

BAB II

TINJAUAN PUSTAKA

2.1 Website

World Wide Web atau biasa disebut *web* sebetulnya merupakan suatu kumpulan informasi pada beberapa server komputer yang terhubung satu sama lain dalam jaringan internet. Informasi-informasi dalam *web* mempunyai *link-link* yang menghubungkan informasi tersebut ke informasi lain di dalam jaringan internet[1].

Situs *web* (*website*) adalah kumpulan dari halaman *web* yang saling berhubungan dengan hal-hal lain yang terkait, seperti dokumen dan gambar yang disimpan dalam suatu server *web*. Server *web* (*web server*) adalah komputer yang melayani permintaan halaman-halaman *web* dan mengirimkannya ke komputer. *Web* adalah gudang informasi global. Salah satu kegunaan utama dari *web* adalah pencarian informasi spesifik, termasuk teks, gambar, audio, dan video[2].

Website adalah keseluruhan halaman-halaman *web* yang terdapat dalam sebuah domain yang mengandung informasi. Sebuah *website* biasanya dibangun atas banyak halaman *web* yang saling berhubungan. Hubungan antara satu halaman *web* dengan halaman *web* yang lainnya disebut dengan *hyperlink*, sedangkan teks yang dijadikan media penghubung disebut *hypertext*[3].

2.2 Metodologi Pengembangan Sistem

Prototyping sistem informasi adalah suatu teknik yang sangat berguna untuk mengembangkan informasi tertentu mengenai syarat-syarat informasi pengguna secara cepat. Dengan menggunakan *prototyping*, analisis sistem berupaya memperoleh reaksi awal dari para pengguna dan pihak manajemen terhadap prototipe, saran-saran pengguna terhadap perubahan atau pemecahan masalah sistem yang dibuat prototipenya, sehingga memungkinkan dilakukan inovasi mengenai prototipe tersebut, serta rencana-rencana revisi yang mendetail dengan bagian-bagian sistem yang perlu dilakukan lebih dulu[4].

Prototipe memberikan ide bagi pembuat maupun pemakai tentang cara sistem berfungsi dalam bentuk lengkapnya. Proses menghasilkan sebuah prototipe disebut *Prototyping*[4].

Jenis-jenis informasi yang dicari saat melakukan *prototyping* adalah sebagai berikut[4]:

1. Reaksi awal dari pengguna

Saat analisis sistem menampilkan sebuah prototipe sistem informasi, maka analis akan tertarik dengan reaksi pengguna dan pihak manajemen terhadap prototipe. Analis ingin tahu secara mendetail bagaimana reaksi mereka saat bekerja dengan prototipe dan apakah fitur-fitur sistem yang diprototipekan sudah sesuai dengan kebutuhan mereka. Untuk mengetahui reaksi dari pengguna ini dapat digunakan lembar evaluasi.

2. Saran-saran dari pengguna

Analisis juga tertarik dengan saran-saran pengguna dan pihak manajemen perbaikan terhadap prototipe yang ditampilkan. Saran-saran diperoleh dari pengalaman saat bekerja dengan prototipe selama periode waktu tertentu. Waktu yang dihabiskan pengguna saat bekerja dengan prototipe biasanya tergantung kepada dedikasi mereka serta ketertarikan atas proyek sistem. Saran-saran merupakan hasil dari interaksi pengguna dengan prototipe serta refleksi mereka atas interaksi tersebut. Saran yang diperoleh dari pengguna member petunjuk pada analisis tentang cara-cara memperbaiki, mengubah atau menghentikan prototipe sehingga bias memenuhi kebutuhan pengguna dengan lebih baik.

3. Inovasi

Inovasi prototipe merupakan bagian dari informasi yang dicari oleh tim analisis sistem. Inovasi adalah kemampuan-kemampuan sistem baru yang tidak dianggap berhubungan dengan waktu saat pengguna mulai berinteraksi dengan prototipe. Inovasi-inovasi ini memberi nilai tambah terhadap fitur-fitur yang diprototipekan sebelum dengan menambahkan sesuatu yang baru atau yang lebih inovatif.

4. Rencana revisi

Rencana revisi membantu mengidentifikasi prioritas apa yang akan diprototipekan selanjutnya. Informasi yang terkumpul dalam fase *prototyping* memungkinkan analis menyusun prioritas-prioritas dan memberi pengarahannya kembali rencana-rencana tersebut dengan lebih efisien, serta dengan gangguan minimum. Karena fitur inilah, *prototyping* dan perencanaan bisa dilaksanakan bersama-sama.

Langkah-langkah yang terdapat dalam pengembangan prototipe adalah sebagai berikut[4]:

1. Pengumpulan kebutuhan.

Analisis sistem mewawancarai pemakai untuk mendapatkan gagasan dari apa yang diinginkan pemakai terhadap sistem.

2. Membangun prototipe.

Analisis sistem mungkin bekerja sama dengan spesialis informasi lain, menggunakan satu atau lebih pendekatan *prototyping* untuk mengembangkan sebuah prototipe.

3. Mengevaluasi *prototyping*.

Analisis mendidik pemakai untuk menggunakan prototipe dan memberi kesempatan kepada mereka untuk membiasakan diri dengan sistem. Pemakai member masukan kepada analisis apakah prototipe memuaskan. Jika ya, langkah 4 akan diambil, jika tidak, prototipe direvisi dengan mengulangi langkah 1,2 dan 3.

Daya Tarik *Prototyping*.

Pemakai maupun spesialis informasi menyukai *prototyping*, untuk alasan-alasan, sebagai berikut[4]:

1. Komunikasi antara analisis sistem dengan pemakai, membaik.
2. Analisis dapat bekerja dengan lebih baik dalam menentukan kebutuhan pemakai.
3. Pemakai berperan lebih aktif dalam pengembangan sistem.
4. Spesialis informasi dan pemakai menghabiskan lebih sedikit waktu dan usaha dalam pengembangan sistem
5. Penerapan menjadi mudah karena pemakai mengetahui apa yang diharapkan.

Keuntungan-keuntungan ini memungkinkan *prototyping* menghemat biaya pengembangan dan meningkatkan kepuasan pemakai dengan sistem yang dihasilkan.

2.3 Alat Pengembangan Sistem

Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh model-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO). UML merupakan standar yang relatif terbuka yang dikontrol oleh *Object Management Company* (OMC), sebuah konsorsium terbuka yang terdiri dari banyak perusahaan. *Unified Modeling Language* (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, menggambarkan, dan membangun sistem perangkat seperti halnya pada *business modeling* dan sistem lainnya. UML tidak berdasarkan pada bahasa pemrograman tertentu. Standar spesifikasi UML dijadikan standar *defacto* oleh OMG (*Object Management Group*) pada tahun 1995. UML yang berorientasikan *object* mempunyai beberapa notasi

standar. Spesifikasi ini menjadi populer dan standar karena sebelum adanya UML, telah ada berbagai macam spesifikasi yang berbeda. Hal ini menyulitkan komunikasi antar pengembang perangkat lunak. Untuk itu beberapa pengembang spesifikasi yang sangat berpengaruh berkumpul untuk membuat standar baru[5].

2.3.1 Use Case Diagram

Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan. *Use case* diagram menampilkan aktor mana yang menggunakan *use case* mana, *use case* mana yang memasukkan *use case* lain dan hubungan antara aktor dan *use case*. *Use case* diagram menggambarkan interaksi antara sistem dengan sistem eksternal dan pengguna. Dengan kata lain, secara grafis menggambarkan siapa yang menggunakan sistem dan dengan cara apa pengguna mengharapkan untuk berinteraksi dengan sistem.

Use case diagram dapat digunakan selama proses analisis untuk menangkap *requirement* sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *use case* diagram berperan untuk menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa *use case* diagram. Kebutuhan atau *requirementsystem* adalah fungsional apa yang harus disediakan oleh sistem kemudian didokumentasikan pada model *use case* yang menggambarkan fungsi sistem yang diharapkan (*use case*), dan mengelilinginya (*actor*), serta hubungan antara aktor dengan *use case* (*Use Case Diagram*) itu sendiri. *Use case* diagram memiliki dua komponen utama, yaitu[5]:

a. *Actor*

Pada dasarnya *actor* bukanlah bagian dari *use case* diagram, namun untuk dapat terciptanya suatu *use case* diagram diperlukan beberapa *actor*. *Actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah *actor* mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima, dan memberi informasi pada sistem. *Actor* hanya berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. *Actor* digambarkan dengan *stick man*. *Actor* dapat digambarkan secara umum atau spesifik, dimana untuk membedakan kita dapat menggunakan *relationship*.

Ada beberapa kemungkinan yang menyebabkan *actor* tersebut terkait dengan sistem antara lain:


1. Yang berkepentingan terhadap sistem dimana adanya arus informasi, baik yang diterimanya maupun yang dia inputkan ke sistem.
 2. Orang ataupun pihak yang akan mengelola sistem tersebut.
 3. *External resource* yang digunakan oleh sistem.
 4. Sistem lain yang berinteraksi dengan sistem yang akan dibuat.
- b. *Use Case*

Use case adalah gambaran fungsional dari suatu sistem, pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun. *Use case* diagram adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian.

Cara menentukan *use case* dalam suatu sistem:

1. Pola perilaku perangkat lunak aplikasi
2. Gambaran tugas dari sebuah *actor*
3. Sistem atau “Benda” yang memberikan sesuatu yang bernialai kepada *actor*
4. Apa yang dikerjakan oleh suatu perangkat lunak.

UNIVERSITAS
MIKROSKIL

SIMBOL	NAMA	KETERANGAN
	<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
	<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
	<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
	<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara eksplisit.
	<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
	<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (<i>sinergi</i>).
	<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Gambar 2. 1 Simbol-simbol Use Case Diagram[5]

2.3.2 Class Diagram

Class diagram merupakan himpunan dari objek-objek yang sejenis. Sebuah objek memiliki keadaan sesaat (*state*) dan perilaku (*behavior*). *State* sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam *attribute/properties*. Sedangkan perilaku suatu objek mendefinisikan bagaimana sebuah objek bertindak /beraksi dan memberikan reaksi. Notasi-notasi yang digunakan dalam diagram kelas adalah sebagai berikut[6]:

a. *Class*

Menyatakan kelas yang digunakan. Diagram ini berisikan tiga komponen, yaitu nama kelas, atribut dalam kelas, dan *behavior*. Atribut merepresentasikan parameter dan data-data yang terdapat dalam kelas. *Behavior* menyatakan fungsi atau *method* yang berlaku dalam kelas tersebut.

b. *Inheritance*

Menyatakan hirarki dari suatu kelas sebagai komponen kelas lain yang juga disebut sub objek.

c. *Agregation*

Merupakan sebuah bentuk asosiasi yang menyatakan bagian dari keseluruhan dan digambarkan dalam notasi berbentuk *diamond*.

d. *Message*

Message (Pesan) merupakan cara untuk berhubungan antara satu objek dengan objek lain. Suatu pesan dikirimkan oleh suatu objek kepada objek tertentu dapat digambarkan sebagai anak panah. Objek pengirim mengirimkan pesan kepada objek penerima supaya objek penerima melaksanakan salah satu metode yang dimilikinya.

2.4 Wedding Organizer

Wedding Organizer adalah suatu jasa yang berfungsi secara pribadi membantu calon pengantin dan keluarga dalam perencanaan dan supervisi pelaksanaan rangkaian acara pesta pernikahan sesuai dengan jadwal dan *budget* yang telah ditetapkan. Sistem informasi *Wedding organizer* dibuat dengan memanfaatkan teknologi informasi berbasis *web*, dengan tujuan untuk memperluas area promosi dan penjualan paket pernikahan dan mempermudah pemesanannya. Sistem informasi persewaan *Wedding Organizer* menggunakan *Balsamiq mockups 3*, metode yang digunakan dalam pengembangan sistem ini yaitu menggunakan metode *prototyping* (analisa, *design*).

Sistem informasi *Wedding Organizer* sangat membantu bagi orang-orang yang tidak mau repot dengan masalah perencanaan pernikahan mereka. Dengan berkembangnya teknologi sekarang ini *Wedding Organizer* sudah mulai berpindah dari proses manual kedalam proses *online* yaitu sistem informasi *Wedding Organizer* yang berbaris *web*[7].

2.4.1 Kelebihan dan kekurangan Wedding Organizer

Kelebihan *Wedding Organizer*[8]:

1. Lebih Murah, Biasanya dengan menggunakan paket dari *Event Organizer* (EO) *Wedding* bisa diberi diskon khusus atau bonus tambahan bahkan sebelum terjadi diskon bisa melakukan negosiasi harga terlebih dahulu, tetapi *budget* juga disesuaikan.
2. Menghemat Waktu, dengan adanya *Organizer* maka para pasangan bisa menghemat waktu sehingga bagi pasangan yang sibuk sekali bekerja dan sulit mengatur waktu maka dapat memilih *Wedding Organizer* yang akan mengurus pernikahan.

3. Dikoordinasikan 1 orang, Minimal akan ada 1 orang yang akan mengkoordinasikan semua *vendor-vendor* yang sudah bekerja sama dengan jasa *organizer* tersebut. Pada satu orang ini bisa berkonsultasi dengan beberapa masalah yang diinginkan seputar pernikahan.
4. Mengurangi *Stress*, menjadi calon pengantin akan membuat tingkat *stress* bertambah, mempersiapkan diri untuk peran yang baru sebagai istri sudah cukup menguras pikiran terlebih lagi harus menyiapkan segala sesuatu yang berhubungan dengan acara pesta.
5. Lebih Tenang, mungkin memang setiap pasangan harus waspada karena tidak semudah itu memberikan kepercayaan kepada orang lain, namun jika setiap pasangan menggunakan jasa *wedding organizer* pasti setiap pasangan mempercayainya karena sudah terbukti integritas dan kredibilitasnya.

Kekurangan *Wedding Organizer*[8]:

1. Dalam sistem paket banyak memberikan keuntungan akan tetapi dengan terbatasnya permintaan maka tidak semua *detail* yang diinginkan pasangan bisa dipenuhi oleh *vendor* yang ada. jika sudah mengetahui keinginan pesta pernikahan yang seperti apa, sebaiknya mencari dulu *vendor-vendor* yang dapat memenuhi keinginan pasangan atau tidak. Kemudian pasangan bisa memutuskan untuk mengambil sistem paket atau tidak.
2. Biaya untuk *wedding organizer* bisa lebih mahal atau bisa lebih murah, tergantung setiap pasangan yang ingin menikah. Jika pasangan memiliki kenalan atau teman dan kerabat maka biaya biasanya diberi potongan harga.
3. Terkadang *Wedding Organizer* kurang profesional karena tidak melibatkan keluarga sebagai pemantau dan kerap terjadi, hal seperti inilah yang harus dihindari. *Wedding Organizer* seperti itu hanya akan fokus pada calon pengantin saja dan lupa bahwa keluarga seperti orangtua perlu diikutsertakan, tetapi bukan sebagai pengambil keputusan.