Use Case Diagram
7. Realization
8. Sequence Diagram

9. Interaction
10. Class Diagram
11. Dependency
12. Package
13. Note
14. Interface, dll

Referensi

1. Simon Bennet, Steve McRobb and Ray Farmer, *Object Oriented Systems Analysis and Design Using UML,* Edisi 3. ; McGraw Hill, 2006. (SB)