SOFTWARE DESIGN PATTERN PADA SITUS-SITUS *E-COMMERCE*

Ahmad Hanafi

Program Studi Teknik Informatika
STMIK Jenderal Achmad Yani Yogyakarta

hanafi@mti.ugm.ac.id; ahmadhanafi@live.com

Abstrak

Situs E-Commerce adalah situs yang paling produktif dibanding dengan situs-situs yang lain. Waktu merupakan elemen penting, semakin cepat sebuah situs online dan beroperasi semakin cepat keuntungan pemilik usaha. Untuk itu pengembangan situs e-Commerce diharapkan singkat. Namun waktu pengembangan yang singkat menghasilkan situs yang kurang optimal. Diperlukan sebuah metode yang cepat namun mampu menghasilkan rancangan e-commerce yang baik.

Diperlukan sebuah pendekatan baru yang cepat namun mampu mengidentifikasi dengan tepat apa yang dibutuhkan oleh situs-situs e-commerce untuk kemudian dibangun juga dengan cepat. Menurut Yakoub (2003) metode software design pattern mampu menjawab hal tersebut.

Design pattern yang spesifik teridentifikasi dalam web portal e-marketplace adalah (Hanafi, 2011): Product page, Carousel, Shopping cart, Product comparison, Rating an object, Page grids, Purchase process. Dengan teridentifikasinya design patten yang ada, maka proses pengembangan situs-situs e-commerce akan lebih cepat dengan memotong proses spesifikasi.

Kata Kunci: *e-commerce*, *software development*, *design pattern*, YUI.

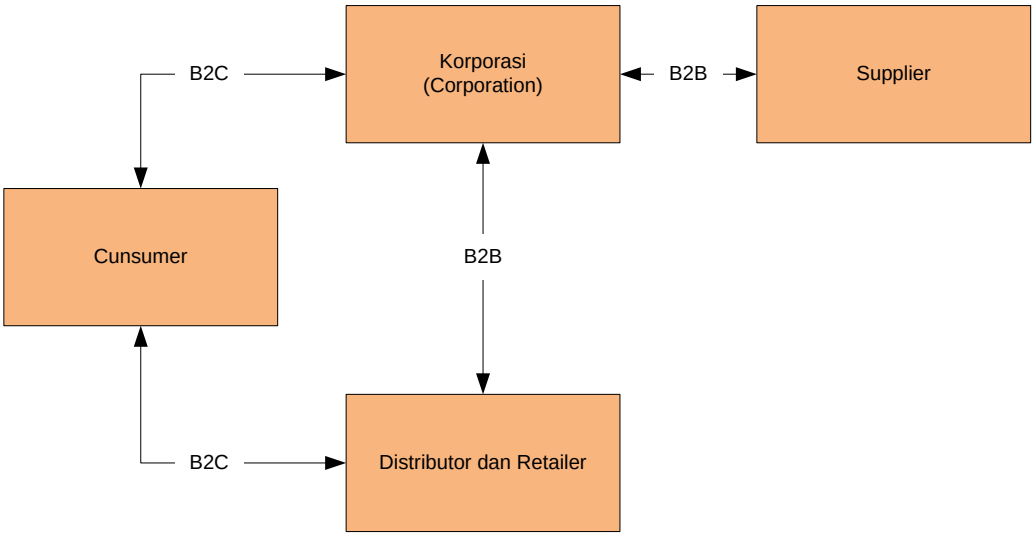
1. Pendahuluan

E-commerce adalah segala macam kegiatan bisnis yang dilakukan secara *online*, misalnya menjual dan membeli barang atau jasa dari Internet (Netlingo, 2012). Situs *e-commerce* adalah situs yang paling produktif dibanding dengan situs-situs yang lain. Waktu merupakan elemen penting, semakin cepat sebuah situs *online* dan beroperasi semakin cepat keuntungan pemilik usaha. Untuk itu pengembangan situs *e-commerce* diharapkan singkat. Namun waktu pengembangan yang singkat menghasilkan situs yang kurang optimal, hal ini dikarenakan proses pengembangan sistem yang seharusnya ada, dihilangkan.

Diperlukan sebuah pendekatan baru yang cepat namun mampu mengidentifikasi dengan tepat apa yang dibutuhkan oleh situs-situs *e-commerce* untuk kemudian dibangun juga dengan cepat. Menurut Yakoub (2003) metode *software design pattern* mampu menjawab hal tersebut.

Bentuk situs *e-commerce* bervariasi. Situs yang diperuntukkan untuk menjual barang langsung pada pelanggan disebut *e-commerce* B2C (*Business to Customer*). Situs *e-commerce* yang diperuntukkan untuk menjual barang atau jasa untuk sesama entitas bisnis disebut sebagai B2B (*Business to Business*).

Dengan melihat banyaknya situs-situs *e-commerce* yang ada, maka diketemukan beberapa persamaan-persamaan diantara situs-situs *e-commerce* tersebut. Kesamaan-kesamaan tersebut bisa digunakan untuk mempersingkat proses pengembangan sistem yaitu pada tahap identifikasi kebutuhan informasi atau *requirement specification*.



Gambar 1: Bagan B2C dan B2B

Proses pengembangan perangkat lunak melewati dua macam cara yaitu proses yang bersifat spesifikasi teknis dan permasalahan perancangan perangkat lunak. Dua hal tersebut memiliki porsi yang besar dalam alokasi pengembangan perangkat lunak. Proses spesifikasi yang baik adalah proses yang mampu mengidentifikasi kebutuhan secara efisien dan hasil spesifikasi mampu menghadapi perubahan dan kebutuhan yang akan datang dalam kontek pembuatan dan pengembangan sebuah sistem. Dengan mengurangi waktu yang dibutuhkan untuk spesifikasi dan perancangan maka perangkat lunak akan lebih cepat selesai.

Pendekatan *design pattern* dapat mempercepat proses pengembangan perangkat lunak dan juga menghindarkan permasalahan dikemudian hari. "*Design patterns capture many of the structures that result from refactoring*" (Gamma et al. 1995, pp. 354), *Design pattern* mengurangi kemungkinan untuk melakukan aktifitas *refactoring*.

2. *Software Design Pattern*

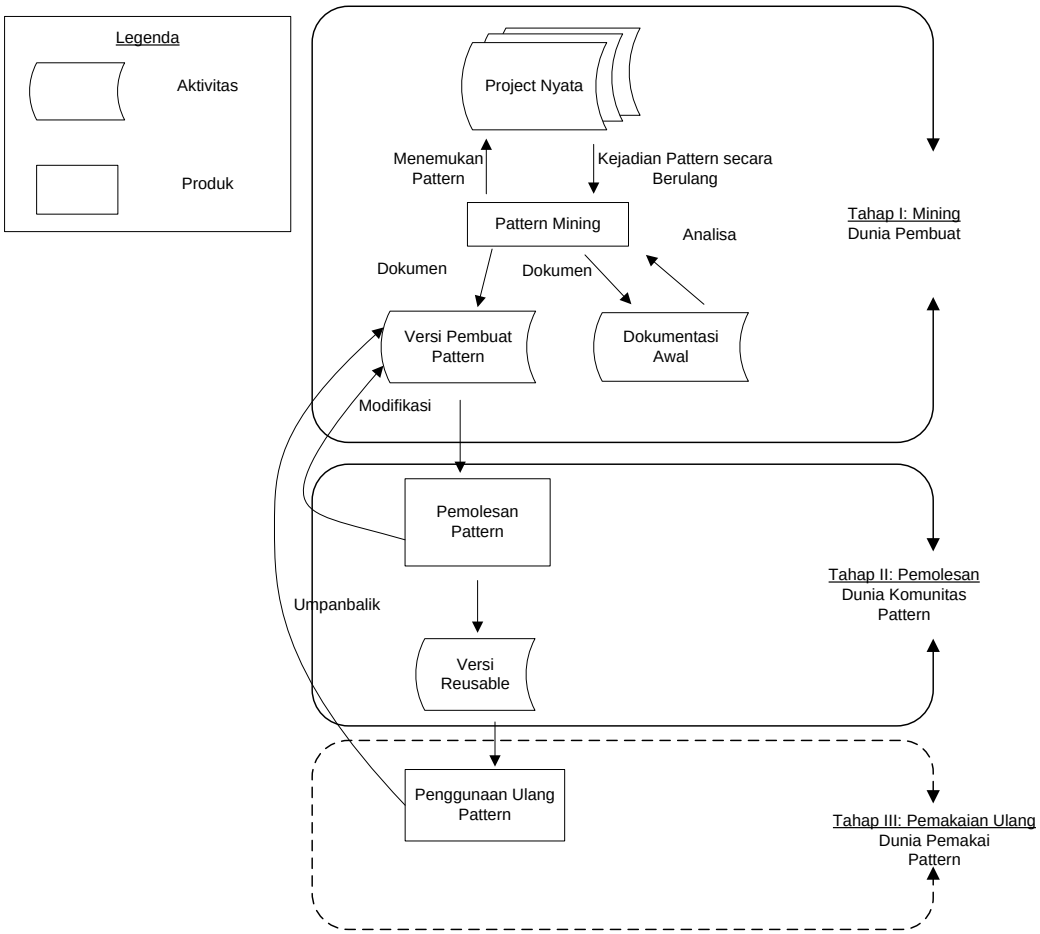
Design Pattern adalah sebuah solusi yang berulang-ulang untuk permasalahan yang dijumpai berulang kali di dalam sebuah pengembangan aplikasi di dunia nyata (dofactory, 2009). Pendekatan ini akan membuat sebuah

gambaran tentang sebuah kumpulan pola dalam kode program dan membuat kategorisasi dari hal tersebut. Kategorisasi tersebut menghasilkan sebuah nama *design pattern*. Selain itu, sebuah *design pattern* bisa dibentuk dari kesamaan fungsi dan kebutuhan sistem.

Tabel 1: Perbandingan pendekatan konvensional dengan pendekatan *design pattern*

Aspek	Tanpa *Design Pattern*	*Design Pattern*
Perancangan	Perancangan dari awal	perancangan tidak perlu dari awal
Identifikasi Permasalahan	Melalui proses pengujian berlanjut	Permasalahan dapat diidentifikasi dari pengembangan perangkat lunak serupa, sehingga dapat dihindari
Waktu pengembangan	Lama	Cepat
Pihak yang terlibat	Banyak	Sedikit
Pengembangan ke depan	Tidak memiliki pengaruh	Semakin mudah
Kemungkinan Terjadi *Refactoring*	Besar	Kecil

Proses *design pattern* tersebut dibagi menjadi tiga tahap, antara lain (Yacoub, 2003): Penambangan, Pemolesan dan Pemakaian ulang.



Gambar 2: Tahap dokumentasi *Software Design Pattern*

3. *Design Pattern* dalam *E-Commerce*

Pengamatan dilakukan pada dua situs di bawah ini. Pengamatan dilakukan dengan memperhatikan fitur-fitur dan proses bisnis yang ada pada dua situs berikut ini. Situs yang pertama adalah Urbanesia. Urbanesia merupakan sebuah situs *web* 2.0 yang memberikan daftar direktori yang berisi tempat usaha yang tersebar di kota Jakarta dan sekitarnya. Situs yang kedua adalah Tokopedia. Situs ini adalah salah satu portal belanja terkemuka di Indonesia dan telah mengadopsi *web* 2.0 dan telah berhasil menjaring 3.131 toko *online* yang menjajakan 32.032 item barang.

Design pattern yang umum digunakan oleh *e-marketplace* juga terdapat dalam situs-situs *web* yang lain. Namun beberapa *design pattern* yang spesifik teridentifikasi dalam *web* portal *e-marketplace* adalah sebagai berikut (Hanafi, 2011):

1. *Product page*
2. *Carousel*
3. *Shopping cart*
4. *Product comparison*
5. *Rating an object*
6. *Page grids*
7. *Purchase process*

4. Pembahasan *Design Pattern*

4.1 *Product Page*

Nama *design pattern*: *Product Page*

Definisi *design pattern*:

Sebuah halaman yang terdiri dari gambar dan deskripsi singkat tentang suatu produk tertentu. Biasanya terdiri dari banyak produk dalam satu halaman dan berisi kumpulan tautan yang relevan.

Masalah yang diselesaikan:

Pengguna ingin mengetahui detail dari sebuah produk dalam rangka memilih sebuah produk.

Kapan menggunakan *pattern* ini?

- Digunakan untuk menampilkan informasi dari sebuah produk di sebuah *webshop*, situs *web* merk, situs *web* perbandingan produk atau situs *web* lain yang menawarkan produknya.

- Digunakan untuk menampilkan produk yang dijual secara detail, mendekati tampilan fisik sebenarnya.

Bagaimana solusinya?

Tampilkan produk tersebut dan informasi yang berhubungan dengan kelompok produk tersebut dalam bentuk potongan informasi. Berikan pilihan berupa tautan ke produk-produk lain yang relevan.

Alasan pemakaian *pattern* ini:

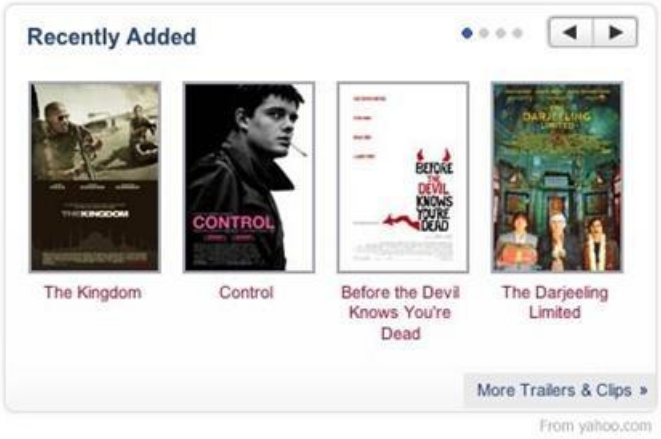
Tujuan utama pemakaian pattern ini adalah mengubah pengunjung situs menjadi pelanggan. Pakai halaman produk sebagai media untuk merayu pengguna untuk mengambil keputusan untuk membeli satu atau lebih produk yang dijual.

Aksesibilitas:

Dengan menampilkan *product page* pada awal *homepage* maka pengunjung akan disajikan deretan tampilan dari produk yang dijual, pengunjung mendapat kesan bisnis inti dari situs, bahwa situs ini serius dalam menjual produk.

Sumber Pustaka: ui-pattern.com

4.2 *Carousel*



Gambar 3: Implementasi *design pattern carousel*

Nama *design pattern*: *Carousel*

Definisi *design pattern*:

Sebuah bagian dari halaman (biasanya 1/3 bagian halaman muka) yang memiliki fungsi untuk memanfaatkan momentum tampilan layar yang sebentar untuk menampilkan sejumlah informasi yang penting. Selain itu dalam waktu yang sama menyediakan pengalaman yang menarik dalam pemilihan obyek,

dengan jalan menampilkan sebuah set obyek dalam bentuk gambar dengan batasan waktu tampil terbatas sehingga mengakomodir semua objek yang ingin ditampilkan. Merencanakan urutan obyek, menyediakan sarana yang berbeda untuk memutar dan berkomunikasi secara gamblang elemen yang sedang disorot adalah penting agar desain berhasil.

Masalah yang diselesaikan:

Pengguna menginginkan untuk dapat menjelajah sebuah set barang atau produk dan kemungkinan untuk memilih salah satu di antaranya.

Kapan menggunakan *pattern* ini?

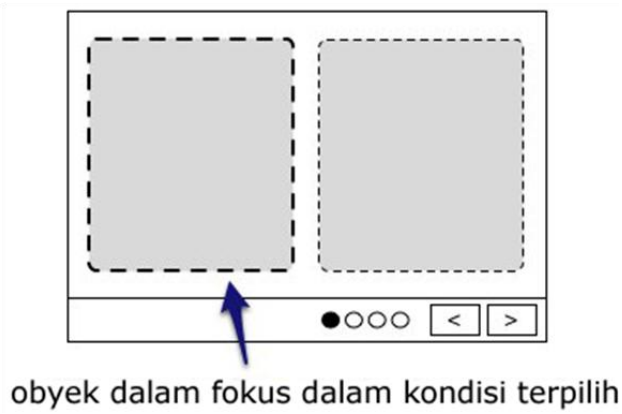
Ketika model pemikiran pengguna terikat pada serangkaian topik atau serangkaian obyek-obyek yang memiliki keterkaitan dengan gambar, misalnya ingin merepresentasikan secara visual sebuah obyek, namun ketersediaan layar atau halaman terlalu kecil untuk mengakomodir semua obyek.

Bagaimana solusinya:

1. Berikan fokus

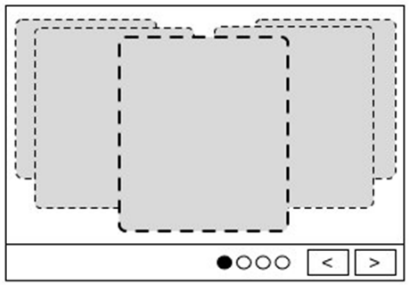
   Tampilkan setidaknya satu obyek dan berikan pilihan untuk menampilkan sebuah obyek dalam fokus, bisa difokuskan pada obyek paling tengah dari rangkaian gambar obyek.

   - Obyek dalam fokus bisa memiliki kondisi ”terpilih” (jika begitu, *carousel* harus mendukung kondisi tersebut pada semua gambar).



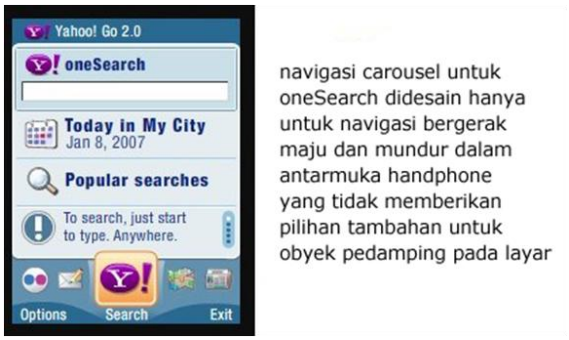
Gambar 4: Perancangan fokus obyek

   - Obyek yang dalam fokus bisa diberikan perlakuan khusus, misalnya perubahan ukuran gambar (seringkali penambahan ukuran gambar atau bisa juga pengurangan ukuran gambar seiring dengan informasi tambahan), atau indikator yang jelas dengan menempatkan obyek terpilih di depan semua rangkaian obyek.



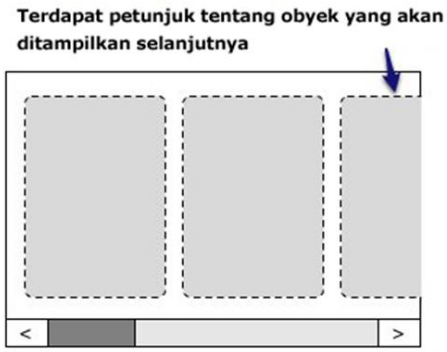
Gambar 5: Perlakuan obyek utama

- Fungsionalitas bisa ditawarkan untuk sebuah obyek terfokus, misalnya tandai (menambahkan dalam sebuah daftar khusus atau simpan sebagai favorit) atau anti-tandai (contohnya ”jangan tampilkan lagi”).
- *Carousel* harus memberikan keleluasaan kepada pengguna untuk melakukan sesuatu kepada obyek yang terlihat atau mendapatkan informasi lebih tentang obyek yang ditampilkan. Kecuali pada *carousel* yang digunakan dalam mekanisme navigasi maju dan mundur (pada antarmuka *handphone*).



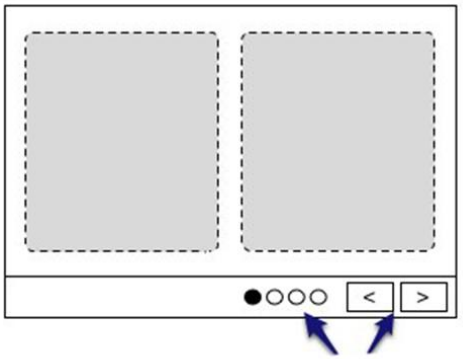
Gambar 6: Carousel mekanisme navigasi

2. Berikan fasilitas *scrolling* (menggulung)
   - Berikan pilihan (beberapa atau semua) satu atau lebih obyek tambahan pada kedua sisi dari obyek inti untuk mengundang pengguna untuk melakukan *scroll*.



Gambar 7: Petunjuk interaksi

- Tawarkan mekanisme untuk melihat bagian sebelum atau sesudah yang akan ditampilkan dalam jendela tampilan.
  - Mekanisme ini bisa berupa panah, *click and drag*, *scrollbar* atau petunjuk tampil-hilang tentang sebuah konten yang akan tampil.
  - Pendekatan ”*belts and suspenders*” (sabuk dan pegas) menampung mekanisme *multiscroll* dan tidak menambah kompleksitas antarmuka.

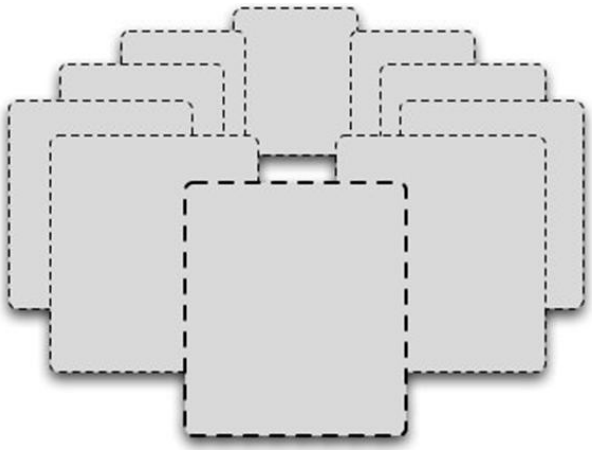


Gambar 8: Pendekatan ”*belts and suspenders*”

  - Panah *scroll* (gulungan) bisa jadi kurang nyaman untuk pencarian jejak kembali jika diletakkan berjauhan.
  - Berikan pilihan *carousel* untuk berputar secara otomatis untuk mengundang *scrolling* manual.
  - Jika kontek kerja tidak bisa didefiniskan dengan jelas, buat *carousel* dimuat bersama dengan animasi untuk menunjukkan bagaimana cara kerjanya dan memberitahu pengguna untuk berinteraksi dengan itu. Sebagai contoh, muat gambar-gambar *carousel* ke ”panggung” dari kiri ke kanan dan berhenti dimana gambar pertama dan terakhir nampak sebagian. Hal ini memberitahu pengguna bahwa mereka bisa berinteraksi dengan *carousel*. Itu juga memberi tambahan tentang definisi dari area lihat sebuah konten serta dukungan navigasinya.
  - *Scrolling* bisa berlanjut (sebuah *loop*) atau linear.
  - *Scrolling* bisa menampakkan satu obyek setiap waktu atau sebuah halaman (rangkaian) obyek dalam waktu yang sama.
  - *Scrolling* bisa berupa unidimensional (biasanya horisontal) atau bideminsional (horisontal dan vertikal).

Gambar 9: Petunjuk interaksi dengan bantuan animasi

- Sediakan pengguna sebuah petunjuk visual untuk memberitahukan kondisi pengguna saat itu.
- *Carousel* bisa ditampilkan dengan berbagai macam cara, dari cara yang paling sederhana seperti efek *filmstrip* atau *slideshow* ke sebuah cara yang lebih komplek seperti perspektif 3-D menirukan metafora rotasi.



Gambar 10: *Carousel* perspektif 3-D

3. Rancang urutan obyek

   Pertimbangkan dengan hati-hati urutan tampilan dari *carousel*, perlu diingat bahwa pengguna bisa saja tidak melakukan penjelajahan seluruh obyek dalam rangkaian, buatlah obyek yang penting untuk tampil di awal.

Alasan pemakaian *pattern* ini:

Carousel membuat desainer untuk bisa menggunakan tampilan yang menggunakan waktu nyata sebuah halaman dan dalam waktu yang sama memberikan pengalaman menarik untuk pemilihan obyek.

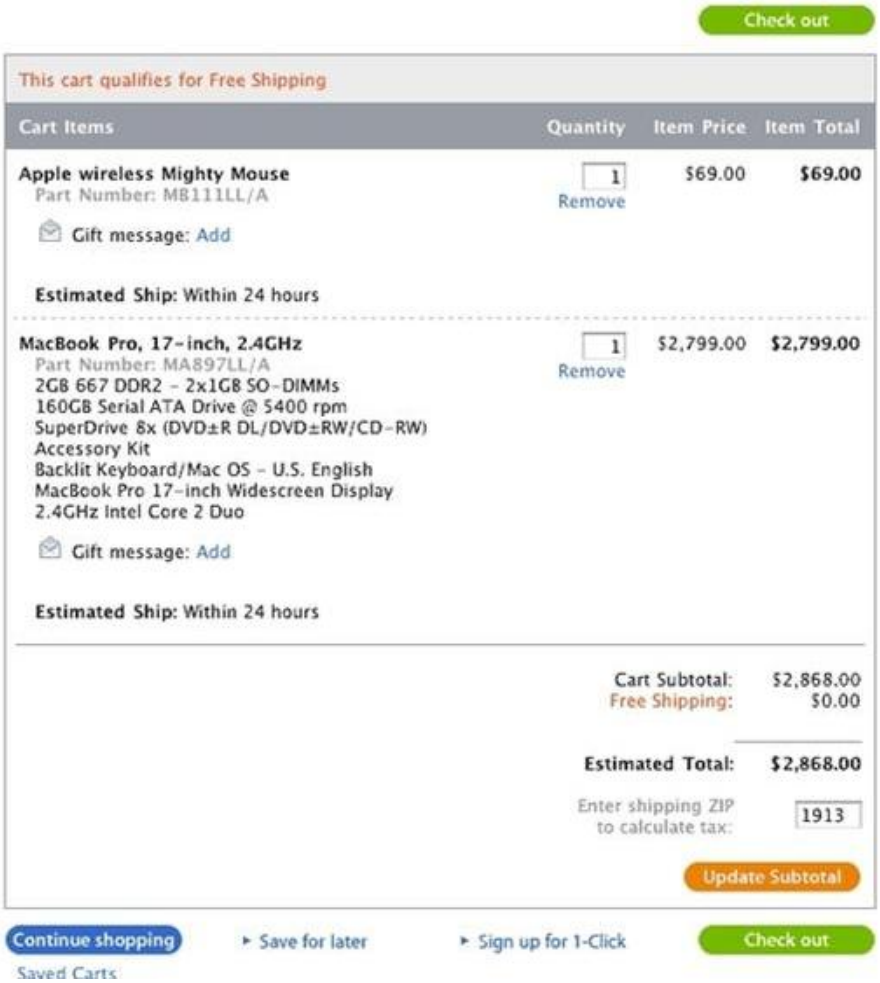
Aksesibilitas:

Carousel mampu memberikan fungsi estetika dan fungsionalitas penyajian informasi tanpa menghabiskan dimensi halaman *web*.

Sumber pustaka: ui-pattern.com & ypatterns

4.3 *Shopping Cart*

Nama *design pattern*: *Shopping cart*



Gambar 11: Implementasi *design pattern Shopping Cart*

Definisi *design pattern*:

Sebuah fungsi yang dibuat dalam bentuk *form* yang fungsinya mirip dengan keranjang belanja konvensional. *Form* ini berisikan nama produk, jumlah, harga, gambar produk, deskripsi singkat produk dalam bentuk barisan/blok.

Masalah yang diselesaikan:

Pemakai ingin membeli produk lebih dari satu, penggunaan tidak mengganggu alur aktifitas dari pengguna untuk melakukan sesi *online shopping*.

Kapan menggunakan *pattern* ini?

- Digunakan jika pemakai diperbolehkan untuk berbelanja barang lebih dari satu.
- Digunakan jika pemakai diperbolehkan untuk belanja barang lebih dari satu item pada produk yang sama.
- Digunakan saat pemakai untuk kembali dari melihat-lihat bagian lain dari *website* saat melakukan sesi belanja.
- Digunakan agar pemakai bisa kembali lagi ke sesi pemesanan atau pembelian barang.

Bagaimana solusinya?

Sebuah *shopping cart* adalah sekumpulan produk yang terpilih yang mana pemakai bisa melakukan penambahan produk atau menguranginya. Lebih lagi pemakai bisa melakukan pengurangan atau penambahan jumlah dari produk-produk yang terpilih untuk kemudian ditampilkan subtotal jumlah dan harganya. Pemakai diberikan pilihan selanjutnya yaitu pemilihan proses pembayaran dan pengiriman. Pemakai juga diberikan keleluasaan untuk melihat ringkasan pembelian yang sedang dilakukan dan juga keleluasaan untuk membatalkan transaksi kapanpun juga.

Pemakai tidak diharuskan untuk melakukan proses registrasi atau login pemakai terlebih dahulu untuk bisa memakai fitur ini. Masing-masing pemakai akan bisa diidentifikasi dari *instance* yang dikelola oleh sistem. Saat pemakai sudah siap untuk melanjutkan ke tahap berikutnya yaitu pembelian/pemesanan, pemakai baru diminta untuk memberikan informasi kontak dan melakukan proses registrasi.

Alasan pemakaian *pattern* ini

- *Shopping cart* adalah metafora yang tepat. Semua produk yang dipilih masuk ke *shopping cart* bukan berarti akan dibeli.

- Pemakai bisa tetap fokus melakukan penjelajahan produk-produk yang ditawarkan, melakukan navigasi ke semua menu dan kembali melakukan proses pemilihan produk tanpa harus khawatir akan kehilangan produk-produk yang telah dipilih.
- Bisa dimodifikasi menjadi sebuah *widget* untuk bisa ditampilkan disemua halaman sehingga memudahkan pemakai dalam mengingat status belanja. *Widget* adalah bagian sistem dari panel yang tampilannya tidak mengganggu tata letak halaman.

Aksesibilitas:

Dengan memisahkan antara aktifitas penjelajahan dengan *shopping cart* maka pemakai akan lebih mudah dalam mengingat status produk yang sudah dipilih atau yang belum dipilih. Selain itu informasi harga yang akan dibayar dari masing-masing item produk.

Sistem bisa mengambil informasi dari *shopping cart* untuk kemudian dibuatkan sebuah modul/bagian program yang bisa mendefinisikan proses pengiriman dan pilihan pelacakan produk.

Sumber pustaka: ui-pattern.com & welie.com

4.4 *Product comparison*

Nama *design pattern*: *Product Comparison*

Definisi *design pattern*:

Sebuah penyajian data yang menunjukkan dua buah data atau lebih dalam satu bentuk halaman dimana secara visual pemakai akan dengan mudah membedakan antara satu item dengan yang lain.

Matrox G200 MMS vs Millennium G550 DualDVI Comparison Chart

Product features	Matrox G200 MMS	Millennium G550 Dual DVI
Matrox graphic chip	G200	G550
Graphics chips per board	2 or 4	1
Connectors on board	1 LFH-60 on dual SKU 2 LFH-60s on quad SKU	1 LFH-60
Included cables	1 or 2 analog cables*	1 analog cable
Number of outputs	2 or 4	2
Bus type	PCI only	AGP only (4X/2X)

Gambar 12: Implementasi *design pattern Product Comparison*

Masalah apa yang diselesaikan?

Pemakai menginginkan untuk membandingkan produk-produk yang mirip.

Kapan menggunakan *pattern* ini?

- Digunakan pada *website* belanja, produk perusahaan, situs lelang.
- Pemakai mengunjungi halaman muka situs atau menemui daftar produk yang dijual kemudian pemakai menginginkan adanya informasi komparatif antara item yang ingin diperbandingkan. Dengan memberikan fasilitas ini maka pemakai akan diberikan pilihan informasi lebih mudah dicerna dan membantu pengguna dalam melakukan pemilihan produk yang tepat.
- Atribut dari produk yang ada menjadi pembeda antara satu item dengan yang lain.

Bagaimana solusinya?

Tampilkan fitur dari produk dalam bentuk matriks. Matriks perbandingan bisa diturunkan dari sejarah penjelajahan halaman atau dari sebuah aksi pemakai (melalui mengklik sebuah tombol tertentu) kemudian berikan fasilitas filter tentang atribut apa saja yang akan ditampilkan dalam matriks perbandingan.

Aksesibilitas:

Satu halaman berisi perbandingan data yang mirip atau data item yang akan diperbandingkan akan membuat proses penyampaian informasi akan lebih mudah dicerna dan efektif.

Sumber pustaka: welie.com (welie design pattern)

4.5 *Rating an object*

Nama *design pattern*: *Rating an object*

Definisi *design pattern*:

Berupa sebaris item yang bisa diklik (seringkali berupa bintang) yang menyala saat diklik. Tujuannya mengundang pemakai untuk memberikan ulasan singkat tentang produk yang bersangkutan.

Gambar 13: Implementasi *design pattern rating an object*

Masalah apa yang diselesaikan?

Pemakai ingin secara cepat memberikan opini mereka pada sebuah obyek. Proses ini diusahakan seminimal mungkin tidak mengganggu aliran aktifitas yang sedang dilakukan oleh pemakai.

Kapan menggunakan pattern ini?

- Digunakan saat pemakai diharapkan bisa menuliskan opininya secara cepat.
- Digabungkan dengan ulasan-ulasan agar lebih kaya dalam hal pengalaman pemakai.
- Rating dikumpulkan bersama untuk menampilkan peringkat rata-rata dari sebuah obyek dari pemakai yang bervariasi. Hal ini bisa digunakan untuk mencari objek apa yang paling disukai oleh pemakai.

Bagaimana solusinya?

Beberapa hal yang perlu dilakukan untuk bisa menjalankan *rating an object*, antara lain:

1. Tampilkan item yang bisa diklik.
2. Kondisi awal haruslah kosong dan berikan sebuah pesan yang mengundang opini pemakai (contoh: ”rate it!”, ”opini anda?”).
3. Saat kursor bergerak di atas ikon yang yang ada, berikan indikasi level peringkat dan berikan deskripsi dari peringkat tersebut.
4. Saat pemakai memilih bintang 3 atau bintang 5, lakukan penyimpanan rating dan pengakumulasian rata-rata peringkat kemudian tampilkan dalam tampilan terpisah.
5. Pemakai bisa mengubah rating yang telah mereka berikan.
6. Tampilkan sebuah agregasi atau peringkat rata-rata.

Alasan pemakaian pattern ini:

Rating sebuah obyek menyediakan bentuk ringan dari sebuah model *user engagement*. Rating biasanya dikaitkan dengan ulasan-ulasan untuk mendorong aktifitas dari kontribusi dari pemakai agar lebih kaya.

Aksesibilitas:

Menggunakan DHTML dan CSS untuk menampilkan tahap *rollover* sehingga terkumpul rating secara instan.

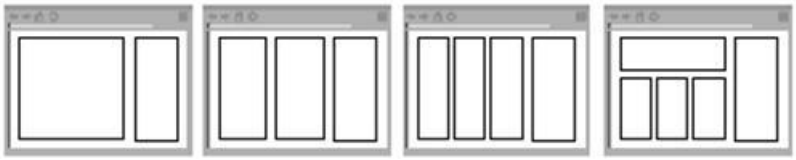
Sumber pustaka: Ypatterns

4.6 *Page grids*

Nama *design pattern*: *Page Grids*

Definisi *design pattern*:

Sebuah barisan/blok yang terstandarisasi yang menyediakan daya tarik ke pengguna dan pengalaman tersendiri, sebuah konsistensi dan juga memberikan fleksibilitas untuk desainer dalam memasukkan elemen halaman yang bersifat umum dan dinamis, serta memberikan kemudahan dalam memberikan standar ukuran untuk elemen iklan. Elemen iklan ini bisa berasal dari pihak ketiga. Selain fungsi di atas, *Page Grids* akan memberikan kemudahan dalam penggunaan ulang kode program dan meningkatkan mutu modul-modul yang ada.



Gambar 14: Implementasi *design pattern Page Grids*

Masalah apa yang diselesaikan?

Situs *web* membutuhkan konsistensi diantara elemen umum halaman *web*, lebar halaman, pembagian ruang, penggunaan iklan dan barisan kode.

Kapan menggunakan *pattern* ini?

- Saat menggunakan dan mengelola halaman *web* yang berjumlah banyak
- Halaman *web* dibuat oleh sekelompok orang dan individu yang berbeda. Misalnya *corporate blog*, *subdomain*.

Bagaimana solusinya?

Halaman *web* yang sukses seringkali menerapkan sebuah konsep tata letak yang menggunakan bantuan sekumpulan baris yang menutup halaman *web* tersebut. Tata letak tersebut akan terlihat rapi dan teratur.

Pertama, buat sekumpulan baris bantuan yang menutup halaman *web*. Sesuaikan dengan kebutuhan posisi iklan, elemen dinamis dan seterusnya.

Kemudian, buatlah cetakan dan kode untuk mendukung desainer dan pengembang sistem:

- Untuk desainer *web*, buat garis besar dari cetakan dengan menggunakan aplikasi editor grafis. Pemberian detail berupa lebar kolom dan *gutter* akan sangat membantu.

- Untuk pengembang sistem, dibuatkan sebuah cetakan berbentuk file CSS yang mengakomodir variasi halaman (seperti jumlah kolom). Pencantuman lebar kolom dan *gutter* juga dilakukan.

Alasan pemakaian *pattern* ini:

- Cetakan akan menghemat waktu penyiapan halaman bagi desainer *web* dan memberikan mereka waktu lebih untuk menyiapkan isi dari situs dan merumuskan fitur yang lebih menarik.
- Konsistensi antar laman dan elemen laman berkontribusi positif dalam hal langkah *branding* dan *user experience*.
- Kode sumber yang sama akan menawarkan keuntungan sebagai berikut:
    1. Mengurangi jumlah ketidakjelasan atau ketidak-konsistensi-an tata letak laman.
    2. Mempercepat pengembangan dan *update* halaman keseluruhan.

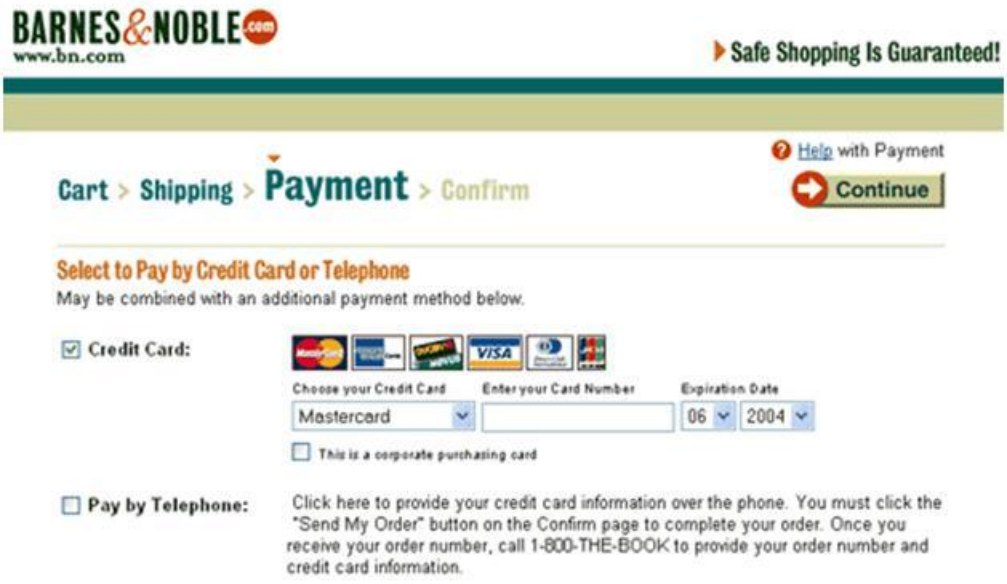
Aksesibilitas:

Dengan adanya struktur laman yang bersih dan konsisten maka akan memberikan kemudahan navigasi dan rasa nyaman bagi para pemakai. Dengan adanya CSS maka tidak perlu adanya pengaturan urutan isian dalam struktur laman HTML.

Sumber pustaka: ypatterns

4.7 *Purchase process*

Nama *design pattern*: *Purchase process*



Gambar 15: Implementasi *design pattern Purchase Process*

Definisi *design pattern*:

Sebuah antarmuka berupa *wizard* atau halaman yang berisi instruksi khusus untuk menyelesaikan proses pembelian/pemesanan barang.

Masalah apa yang diselesaikan?

Pemakai ingin membeli produk yang telah dipilih atau yang telah dimasukkan ke *shopping cart*.

Kapan menggunakan *pattern* ini?

- Digunakan pada halaman situs yang memiliki fitur penjualan produk.
- Digunakan pada halaman situs yang menawarkan pemesanan sebuah tempat atau barang.

Bagaimana solusinya?

Beberapa hal perlu dilakukan untuk bisa menjalankan *purchase process*, antara lain:

1. Identifikasi pelanggan;
2. Pemilihan alamat pengiriman barang;
3. Pemilihan metode pembayaran;
4. Tampilan pesanan keseluruhan dari pesanan;
5. Konfirmasi dan penempatan pesanan; dan
6. Penerimaan konfirmasi melalui surel atau fasilitas lain.

Pemakai bisa dengan mudah membatalkan transaksi kapanpun dan di tahap apapun. Pertimbangan adanya *wizard* adalah untuk memberikan instruksi yang berlanjut dan terfokus pada proses pembayaran sehingga memimalisir distraksi dan kesalahan pemakai. Tata letak laman web diusahakan sesederhana mungkin dengan menghilangkan elemen-elemen navigasi yang tidak perlu dan penggunaan tata letak *single column*.

Pemakai melalukan *login* dan pelengkapan data diri pada akhir transaksi. Ini dimaksudkan untuk mengantisipasi pemakai yang malas melakukan registrasi yang tidak perlu saat awal proses.

Setiap transaksi yang telah selesai harus dibuatkan sebuah konfirmasi atau sebuah tanda terima. Hal ini bisa dilakukan dengan mengirimkan surel konfirmasi. Surel ini berisikan informasi nomer pesanan, nama-nama produk, kuantitas produk, subtotal dan total harga barang. Ada baiknya diberikan sebuah fasilitas pelacakan status barang pesanan kepada pemakai.

Alasan pemakaian *pattern* ini:

Para pembeli pemula atau pelanggan yang jarang melakukan perbelanjaan *online* akan sangat terbantu dengan adanya *wizard* yang menyediakan instruksi khusus dan langkah-langkah proses yang mudah. Pelanggan yang sering melakukan proses pembayaran/pembelian *online* biasanya memakai alamat pengiriman atau nomor kartu kredit yang sama dengan transaksi sebelumnya, maka dari itu proses bisa lebih diefisienkan dengan menampilkan halaman rangkuman dan sebuah tombol beli tanpa harus mengisi ulang *form* yang sama.

Aksesibilitas:

Membuat proses pembelian lebih nyaman dengan *wizard* dan beberapa perubahan tata letak agar lebih terfokus pada proses pembelian.

Sumber pustaka: welie.com.

5. Kesimpulan

Design pattern yang ditemukan didalam pengembangan *web portal e-commerce* antara lain adalah:

1. *Product page*

   Sebuah halaman yang terdiri dari gambar dan deskripsi singkat tentang suatu produk tertentu. Biasanya terdiri dari banyak produk dalam satu halaman dan berisi kumpulan tautan yang relevan.

2. *Carousel*

   Sebuah bagian dari halaman yang memiliki fungsi untuk memanfaatkan momentum tampilan layar yang sebentar untuk menampilkan sejumlah informasi yang penting

3. *Shopping cart*

   Sebuah fungsi yang dibuat dalam bentuk *form* yang fungsinya mirip dengan keranjang belanja konvensional.

4. *Product comparison*

   Sebuah penyajian data yang menunjukkan dua buah data atau lebih dalam satu bentuk halaman dimana secara visual pemakai akan dengan mudah membedakan antara satu item dengan yang lain.

5. *Rating an object*

   Berupa sebaris item yang bisa diklik (seringkali berupa bintang) yang menyala saat diklik.

6. *Page grids*

   Sebuah barisan/blok yang terstandarisasi yang menyediakan daya tarik ke pengguna dan konsistensi bagi developer.

7. *Purchase process*

   Sebuah antarmuka berupa *wizard* atau halaman yang berisi instruksi khusus dan bertahap untuk menyelesaikan proses pembelian/pemesanan barang.

Daftar Pustaka

Hanafi, A., 2011, *Desain dan Prototipe Sistem Informasi Web Portal E-Shop Komputer dengan Pendekatan Software Design Pattern*, Thesis, Teknik Elektro, Fakultas Teknologi Informasi, Universitas Gadjah Mada, Yogyakarta.

Dofactory, 2009, Design Patterns. *http://www.dofactory.com/Patterns/Patterns.htm*, diakses tanggal 15 September 2009, pukul 10.15 WIB.

Mcleod, R., 1998, *Management Information System*, Prentice Hall, New Jersey.

Miniwats, M., 2009, Asia Internet Usage and Population, *http://www.internetworldstats.com/stats3.htm#asia*, diakses tanggal 19 Agustus 2009, pukul 17.15 WIB.

Netlingo, 2012, E-Commerce Definitions, *http://www.netlingo.com/dictionary/e.php*, diakses tanggal 26 November 2012.

Toxboe, A., 2011, UI Patterns, *http://ui-patterns.com/patterns*, diakses tanggal 11 Januari 2011, pukul 9.04 WIB.

Yacoub, S.M., 2003, *Pattern-Oriented Analysis and Design: Composing Patterns to Design Software Systems*, Addison-Wesley.

Yahoo!, 2010, Yahoo! Design Pattern Library, *http://developer.yahoo.com/ypatterns*, diakses tanggal 10 Januari 2011, pukul 21.05 WIB.