

## **BAB 2**

### **TINJAUAN PUSTAKA**

#### **2.1 Penelitian Terkait**

Penelitian yang disusun penulis ini berpedoman pada penelitian terdahulu yaitu sebagai berikut:

Penelitian yang disusun penulis berdasarkan penelitian dengan judul “Pembuatan Sistem Informasi Pendidikan Dan Pelatihan Dalam Jabatan Berbasis *Web* Pada Bagian Pengembangan Pegawai Direktorat Jenderal Perbendaharaan” yang disusun oleh Rachmat Gerhantaara, dan Febriliyan Samopa (2013). Proses administrasi selama ini dilakukan dengan semi manual, dimana permintaan dan pendaftaran calon peserta diklat dilakukan secara manual. Sedangkan untuk administrasi data peserta maupun lulusan proses diklat telah menggunakan aplikasi *desktop* yang bersifat *stand-alone*. Sehingga, dirancang sebuah sistem informasi mengenai pendidikan dan pelatihan berbasis *web* yang mengambil sumber dari proses bisnis yang ditetapkan pada Bagian Pengembangan Pegawai Sekretariat Direktorat Jenderal Perbendaharaan. Proses pembangunan yang dilakukan berbasis pada *iconix process* yang tersusun atas beberapa tahapannya [1].

Penelitian lainnya yang disusun penulis berdasarkan penelitian dengan judul “Implementasi *E-commerce* Berbasis Prestashop Pada UD. Bumi Putera” yang disusun oleh Muhammad Iqbal Fanani, dan Lalang Erawan (2015). UD. Bumi Putera menggunakan media penyampaian informasi yang masih sederhana berupa penyebaran brosur dan letak perusahaan yang menjual produk *inflatable* tersebut tidak strategis sehingga dalam pemasarannya belum maksimal. Dengan masalah yang dihadapi oleh UD. Bumi Putera, maka dibutuhkan sebuah *e-commerce*. Oleh karena itu, dirancang dan dibuat *website e-commerce* berbasis Prestashop untuk memfasilitasi pelanggan dalam proses pembelian produk serta dapat

meningkatkan pendapatan dan penjualan perusahaan. Metode pengembangan sistem yang digunakan adalah *waterfall* [2].

Penelitian lainnya yang disusun penulis berdasarkan penelitian dengan judul “Pengembangan Aplikasi *Web* Dengan *ICONIX Process* Dan UML Studi Kasus: Sistem Manajemen Isi” yang disusun oleh Yulianta, dan Petrus Mursanto (2012). Dalam hal ini pengembangan pada aplikasi sangat memerlukan strategi di dalam manajemen isi, hal ini dilihat dari kompleksitas yang meningkat sehingga modifikasi sulit dilakukan dan seandainya modifikasi dilakukan, kestabilan sistem aplikasi menjadi terganggu. Sehingga diperlukan adanya sebuah aplikasi berbasis *web* yang dapat digunakan untuk mengelola seluruh sumber daya situs web. Tujuannya agar memudahkan pencarian informasi dan dapat menangani data volume yang cukup besar sehingga dapat digunakan di dalam pembuatan situs yang lebih besar. Oleh karena itu, dikembangkan aplikasi CMS yang berupa prototipe aplikasi manajemen isi yang diberi nama xCMS. Metode yang digunakan adalah *ICONIX Process* [3].

Penelitian lainnya yang disusun penulis berdasarkan penelitian dengan judul “Pembuatan Sistem Informasi Beasiswa Internal Direktorat Jenderal Perbendaharaan Menggunakan PHP dan MySQL” yang disusun oleh Candra Dwi Aprida, dan Febriliyan Samopa (2013). Pengelolaan beasiswa yang dilakukan Bagian Pengembangan Pegawai Direktorat Jenderal Perbendaharaan belum didukung oleh aplikasi. Dokumentasi terhadap data-data yang ada dalam bentuk *hardcopy* sehingga ketika data-data tersebut diperlukan maka berkas data tersebut harus dicari terlebih dahulu dalam tempat penyimpanan berkas. Selain memerlukan waktu dengan semakin banyaknya berkas yang ada, tentunya juga memerlukan tempat penyimpanan berkas yang semakin banyak. Hal ini menjadi masalah ketika data-data tersebut diperlukan dengan segera. Sehingga, dibuatlah sistem informasi beasiswa internal untuk memudahkan proses *management* data terkait dengan data yang ada. Metode pembangunan sistem yang digunakan adalah *Iconix Process* [4].

Dari beberapa penelitian terkait yang telah dijabarkan sebelumnya, lalu penulis merangkumnya ke bentuk tabel yang terdiri dari nama peneliti beserta tahun, permasalahan, metode yang digunakan, dan hasil penelitian. Rangkuman penelitian terkait berbentuk tabel sebagai berikut:

**Tabel 2.1 Penelitian Terkait**

No.	Nama Peneliti dan Tahun	Masalah	Metode	Hasil
1.	Rachmat Gerhantaara, dan Febriliyan Samopa (2013).	Proses administrasi selama ini dilakukan dengan semi manual, dimana permintaan dan pendaftaran calon peserta diklat dilakukan secara manual.	<i>Iconix process.</i>	Sistem informasi mengenai pendidikan dan pelatihan berbasis <i>web</i> .
2.	Muhammad Iqbal Fanani, dan Lalang Erawan (2015).	UD. Bumi Putera menggunakan media penyampaian informasi yang masih sederhana berupa penyebaran brosur dan letak perusahaan yang	<i>Waterfall.</i>	<i>Website e-commerce</i> berbasis Prestashop pada UD. Bumi Putera.

No.	Nama Peneliti dan Tahun	Masalah	Metode	Hasil
		menjual produk <i>inflatable</i> tersebut tidak strategis sehingga dalam pemasarannya belum maksimal.		
3.	Yulianta, dan Petrus Mursanto (2012).	Kompleksitas yang meningkat sehingga modifikasi sulit dilakukan dan seandainya modifikasi dilakukan, kestabilan sistem aplikasi menjadi terganggu.	<i>ICONIX Process.</i>	Prototipe aplikasi manajemen isi yang diberi nama xCMS.
4.	Candra Dwi Aprida, dan Febriliyan Samopa (2013).	Pengelolaan beasiswa yang dilakukan Bagian Pengembangan Pegawai Direktorat Jenderal Perbendaharaan belum didukung oleh aplikasi.	<i>Iconix Process.</i>	Aplikasi pengelolaan beasiswa internal yang ditujukan untuk membantu pengelolaan beasiswa pada Bagian Pengembangan Pegawai Direktorat

No.	Nama Peneliti dan Tahun	Masalah	Metode	Hasil
		<p>Dokumentasi terhadap data-data yang ada dalam bentuk <i>harcopy</i> sehingga ketika data-data tersebut diperlukan maka berkas data tersebut harus dicari terlebih dahulu dalam tempat penyimpanan berkas. Selain memerlukan waktu dengan semakin banyaknya berkas yang ada, tentunya juga memerlukan tempat penyimpanan berkas yang semakin banyak.</p>		<p>Jenderal Perbendaharaan.</p>

Oleh karena itu, disusunnya penelitian ini bertujuan untuk membangun sebuah *e-commerce* dimana dirancang dengan menggunakan metode *ICONIX Process* dan dibangun menggunakan CMS (*Content Management System*) dalam pengimplementasiannya. Sehingga nantinya dapat memudahkan konsumen dalam melakukan transaksi pembelian secara *online* tanpa harus datang langsung ke toko sehingga waktu yang digunakan lebih cepat dan efektif. Selain itu, bagi pihak perusahaan dapat menerapkan strategi penjualan dengan lebih efektif dan efisien.

## **2.2 Situs Web (*Website*)**

### **2.2.1 Pengertian Situs Web**

Situs web adalah kumpulan laman *web* yang digabung menjadi satu. Kumpulan laman *web* tersebut saling berhubungan melalui fasilitas *hyperlink*. Situs web tersebut membawa misi tertentu, baik itu pribadi, kelompok, atau bisnis [5].

### **2.2.2 Jenis-Jenis Situs Web**

Situs web terbagi menjadi dua jenis, yaitu [6]:

#### **1. Situs Web Statis**

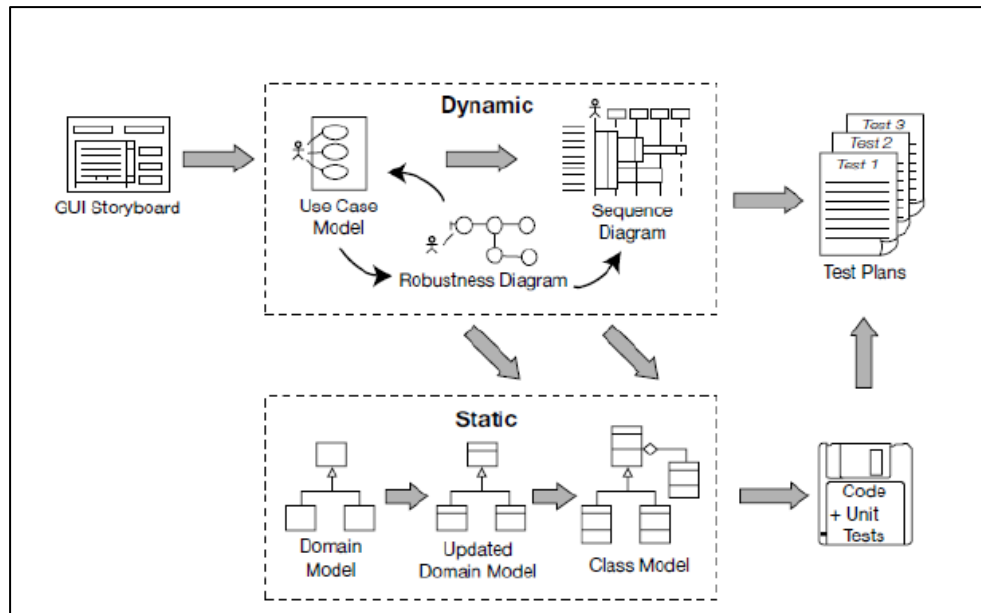
Situs web statis adalah situs web yang mempunyai laman konten yang tidak berubah-ubah. Untuk mengubah laman kontennya harus secara manual, misal dengan mengubah kode program. Situs web statis ini tidak menggunakan basis data.

#### **2. Situs Web Dinamis**

Situs web dinamis merupakan situs web yang secara susunan ditujukan untuk *update* sesering mungkin. Situs web dinamis ini menggunakan basis data, dan kebanyakan situs web dinamis rata-rata menggunakan CMS yang sudah siap pakai seperti Wordpress, Joomla, Prestashop, dan sebagainya.

### 2.3 *ICONIX Process*

Metode yang digunakan untuk membangun sistem adalah metode *ICONIX Process*. Berikut ini adalah bagan pemodelan *ICONIX Process*:



**Gambar 2.1** Bagan Pemodelan *ICONIX Process* [7]

*ICONIX Process* terdiri dari empat tahap, yaitu [7]:

#### 1. *Requirements*

##### a. *Functional Requirements*

Aktivitas dimana data-data dikumpulkan serta diolah sesuai dengan kebutuhan fungsional yang nantinya diperlukan di dalam pembangunan atau pembangunan perangkat lunak. Hal ini menjadi modal dalam pembangunan atau pembangunan perangkat lunak dimana semua kebutuhan mulai dari kebutuhan fungsional maupun non fungsional dianalisa dan diolah sehingga menjadi satu bagian kebutuhan terhadap pembangunan atau pembangunan perangkat lunak.

##### b. *Domain Modeling*

Tahapan pada bagian statis UML dimana data-data yang didapatkan berasal dari kebutuhan fungsional maupun non fungsional yang diekstrak menjadi

beberapa bagian untuk dapat dihubungkan sesuai kebutuhan perangkat lunak.

c. *GUI Storyboard*

Tahapan pembangunan tampilan antarmuka pengguna.

d. *Use Case Modeling*

Aktivitas dimana bagian dari tahapan *ICONIX Process* dilakukan sebuah pengidentifikasian terhadap aktor serta aktifitas kegiatan proses bisnis yang sedang berjalan sehingga memaparkan terhadap apa saja kegiatan yang dilakukan pengguna yang kaitannya terhadap tanggapan sistem.

2. *Analysis and Preliminary Design*

a. *Robustness Analysis*

Pengembangan dari tahapan analisa kemudian dilakukan proses tahapan desain.

b. *Update Domain Model*

Tahapan pengembangan dengan menghilangkan beberapa *class* yang *redundant* atau ambigu serta menambahkan beberapa *class* yang tidak ada serta atribut di dalam pemodelan domain jika terdapat obyek baru yang muncul.

3. *Detailed Design*

a. *Sequence Diagram*

Tahapan pemodelan *sequence diagram* dimana disusun terhadap diagram alir yang dilanjutkan dari tahapan *robustness diagram*. Berisi aktivitas yang dilakukan oleh pengguna ketika akan berinteraksi langsung dengan sistem yang dirancang atau dibangun.

b. *Update Domain Model*

Tahapan penambahan model yang didasarkan pada hasil pengembangan *sequence diagram* yang disesuaikan terhadap kebutuhan perangkat lunak.



#### 4. *Implementation*

##### a. *Coding/Unit Testing*

Tahapan dimulainya pengimplementasian sistem dimana dilakukan proses pembuatan *coding* atau proses penerjemahan setelah pengembangan model yang dirancang sebelumnya.

##### b. *Integration and Scenario Testing*

Tahapan yang dilanjutkan setelah proses pembuatan *coding* atau proses penerjemahan ke dalam tahapan pengujian (*testing*). Tujuannya, agar pengembangan model sistem yang telah dibuat dapat diukur apakah telah sesuai dengan yang diinginkan oleh pengguna maupun pengembang. Pengujian (*testing*) dapat dilakukan dengan *white box testing* atau *black box testing*.

#### 2.4 *Unified Modeling Language (UML)*

*Unified Modeling Language (UML)* merupakan suatu bahasa yang berdasarkan grafik atau gambar untuk memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan suatu sistem [2].

##### 2.4.1 *Use Case Diagram (Diagram Use Case)*

*Use case diagram* merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing – masing diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. Diagram ini penting untuk menggambarkan, merincikan [8]:

1. Sistem yaitu sesuatu yang hendak kita bangun
2. Aktor

Merupakan representasi dari seseorang atau sesuatu seperti: perangkat, sistem yang lain yang berinteraksi dengan sistem. Interaksi maksudnya adalah ketika aktor mengirimkan pesan atau menerima kepada sistem atau dari sistem serta mempertukarkan informasi.

### 3. *Use Case*

Merupakan suatu gambaran dari fungsionalitas sebuah sistem, agar pengguna sistem paham kaitannya terhadap sistem yang dibangun. Jadi seperti interaksi antara sistem dan aktor terhadap penukaran pesan serta tindakan yang dilakukan pada sistem. Contohnya:

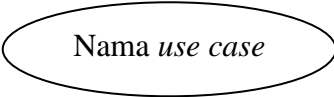
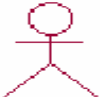

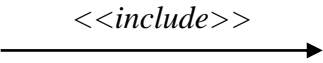
- a. *Include, use case* merupakan bagian dari *use case* yang lain.
- b. *Extend, use case* memperluas perilaku *use case* yang lain.

### 4. Relasi

Relasi antara aktor dengan *use case*.

Simbol-simbol berikut ini merupakan simbol-simbol yang digunakan pada *use case diagram* [9]:

**Tabel 2.2 Simbol *Use Case Diagram***

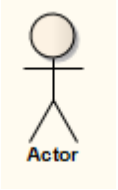



Simbol	Keterangan
<p><i>Use Case</i></p> 	Menggambarkan suatu proses kegiatan, biasa terhubung oleh pesan antar unit atau aktor.
<p>Aktor/<i>Actor</i></p> 	Mendefinisikan entitas yang melakukan interaksi terhadap sistem informasi atau unit dari suatu <i>use case</i> .
<p>Asosiasi/<i>Association</i></p> 	Mendefinisikan penghubung sebagai komunikasi aktor dan <i>use case</i> .
<p>Menggunakan/<i>Include/Use</i></p> 	Mendefinisikan relasi <i>use case</i> tambahan ke sebuah <i>use case</i> .

### 2.4.2 Robustness Diagram (Diagram Keandalan)

*Robustness diagram* merupakan *class diagram* dan *activity diagram* yang di campur sekaligus representasi dari perilaku setiap *use case* yang sudah di definisikan di awal. Interaksi antar objek dapat digambarkan dengan garis yang saling menghubungkan dengan masing-masing tindakan setiap objek [10].

Simbol-simbol berikut ini merupakan simbol-simbol yang digunakan untuk menyusun *robustness diagram*:

**Tabel 2.3 Simbol Robustness Diagram**

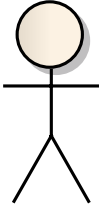


Simbol	Keterangan
 <p>Actor</p>	Pengguna sistem; pengguna dapat berarti manusia, mesin, bahkan sistem lain atau sub sistem dalam model.
 <p>Boundary</p>	Elemen berupa <i>interface</i> yang berinteraksi langsung dengan aktor.
 <p>Control</p>	Elemen yang mengatur dan mengontrol setiap aktifitas yang akan terjadi dengan basis data maupun <i>interface</i> .
 <p>Communication</p>	Elemen yang menunjukkan basis data.



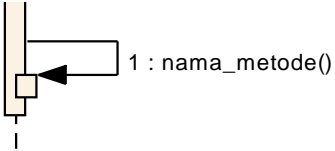

### 2.4.3 Sequence Diagram (Diagram Sekuensial)


*Sequence diagram* adalah diagram yang memodelkan skenario atau kejadian yang terjadi pada saat melakukan atau pada saat sistem dieksekusi. Biasanya *sequence diagram* mengurutkan sebuah obyek dari kiri ke kanan dan yang melakukan sebuah inisiasi biasanya diawali dari sebelah kiri hingga berurutan sampai kanan.

Simbol-simbol berikut ini merupakan simbol-simbol yang digunakan pada *sequence diagram* [9]:

**Tabel 2.4 Simbol Sequence Diagram**

Simbol	Keterangan
<p data-bbox="325 875 403 902">Aktor</p>  <p data-bbox="544 1162 624 1189"><b>Actor</b></p>	<p data-bbox="876 875 1340 1016">Menggambarkan orang yang berinteraksi terhadap sistem informasi nantinya.</p>
<p data-bbox="325 1263 592 1290">Garis Hidup/<i>Lifeline</i></p> 	<p data-bbox="876 1263 1337 1290">Menyatakan kehidupan suatu objek.</p>
<p data-bbox="325 1442 403 1469">Objek</p> <div data-bbox="384 1487 764 1563" style="border: 1px solid black; padding: 2px; width: fit-content; margin: 0 auto;"> <p data-bbox="424 1503 724 1529"><u>Nama objek : nama kelas</u></p> </div>	<p data-bbox="876 1442 1340 1525">Menyatakan objek yang berinteraksi pesan.</p>
<p data-bbox="325 1615 491 1641">Waktu Aktif</p> 	<p data-bbox="876 1615 1340 1868">Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya.</p>

Simbol	Keterangan
<p>Pesan Tipe <i>Create</i></p> <p style="text-align: center;">&lt;&lt;create&gt;&gt;</p> 	<p>Suatu objek membuat objek lain, arah panah mengarah pada objek yang dibuat.</p>
<p>Pesan Tipe <i>Call</i></p> <p style="text-align: center;">1: nama_metode()</p> 	<p>Suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,</p>  <p>Arah panah mengarah pada objek yang mempunyai operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas sesuai dengan kelas objek yang berinteraksi.</p>
<p>Pesan Tipe <i>Send</i></p> <p style="text-align: center;">1: masukan</p> 	<p>Suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.</p>

Simbol	Keterangan
Pesan Tipe <i>Return</i> 1: keluaran 	Suatu objek yang telah menjalankan suatu operasi/metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

#### 2.4.4 *Class Diagram* (Diagram Kelas)

*Class diagram* adalah susunan kelas-kelas dari suatu sistem yang menunjukkan relasi antar kelas dan penjelasan masing-masing kelas. Pada *class diagram* terdapat nama kelas, atribut, dan operasi kelas tersebut [11].

#### 2.5 *Enterprise Architect*

*Enterprise architect* adalah sebuah *platform* visualisasi untuk merancang dan membangun sebuah sistem perangkat lunak, untuk pemodelan proses bisnis, dan untuk lebih umumnya digunakan tujuan pemodelan. *Enterprise architect* adalah sebuah alat yang sangat maju yang mencakup seluruh aspek dari mulai siklus pengembangan, memberikan penelusuran secara mendetail atau penuh dari mulai awal tahap mendesain sampai mengontrol perkembangan, pemeliharaan, pengujian dan perubahan kontrol [12].

#### 2.6 CMS (*Content Management System*)

CMS (*Content Management System*) adalah sebuah perangkat lunak yang digunakan untuk memanipulasi (mengubah) isi dari suatu situs web. Umumnya, sebuah CMS terdiri dari dua elemen:

1. Aplikasi Manajemen Isi (*Content Management Application/CMA*)
2. Aplikasi Pengiriman Isi (*Content Delivery Application/CDA*)

Elemen CMA digunakan untuk membuat, mengubah, dan menghapus isi dari suatu situs web tanpa perlu mempunyai keahlian sebagai seorang *webmaster*. Sedangkan elemen CDA menggunakan dan menghimpun informasi-informasi yang sebelumnya telah diubah oleh pemilik situs web untuk memperbaharui situs web tersebut. Fitur dari sebuah CMS berbeda-beda. Walaupun begitu, kebanyakan dari perangkat lunak ini mempunyai fitur publikasi berbasis *web*, manajemen format, kontrol revisi, pembuatan indeks, pencarian, dan pengarsipan [13].

## **2.7 XAMPP Server/XAMPP**

XAMPP *server* atau yang sering disebut XAMPP adalah perangkat lunak *server* yang dapat beroperasi pada sistem operasi seperti Windows, Apple, Linux. Melalui XAMPP *server*, aplikasi situs web atau CMS seperti Wordpress, Joomla, Drupal, dan lainnya dapat dioperasikan [14].

## **2.8 Web Browser**

*Web browser* merupakan suatu perangkat lunak yang berfungsi untuk mencari suatu laman *web* yang tersimpan di dalam komputer atau untuk mengakses *internet*. *Web browser* dapat menampilkan naskah, gambar, bunyi, dan sebagainya dari suatu laman *web*. Contoh *web browser* antara lain: Mozilla firefox, Google chrome, Internet explorer, Opera, dan sebagainya [15].

## **2.9 E-commerce**

*E-commerce* atau bisa disebut dengan istilah *e-com*, dimana proses pembelian, penjualan, transfer, pertukaran suatu produk, layanan, atau informasi melalui jaringan komputer, termasuk *internet* [16].

Jenis-jenis dari *e-commerce* antara lain:

1. Bisnis ke Bisnis (*Business to Business/B2B*)  
Transaksi dilakukan baik penjualan maupun pembeli adalah organisasi atau perusahaan.
2. Perdagangan Kolaboratif (*Collaborative Commerce/C-commerce*)  
Transaksi yang dilakukan para mitra bisnis berkolaborasi secara elektronik.
3. Bisnis ke Konsumen (*Business to Consumers/B2C*)

Transaksi dilakukan perusahaan dan pembeli adalah perorangan.

4. Konsumen ke Konsumen (*Consumer to Consumers/C2C*)

Transaksi yang dilakukan seseorang menjual produk atau jasa ke orang lain

5. Konsumen ke Bisnis (*Consumers to Business/C2B*)

Transaksi yang dilakukan pelanggan memberitahukan kebutuhan atas jasa atau produk para pemasok bersaing untuk menyediakannya kepada pelanggan.

6. Perdagangan Intrabisnis (*Intraorganizational*)

Transaksi yang dilakukan pemerintah secara internal untuk memperbaiki operasinya.

7. Pemerintah ke Warga (*Government to Citizen/G2C*)

Transaksi yang dilakukan pemerintah menyediakan layanan ke para warganya.

8. Perdagangan *Mobile* (*Mobile Commerce/M-commerce*)

Transaksi yang dilakukan dalam lingkungan nirkabel.

Untuk menjalankan berbagai aplikasi *e-commerce* perusahaan membutuhkan informasi, infrastruktur, dan layanan pendukung yang tepat. *E-commerce* didukung oleh infrastruktur seperti perangkat keras, perangkat lunak, jaringan, multimedia, dan juga area pendukung lainnya seperti [16]:

1. Orang, yaitu para penjual, pembeli, perantara, ahli sistem informasi, karyawan, dan peserta lainnya.
2. Kebijakan publik, berbagai isu hukum dan kebijakan serta peraturan lainnya seperti perlindungan atas privasi dan perpajakan yang ditetapkan oleh pemerintah.
3. Pemasaran dan periklanan.
4. Pembayaran hingga pengiriman pesanan.
5. Kemitraan bisnis.



## 2.10 *Black Box Testing*

*Black box testing* menekankan pada sejauh mana fungsionalitas sistem berjalan dengan baik sesuai dengan yang diinginkan oleh pengguna dan pengembang.

Pada *black box testing* ini, setidaknya terdapat empat jenis pengujian. Keempat jenis pengujian tersebut meliputi [17]:

### 1. Pengujian *Interface*

Pengujian ini bertujuan untuk mengetahui fungsionalitas dari setiap elemen *interface* yang ada di setiap halaman pada sistem. Elemen tersebut berupa tombol (*button*) yang menjalankan aksi sesuai yang diharapkan oleh pengguna dan pengembang.

### 2. Pengujian Fungsi Dasar Sistem

Pengujian ini bertujuan untuk mengetahui sejauh mana kinerja dari setiap fungsi dasar sistem yang ada di dalam sistem. Fungsi-fungsi ini dalam penerapannya membentuk satu atau sejumlah modul yang dapat digunakan oleh pengembang maupun pengguna.

### 3. Pengujian *Form Handle*

Pengujian ini bertujuan untuk mengetahui seperti apa dan sejauh mana respon sistem terhadap *input* yang diberikan oleh pengguna.

### 4. Pengujian Keamanan

Pengujian ini bertujuan untuk mengetahui sejauh mana tingkat keamanan yang dimiliki oleh sistem untuk dapat memberikan kenyamanan kepada pengguna.

## 2.11 Hubungan *ICONIX Process* dengan Sistem Informasi

Pengertian dari *ICONIX Process* adalah salah satu model dari rekayasa perangkat lunak yang dapat digunakan untuk pengembangan sebuah perangkat lunak. Pendekatan dari *ICONIX Process* berada di antara RUP (*Rational Unified Process*) yang sangat luas dan XP (*eXtreme Programing*) yang sangat sederhana. Maksudnya, *ICONIX Process* menggunakan *use case*, seperti RUP tetapi tanpa banyak pentabelan. Proses *ICONIX* juga relatif kecil dan sederhana seperti XP tetapi tidak mengabaikan analisa dan desain.

*ICONIX Process* yaitu suatu metode dimana tidak terlalu banyak membahas pada analisa, desain maupun implementasi program atau sistemnya. Namun lebih melihat kepada kebutuhan pengguna serta menyederhanakan proses tersebut, sehingga proses pengembangan perangkat lunak akan menjadi lebih efisien.

Menurut ahli lainnya, *ICONIX Process* pada umumnya diadopsi dari metodologi pengembangan yang telah dikenal luas, seperti XP (*eXtreme Programming*), RUP (*Rational Unified Process*), MSF (*Microsoft Solutions Framework*), AUP (*Agile Unified Process*), dan sebagainya. Metodologi pengembangan aplikasi menyediakan pedoman-pedoman (mekanisme yang perlu dilakukan pada tiap tahap siklus pengembangan, berbagai *best practice*, mekanisme uji aplikasi, standar pemrograman, standar analisa dan desain, dan lain-lain) bagi para pengembang sehingga diharapkan dapat bekerja lebih terstruktur, efisien, dan efektif, serta menghasilkan aplikasi yang lebih berkualitas.

Tiap metodologi pengembangan aplikasi memiliki karakteristik tersendiri dengan kelebihan dan kekurangannya dibanding metodologi yang lainnya. Karena karakteristik yang dimiliki belum tentu tepat sepenuhnya, perusahaan-perusahaan memiliki kebebasan untuk melakukan *customisasi* proses pengembangan dari metodologi yang dipilih, yang disesuaikan dengan karakteristik perusahaan di dalam mengembangkan aplikasi. Aktivitas-aktivitas di setiap tahap proses pengembangan yang telah ditentukan tersebut umumnya disesuaikan lagi dengan aplikasi yang dikembangkan.

Bermula dari RUP yaitu metodologi pengembangan perangkat lunak, yang diformulasikan oleh Rational Software Corporation (sekarang menjadi salah satu divisi IBM), yang menggunakan UML (*Unified Modeling Language*) sebagai bahasa pemodelan selama periode pengembangan dan *iterative incremental* sebagai model siklus pengembangan perangkat lunak. Model ini membagi suatu sistem aplikasi menjadi beberapa komponen sistem dan memungkinkan para pengembang aplikasi untuk menerapkan metoda *iterative* (analisa, desain, implementasi, dan pengujian) pada tiap komponen. Dengan menggunakan model ini, RUP membagi tahapan pengembangan perangkat lunaknya ke dalam 4 fase sebagai berikut:

1. *Inception*, merupakan tahap untuk mengidentifikasi sistem yang akan dikembangkan. Aktivitas yang dilakukan pada tahap ini antara lain mencakup analisa sistem eksisting, perumusan sistem target, penentuan arsitektur global target, identifikasi kebutuhan, perumusan persyaratan (fungsional, performansi, keamanan, GUI, dan lain-lain), perumusan kebutuhan pengujian (level unit, integrasi, sistem, performansi, fungsionalitas, keamanan, dan lain-lain), pemodelan diagram UML (diagram *use case* dan *activity*), dan pembuatan dokumentasi.
2. *Elaboration*, merupakan tahap untuk melakukan desain secara lengkap berdasarkan hasil analisa di tahap *inception*. Aktivitas yang dilakukan pada tahap ini antara lain mencakup pembuatan desain arsitektur subsistem (*architecture pattern*), desain komponen sistem, desain format data (protokol komunikasi), desain database, desain antarmuka/tampilan, desain peta aliran tampilan, penentuan *design pattern* yang digunakan, pemodelan diagram UML (diagram *sequence*, *class*, *component*, *deployment*, dan lain-lain), dan pembuatan dokumentasi.
3. *Construction*, merupakan tahap untuk mengimplementasikan hasil desain dan melakukan pengujian hasil implementasi. Pada tahap awal *construction*, ada baiknya dilakukan pemeriksaan ulang hasil analisa dan desain, terutama desain pada domain perilaku (diagram *sequence*) dan domain struktural (diagram *class*, *component*, *deployment*). Apabila desain yang dibuat telah sesuai dengan

analisa sistem, maka implementasi dengan bahasa pemrograman tertentu dapat dilakukan. Aktivitas yang dilakukan pada tahap ini antara lain mencakup pengujian hasil analisa dan desain (misal menggunakan *class responsibility collaborator* untuk kasus pemrograman berorientasi obyek), pendataan kebutuhan implementasi lengkap (berpedoman pada identifikasi kebutuhan di tahap analisa), penentuan *coding pattern* yang digunakan, pembuatan program, pengujian, optimasi program, pendataan berbagai kemungkinan pengembangan atau perbaikan lebih lanjut, dan pembuatan dokumentasi.

4. *Transition*, merupakan tahap untuk menyerahkan sistem aplikasi ke konsumen (*roll-out*), yang umumnya mencakup pelaksanaan pelatihan kepada pengguna dan *testing beta* aplikasi terhadap ekspektasi pengguna.

Kemudian prinsip-prinsip yang ada di dalam pembentukan *ICONIX Process* yaitu:

1. Prinsip

Prinsip-prinsip yang membentuk dasar *ICONIX Process* didasarkan pada nilai-nilai yang baru saja dijelaskan dan ditujukan untuk mendorong keputusan dalam suatu proyek pengembangan sistem. Prinsip-prinsip ini dimaksudkan untuk menjadi lebih konkret daripada nilai-nilai dan lebih mudah diterjemahkan untuk panduan dalam situasi praktis.

2. Umpan balik

*EXtreme Programming* (XP) melihat umpan balik yang paling berguna jika hal itu dilakukan dengan cepat dan mengungkapkan bahwa waktu antara aksi dan umpan balik yang sangat penting untuk belajar dan membuat perubahan. Tidak seperti metode pengembangan sistem tradisional, kontak dengan pelanggan lebih sering terjadi pada iterasi. Pelanggan memiliki pandangan yang jelas ke dalam sistem yang sedang dikembangkan. Ia dapat memberikan umpan balik dan mengarahkan pembangunan sesuai dengan kebutuhan.

3. *Unit test* juga berkontribusi terhadap prinsip umpan balik yang cepat. Ketika menulis kode, pengujian unit langsung menyediakan umpan balik tentang bagaimana sistem bereaksi terhadap perubahan yang telah dibuat. Jika,

misalnya, perubahan mempengaruhi bagian dari sistem yang tidak dalam lingkup para *programmer* yang membuat mereka, bahwa *programmer* tidak akan melihat cacat. Ada kemungkinan besar *bug* akan muncul ketika sistem sedang dalam produksi.

#### 4. Mengasumsikan kesederhanaan

Ini tentang memperlakukan setiap masalah sebagai jika solusi yang “sangat sederhana”. Metode pengembangan sistem tradisional mengatakan untuk merencanakan masa depan dan untuk kode untuk usabilitas. *EXtreme Programming* (XP) menolak ide-ide ini.

Pendukung *EXtreme Programming* (XP) bilang bahwa membuat perubahan besar sekaligus tidak bekerja. *EXtreme Programming* (XP) berlaku perubahan tambahan: misalnya, sistem mungkin rilis kecil setiap tiga minggu. Ketika banyak langkah-langkah kecil yang dibuat, pelanggan memiliki lebih banyak kontrol atas proses pembangunan dan sistem yang sedang dikembangkan.

#### 5. Merangkul perubahan

Prinsip perubahan merangkul tentang tidak bekerja melawan perubahan tetapi memeluk mereka. Sebagai contoh, jika pada salah satu pertemuan berulang-ulang tampak bahwa persyaratan pelanggan telah berubah secara dramatis, programer adalah untuk merangkul rencana ini dan persyaratan baru untuk iterasi berikutnya.

Keunggulan *ICONIX Process* dibandingkan dengan metodologi lain yaitu *ICONIX Process* menggunakan UML secara elegan, tidak berlebihan. Selain itu, *ICONIX Process* menggunakan robustness diagram untuk melakukan analisa kehandalan. Analisa kehandalan untuk mengetahui apakah terdapat objek-objek baru yang sebelumnya tidak teridentifikasi atau objek-objek baru muncul.

Sedangkan kelemahan *ICONIX Process* yaitu proses pemodelan yang dipicu oleh *use case* jadi penentuan model dibangun sejak awal.

Manfaat *ICONIX Process* terhadap sistem membuat penggunaan UML menjadi efisien karena tetap terfokus pada *requirement*. Sehingga *ICONIX Process* menjadi suatu metode dimana tidak terlalu banyak membahas pada analisa, desain maupun implementasi program atau sistemnya. Namun lebih melihat kepada kebutuhan pengguna serta menyederhanakan proses tersebut, sehingga proses pengembangan perangkat lunak akan menjadi lebih efisien.

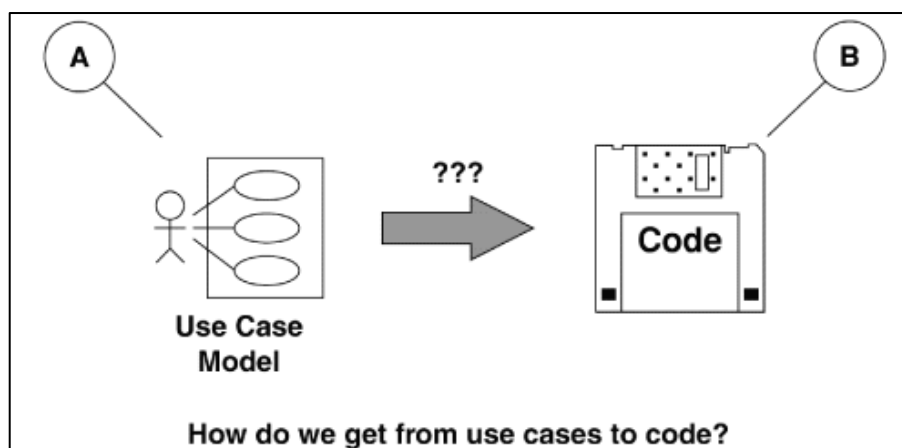
Mengapa pemilihan *ICONIX Process*?, pemilihan *ICONIX Process* sebagai metode pengembangan sistem tidak terlepas dari beberapa fitur yang dimiliki oleh *ICONIX Process* antara lain [3]:

1. *ICONIX Process* merupakan proses yang dipicu oleh *use case* (*use case driven*). Pada *ICONIX Process*, *use case* yang ditentukan sejak awal pengembangan menjadi dasar dalam menentukan model dan perilaku dari sistem yang dibangun.
2. *ICONIX Process* merupakan proses yang berorientasi pada arsitektur (*architecture-centric*). *ICONIX Process* berkonsentrasi pada desain model sebagai arsitektur sistem. Model ini terdiri dari model statis yang akan menjadi kode-kode dan model dinamis yang menggambarkan perilaku sistem.
3. *ICONIX Process* merupakan metode yang iteratif dan bertahap (*iterative-incremental*). Banyak iterasi yang terjadi pada saat menentukan model ranah (*domain model*), saat mengidentifikasi dan menganalisa *use case*, dan iterasi-iterasi lain yang terjadi seiring berjalannya siklus hidup pengembangan sistem. Model statis yang dihasilkan terus diperbaiki secara bertahap dengan bantuan model dinamis (terdiri dari *use case*, *robustness analysis*, dan *sequence diagram*).
4. *ICONIX Process* menawarkan penggunaan UML yang tidak berlebihan bahkan cenderung minimalis karena hanya terdiri beberapa langkah yang dianggap perlu dan telah cukup untuk melakukan analisa berbasis objek.

5. *ICONIX Process* memberikan keterjejukan (*traceability*) yang cukup tinggi. Merujuk kembali kepada kebutuhan awal dapat dilakukan dengan berbagai cara yang mudah pada setiap tahap pengembangan. Keterjejukan ini juga tampak pada kenyataan bahwa setiap objek dapat dilacak langkah demi langkah, dari analisa menjadi desain.

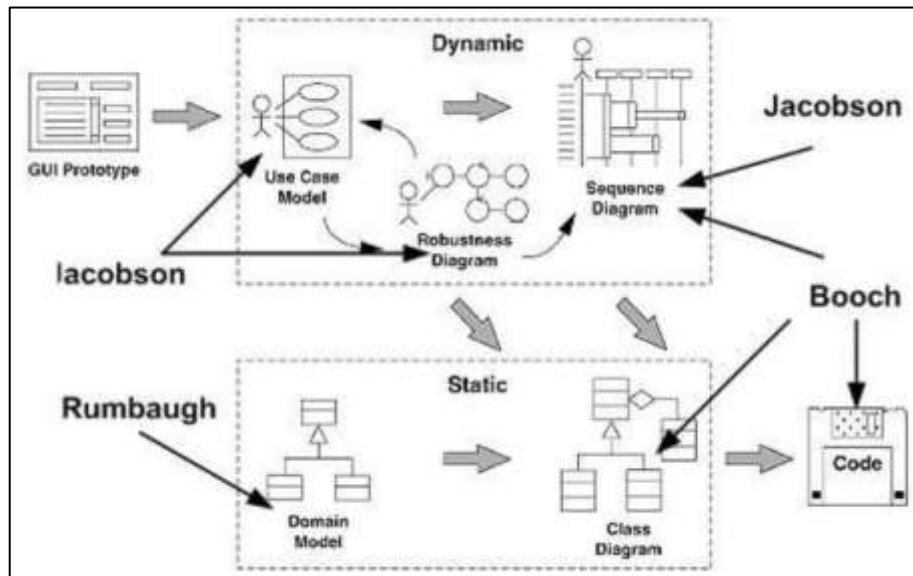
*ICONIX Process* terletak ditengah-tengah antara RUP (*Rational Unified Process*) yang besar dan XP (*eXtreme Programming*) yang sangat kecil. *ICONIX Process* merupakan *use case driven* seperti RUP, tetapi tidak berbelit-belit seperti yang dihasilkan oleh RUP. *ICONIX Process* juga kecil dan singkat seperti XP, tetapi tidak meninggalkan analisa dan desain seperti yang dilakukan XP [3].

Tujuan utama dari *ICONIX Process* adalah bagaimana mewujudkan *use case* yang telah disusun menjadi kode seperti pada Gambar 2.2. Titik A adalah ide-ide tentang apa yang harus dilakukan sistem dengan cara menggambarannya dalam *use case diagram*. Titik B menunjukkan potongan-potongan kode yang komplit, telah diuji, dan bisa mengerjakan apa yang disebutkan pada *use case*. *ICONIX Process* berusaha menjawab proses-proses yang berupa tanda tanya di antara titik A dan titik B [3].



Gambar 2.2 Bagaimana Mengubah *Use Case* Menjadi Kode [3]

Integrasi dari elemen Booch, Rumbaugh, dan Jacobson memberikan gambaran yang menyeluruh dari pengembangan perangkat lunak berorientasi objek seperti pada Gambar 2.3. Tampak bahwa struktur dinamis model terdiri dari *use case diagram*, *robustness diagram*, dan *sequence diagram* dan struktur statis model yang terdiri dari adalah *domain model* dan *class diagram* [3].



**Gambar 2.3 ICONIX Process dan Kontribusi The Three Amigos [3]**

### 1. Pemodelan Ranah [3]

Pemodelan ranah (*domain modeling*) menjadi dasar dari bagian model statis yang dibangun. Pemodelan ranah adalah suatu pekerjaan yang bertugas menemukan objek-objek (berupa kelas-kelas) yang menjadi representasi dari benda-benda dan konsep-konsep dalam dunia nyata.

Pemodelan ini dimulai dengan melakukan abstraksi model dalam dunia nyata yaitu berupa objek-objek konseptual yang turut berpartisipasi dalam sistem yang dibangun. Hal ini dapat dilakukan dengan menyoroti dan mencermati kata benda pada dokumen problem statement sehingga ditemukan objek-objek dalam model ranah. Proses ini kemudian dilanjutkan dengan melakukan identifikasi relasi yang generalisasi dan asosiasi yang mungkin ada di antara objek-objek tersebut.



Dari proses ini akan dihasilkan model ranah sebagai awal dari sistem yang dibangun.

## 2. Pemodelan *Use case* [3]

Pemodelan *use case* diperlukan untuk menjawab pertanyaan dasar dari pengembangan yaitu: “Apa yang akan dilakukan oleh pengguna sistem?” Pemodelan *use case* diperlukan untuk menangkap kebutuhan-kebutuhan pengguna terhadap sistem yang dibangun dengan menggambarkan secara detail seluruh skenario yang akan dilakukan pengguna terhadap sistem dan tanggapan yang akan diberikan oleh sistem.

Proses ini dimulai dengan menemukan aktor-aktor yang terlibat dan aktivitas-aktivitas yang dilakukannya dengan cara mencermati dokumen *problem statement* atau dengan bantuan seseorang yang memahami ranah persoalan yang dihadapi kemudian membuat beberapa usulan *use case* ke dalam *use case diagram*.

*Use case* yang telah berhasil diidentifikasi perlu diwujudkan menjadi sebuah dokumen *use case* sebagai sarana untuk melakukan komunikasi antar tim pengembang sekaligus sebagai dokumentasi dari sistem yang dibangun.

Sedangkan *Unified Modeling Language* atau UML adalah standar untuk memvisualisasikan, membentuk, membangun dan mendokumentasikan suatu sistem yang merupakan himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya.

*ICONIX Process* sangat menganjurkan untuk membuat prototipe aplikasi atau setidaknya desain antarmuka pengguna seiring dengan penyusunan dokumen *use case*. Hal ini akan sangat membantu untuk menyusun *use case* karena teks dalam dokumen *use case* akan cocok dengan elemen GUI (*Graphical User Interface*) pada antar muka pengguna baik dari sisi deskripsi elemen GUI tersebut maupun dari sisi tanggapan sistem sebagai akibat aksi sang aktor. Untuk menyusun dokumen *use case*, berikut ini adalah langkah-langkah yang dianjurkan *ICONIX Process*.

- a. Membuat templat yang berisi *basic course* dan *alternate course*.
- b. Ajukan pertanyaan “Apa yang akan terjadi?” yang akan memulai *basic course*.
- c. Ajukan kembali pertanyaan “Lalu apa lagi yang akan terjadi?” dan terus ajukan pertanyaan ini sampai semua detail *basic course* teridentifikasi.
- d. Ajukan pertanyaan “Hal lain apa yang mungkin terjadi?” yang akan menjadi identifikasi dari *alternate course*.

## 2. Analisa Kehandalan [3]

Analisa kehandalan (*robustness analysis*) diperlukan untuk mengetahui objek-objek apa saja yang terlibat dalam setiap *use case*. Analisa kehandalan (*robustness analysis*) diperkenalkan secara informal oleh Ivar Jacobson pada tahun 1991. Proses ini dilakukan dengan cara menganalisa teks *use case* dan melakukan identifikasi objek-objek yang akan berpartisipasi kemudian melakukan klasifikasi terhadap objek tersebut menjadi tiga tipe objek yaitu:

### a. *Boundary object*

Objek yang digunakan aktor sebagai antarmuka untuk berkomunikasi dengan sistem.

### b. *Entity object*

Objek ini biasanya berupa objek yang berasal dari ranah model.

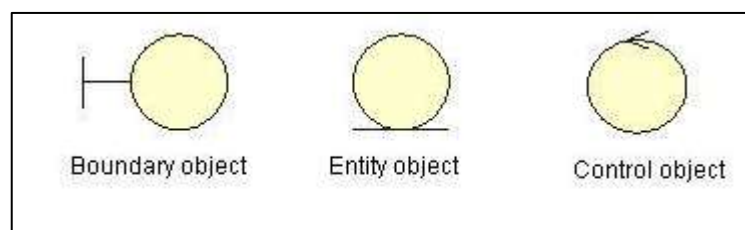
### c. *Control object*

Objek yang menjadi “perekat” antara *boundary object* dan *entity object*.

Analisa kehandalan (*robustness analysis*) berguna untuk memperkecil celah antara analisa (*what*) dan desain (*how*). Pada *ICONIX Process*, analisa kehandalan mempunyai beberapa peranan penting yaitu:

- a. Uji kelayakan – Analisa kehandalan membantu untuk meyakinkan bahwa dokumen *use case* yang disusun sudah benar dan menghindari perilaku sistem yang tidak sesuai.

- b. Uji kelengkapan - Analisa kehandalan membantu meyakinkan bahwa *use case* telah meliputi semua aksi alternatif (*alternate courses*).
- c. Menemukan objek-objek baru – Analisa kehandalan membantu menemukan objek-objek yang mungkin terlewatkan pada saat pemodelan ranah.
- d. Sebagai desain awal – Analisa kehandalan menjadi dasar untuk desain awal dari sistem yang dibangun dengan memperkecil celah antara analisa dan desain seperti yang telah disebutkan dimuka.



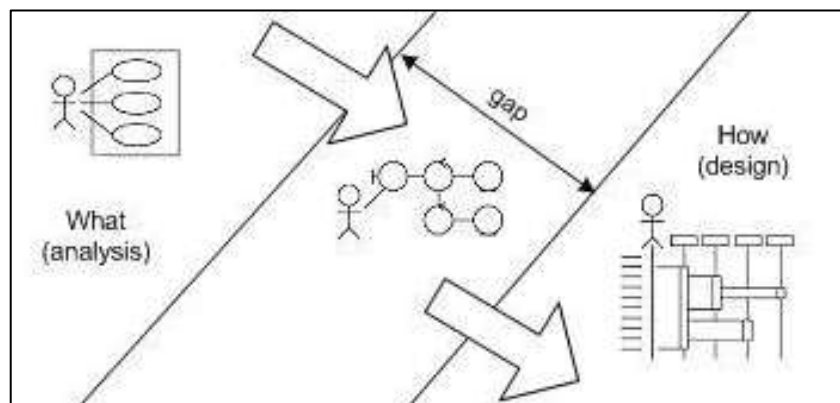
**Gambar 2.4 Simbol dalam *Robustness Diagram* [3]**

Gambar 2.4 menunjukkan simbol-simbol yang digunakan untuk merepresentasikan tipe-tipe objek ini. Walaupun *three amigos* menyadari adanya teknik analisa ini tetapi mereka tidak memasukkannya sebagai bagian UML melainkan hanya menjadikannya sebagai stereotipe suatu objek.

Pada *ICONIX Process*, teknik sederhana ini menjadi sarana penghubung utama antara celah analisa (*what*) dan desain (*how*) seperti yang tampak pada Gambar 2.5.

Langkah-langkah untuk melakukan analisa kehandalan pada sebuah *use case* adalah sebagai berikut:

- a. Menelusuri kalimat demi kalimat, teks pada dokumen *use case*.
- b. Menambahkan aktor yang terlibat pada *use case* sebagai titik awal.



**Gambar 2.5 Robustness Diagram sebagai Jembatan antara Analisa dan Desain [3]**

- c. Menambahkan antar muka pengguna seperti halaman web, tampilan layar, menu, kotak dialog dan sebagainya sebagai *boundary object*.
- d. Menambahkan *controller* bagi setiap aturan-aturan bisnis dan aktivitas-aktivitas yang ada.
- e. Menambahkan *entity object* bagi setiap objek konseptual dalam model ranah.
- f. Membuat garis penghubung yang merepresentasikan komunikasi yang terjadi antar objek-objek tersebut. [3]

Hubungan antara *ICONIX Process* dengan sistem informasi bisa dibilang sangat berkaitan dan menunjang ataupun mendukung proses penghasilan informasi maupun sebagai alat penunjang terbentuknya pengambilan keputusan yang baik. Sistem informasi merupakan sekumpulan sistem yang saling terintegrasi sehingga menghasilkan informasi. Untuk menghasilkan informasi tersebut dibutuhkan sebuah metode di dalam pembangunan maupun pengembangannya, seperti cara bagaimana sebuah sistem dapat saling terintegrasi dan menghasilkan informasi dengan baik. Tentunya sangat dibutuhkan proses dan metode yang tepat tetapi efisien. Berangkat dari sebuah metode *ICONIX Process* yang telah dijelaskan di atas termasuk kenapa memilih metode *ICONIX Process* menjadi lebih efektif karena *ICONIX Process* memiliki kesederhanaan tidak hanya dalam pembentukan desain tetapi juga di dalam informasi yang yang dihasilkan sangatlah mengidentifikasi keinginan user atau mengetahui keinginan pelanggan. Sebuah sistem informasi yang baik tentu diharuskan memiliki gambaran serta informasi

yang sesuai dengan kebutuhan penggunanya. Sistem informasi merupakan kesatuan komponen yang saling berhubungan yang mengumpulkan data, memproses data menjadi informasi, serta menyimpan dan menyalurkan informasi untuk mendukung pengambilan keputusan [18]. Metode *ICONIX Process* membantu pengguna untuk dapat dilakukan pembentukan serta pengembangan yang baik terhadap sistem informasi yang nantinya akan dihasilkan.