PENGEMBANGAN APLIKASI *WEB* DENGAN ICONIX *PROCESS* DAN UML STUDI KASUS: SISTEM MANAJEMEN ISI

Yulianta[1] dan Petrus Mursanto[2]

[1]Magister Teknologi Informasi, Universitas Indonesia
Jl. Salemba Raya 4, Jakarta 12000 Indonesia
yulianta@gmail.com
[2]Enterprise Computing Lab – Fakultas Ilmu Komputer, Universitas Indonesia
Kampus Baru UI Depok, Jawa Barat, 16424 Indonesia
santo@cs.ui.ac.id

Abstrak

Situs *web* sangat kompleks dengan beragam isi dan dinamis karena selalu menampilkan informasi terbaru. Kompleksitas situs *web* akan bertambah saat situs *web* tersebut harus dimodifikasi untuk menambahkan halaman-halaman baru dan fitur-fitur baru sesuai permintaan pengunjung. Untuk menghindari kompleksitas tersebut, perlu diciptakan sebuah aplikasi berbasis *web* yang dapat digunakan untuk mengelola seluruh sumber daya situs *web*. Aplikasi ini dikenal dengan sistem manajemen isi (*content management system*). Makalah ini membagikan pengalaman penulis dalam mengembangkan sistem manajemen isi pada PT X yang dinamai dengan xCMS. Metode pengembangan perangkat lunak yang digunakan adalah ICONIX *Process*. Metode ICONIX *Process* menggunakan *Unified Modeling Language* (UML) sebagai notasi utama untuk mengambarkan dan mendokumentasikan sistem yang dibangun. Makalah ini menguraikan secara garis besar aktifitas-aktifitas dan artefak-artefak yang dihasilkan pada setiap fase pengembangan. Hasil akhir adalah sebuah prototipe aplikasi manajemen isi aplikasi *web* yang diberi nama xCMS.

Kata kunci: ICONIX, *Software engineering*, CMS, UML, *Web application*.

1. Latar Belakang

Perkembangan Internet yang cukup cepat didorong oleh penggunaannya yang makin meluas. Hal ini tidak terlepas dari kemudahan penggunaan internet itu sendiri. Banyak aplikasi yang sudah ada ditulis ulang dalam bentuk aplikasi *web* agar bisa disajikan di internet.

Pengembangan aplikasi *web* memerlukan strategi khusus sesuai dengan karakteristiknya. Situs *web* biasanya sangat kompleks dengan keragaman isi (*content*) dan sangat dinamis karena selalu menampilkan informasi terbaru. Agar selalu diminati pengujung, aplikasi *web* haruslah unik dan harus dapat segera diwujudkan dalam waktu singkat agar pengunjung situs *web* dapat segera menikmatinya.

Untuk mengejar tenggat produksi, sering kali pengembang langsung menuju tahap pemrograman (*coding*) tanpa mencoba untuk mengerti apa yang sebenarnya akan dibangun dan bagaimana cara membangunnya. Akibatnya, pembuatan skrip sisi server (*server-side script*) dilakukan dengan terburu-buru, tabel-tabel dalam basis data ditambahkan sesuai keperluan, sehingga kadang-kadang berakibat pada perubahan arsitektur. Masalah yang timbul adalah kompleksitas yang meningkat sehingga modifikasi sulit dilakukan dan seandainya pun dilakukan, kestabilan sistem aplikasi menjadi terganggu.

Aplikasi sistem manajemen isi (*Content Managemet System*) berbasis *web* dibutuhkan untuk mengatasi masalah ini. Dengan adanya aplikasi CMS, diharapkan seorang yang sangat awam dengan dunia *web* dapat dengan mudah membuat situs *web* miliknya sendiri, menambahkan halaman-halaman baru, dan memperbarui informasi pada situs tersebut. Semua hal tersebut dapat dilakukan secara online dan hanya dibutuhkan sebuah penjelajah *web* (*web yy*) untuk melakukannya.

Aplikasi CMS yang dibangun juga harus mendukung alur kerja (*workflow*) untuk pengembangan situs secara kolaboratif [1], klasifikasi informasi untuk memudahkan pencarian informasi, dan dapat menangani data dalam volume yang besar sehingga dapat digunakan untuk membangun situs lebih besar misalnya *web* portal.

Diperlukan suatu disiplin dan pemodelan yang baik agar proses pengembangan aplikasi CMS ini

dapat berjalan dengan mulus dan aplikasi yang dihasilkan dapat lebih terpelihara di masa mendatang. Dengan semakin diterimanya *Unified Modeling Language* (UML) sebagai notasi untuk mengambarkan dan mendokumentasikan sistem, maka diputuskan untuk menggunakan salah satu metodologi pengembangan perangkat lunak yang mendukung UML yaitu ICONIX *Process*.

Secara umum tujuan penulisan makalah ini adalah:

- Memberikan gambaran proses-proses yang ada pada pengembangan aplikasi *web*.
- Membagikan hasil analisa, disain, dan implementasi aplikasi CMS berbasis *web*.
- Membagikan model aplikasi CMS berbasis *web* dengan menggunakan UML yang digunakan sebagai artefak-artefak pengembangan aplikasi seperti *use case diagram*, *class diagram* dan *sequence diagram*.

Aplikasi CMS yang ada di pasaran sangat beragam, kompleks dan rumit. Aplikasi CMS seperti itu berada diluar lingkup bahasan makalah ini. Aplikasi CMS yang dikembangkan disini hanyalah suatu prototipe sederhana untuk memudahkan pemahaman masalah namun tetap memperhatikan fungsi-fungsi dasar yang harus ada pada aplikasi CMS seperti:

- penciptaan dan penyuntingan isi
- manajemen isi
- penggunaan templat (*template*) untuk menyusun tampilan dan manajemennya
- manajemen menu
- manajemen berkas (*file*)
- manajemen citra (*image*)
- manajemen modul
- manajemen pengguna

2. Metodologi

Pengembangan perangkat lunak merupakan proses yang memerlukan perencanaan yang matang, kerja keras, dan melewati beberapa tahapan pengujian. Proses pengembangan perangkat lunak mempunyai empat peranan [2] yaitu:

- Memberikan bimbingan tentang pengaturan aktivitas tim.
- Menentukan artefak yang harus dibangun.
- Memberikan tugas kepada pengembang sebagai pribadi atau pun sebagai anggota team.
- Memberikan kriteria untuk memonitor dan mengukur hasil dan aktivitas proyek.

Proses pengembangan perangkat lunak menentukan alur kerja (*workflow*), aktivitas-aktivitas, artefak-artefak dan peran (*role*) dari pekerja yang terlibat pada proses pengembangan tersebut.

Alur kerja ini menentukan aktivitas-aktivitas yang harus dilakukan oleh pekerja seperti penentuan kebutuhan, pemodelan, analisa, desain, implementasi, pengujian, dan *deployment*. Aktivitas-aktivitas ini akan menghasilkan artefak-artefak yang berupa model, dokumen-dokumen, diagram-diagram, kode-kode program, dan lain-lain.

ICONIX *Process* dipilih sebagai metodologi untuk mengembangkan aplikasi sistem manajemen isi yang dikembangkan karena semua persyaratan-persyaratan di atas dapat dipenuhi oleh ICONIX *Process* [3].

Pemilihan metodologi menggunakan ICONIX *Process* tidak terlepas dari beberapa fitur utama yang dimilikinya [3] yaitu:

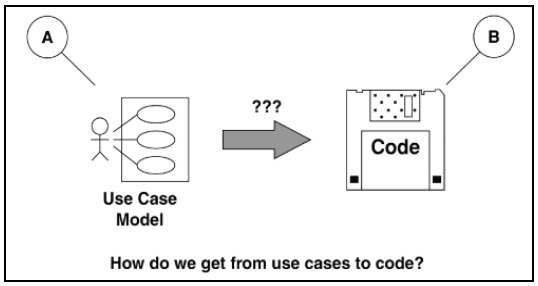
- ICONIX *Process* merupakan proses yang dipicu oleh *use case* (*use case driven*). Pada ICONIX, *use case* yang ditentukan sejak awal pengembangan menjadi dasar dalam menentukan model dan perilaku dari sistem yang dibangun.
- ICONIX *Process* merupakan proses yang berorientasi pada arsitektur (*architecture-centric*). ICONIX 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.
- 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*).
- 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.
- ICONIX *Process* memberikan keterjejakan (*traceability*) yang cukup tinggi. Merujuk kembali kepada kebutuhan awal dapat dilakukan dengan berbagai cara yang mudah pada setiap tahap pengembangan. Keterjejakan ini juga tampak pada kenyataan

bahwa setiap objek dapat dilacak langkah demi langkah, dari analisa menjadi desain.

3. ICONIX *Process*

ICONIX *Process* terletak ditengah-tengah antara *Rational Unified Process* (RUP) yang besar dan *eXtreme Programming* (XP) 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 menanggalkan analisa dan desain seperti yang dilakukan XP [4].

Tujuan utama dari ICONIX *Process* adalah bagaimana mewujudkan *use case* yang telah disusun menjadi kode seperti pada Gambar 1. Titik A adalah ide-ide tentang apa yang harus dilakukan sistem dengan cara menggambarkannya 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 [4].



Gambar 1. Bagaimana mengubah *use case* menjadi kode

Pertanyaan-pertanyaan yang mungkin muncul pada proses tersebut adalah:

- Siapa pengguna sistem (aktor) dan apa yang ingin dilakukannya?
- Apa objek-objek dalam "dunia nyata" (ranah masalah) dan relasi-relasi yang ada?
- Objek apa saja yang terlibat pada masing-masing *use case*?
- Bagaimana objek-objek pada *use case* tersebut saling berinteraksi?
- Bagaimana cara membangun aplikasi yang diinginkan?

ICONIX *Process* yang diperkenalkan oleh Doug Rosenberg berusaha menggabungkan praktek-praktek terbaik (*best practices*) dari tiga metodologi yang terlebih dulu ada yaitu *Object Modelling Techniques* (OMT) oleh James Rumbaugh, *Object-Oriented Software Engineering* (OOSE) oleh Ivar Jacobson dan *Structural Method* (Booch) oleh Grady Booch untuk menjawab pertanyaan-pertanyaan di atas seperti yang tampak pada Tabel 1.

Integrasi dari elemen Booch, Rumbaugh, dan Jacobson ini memberikan gambaran yang menyeluruh dari pengembangan perangkat lunak berorientasi objek seperti pada Gambar 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*.

Tabel 1. Pertanyaan dan jawaban tentang pengembangan perangkat lunak

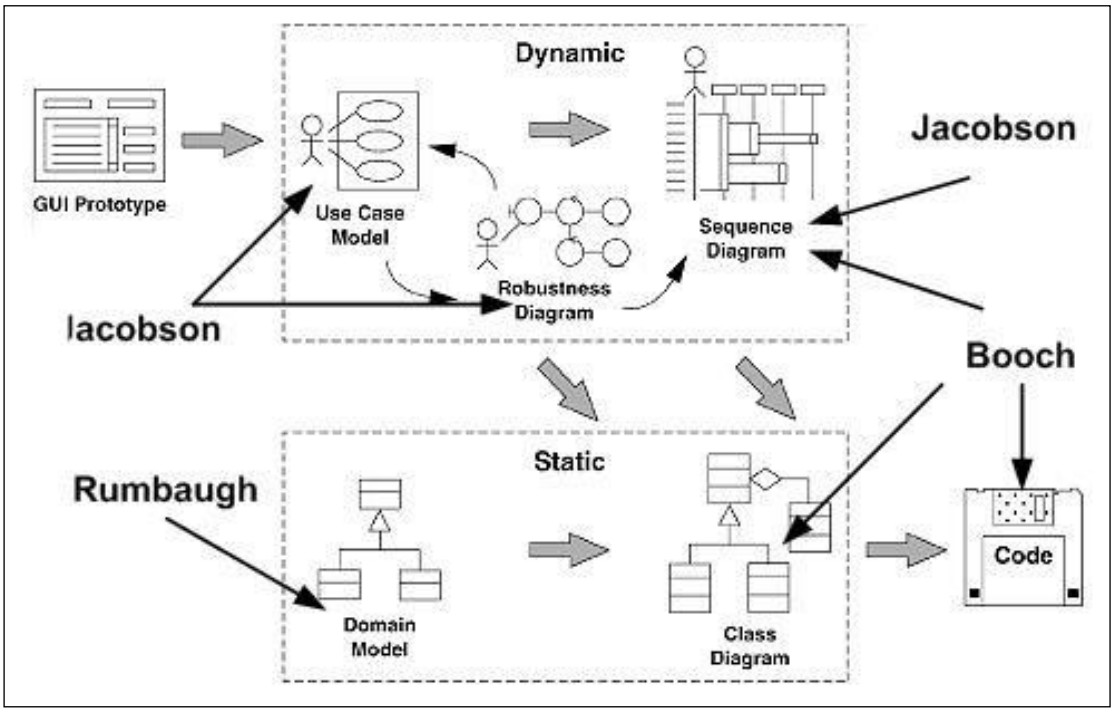
Pertanyaan	OOSE	OMT	Booch	Teknik dengan UML
Pengguna dan aksi yang dilakukan?	•			*Use case diagram*
Objek dalam "dunia nyata"?		•		*Class diagram* pada aras tinggi (gambaran kasar)
Objek pada *use case*?	•			*Robustness analysis*
Interaksi antar objek	•		•	*Sequence diagram* dan/atau *collaboration diagram*
Bagaimana membangunnya?			•	*Class diagram* pada aras rendah (gambaran detil)

Pada bagian berikut, akan dibahas lebih detil langkah-langkah pengembangan pirantu lunak mengikuti ICONIX *Process*.

3.1. Pemodelan Ranah

Pemodelan ranah (*domain modeling*) menjadi dasar dari bagian statis model 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. [3].

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.



Gambar 2. ICONIX *Process* dan Kontribusi *The Three Amigos*

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

3.2. Pemodelan *Use case*

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 mengambarkan secara detil seluruh scenario 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* kedalam *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.

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* [4].

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

4. Analisa Kehandalan

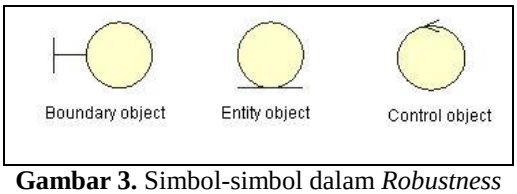
Analisa kehandalan (*robustness analysis*) diperlukan untuk mengetahui objek-objek apa saja yang terlibat dalam setiap *use case*. Analisa kehandalan (*robustness analysis*) diperkenalkan secara informal oleh Ivar Jacobson pada tahun 1991. Proses ini dilakukan dengan cara menganalisa teks *use case*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

- *Boundary object*
  Objek yang digunakan aktor sebagai antarmuka untuk berkomunikasi dengan sistem.
- *Entity object*
  Objek ini biasanya berupa objek yang berasal dari ranah model.
- *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:

- Uji kelayakan – Analisa kehandalan membantu untuk meyakinkan bahwa dokumen *use case* yang disusun sudah benar dan menghindari perilaku sistem yang tidak sesuai
- Uji kelengkapan - Analisa kehandalan membantu meyakinkan bahwa *use case* telah meliputi semua aksi alternatif (*alternate courses*).
- Menemukan objek-objek baru – Analisa kehandalan membantu menemukan objek-objek yang mungkin terlewatkan pada saat pemodelan ranah
- 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.



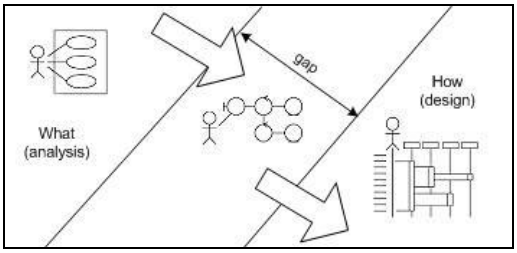
Gambar 3. Simbol-simbol dalam *Robustness Diagram*

Gambar 3 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*) sepeerti yang tampak pada Gambar 4.

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

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



Gambar 4. *Robustness Diagram* sebagai jembatan antara analisa dan desain

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

5. *Model-View-Controller*

Model-View-Controller (MVC) merupa-kan pola umum dalam aplikasi *web* [5]. Prinsipnya adalah menyusun aplikasi menjadi tiga lapis sehingga antara kode dan tampilan merupakan lapisan-lapisan yang terpisah. Hal ini akan membantu desainer grafis dan programmer untuk bekerja bersama-sama tanpa saling menganggu. Desainer grafis dapat berkonsentrasi pada penciptaan tampilan halaman *web* yang memikat, sedangkan programmer berkonsentrasi pada penciptaan paket-paket aplikasi yang lengkap

Bila diperhatikan, terdapat kesesuaian antara pola MVC dengan objek-objek pada analisa kehandalan. *Entity object* merupakan model, *boundary object* merupakan *view*, dan *control object* merupakan *controller* pada pola MVC. Aplikasi xCMS berusaha menerapkan pola MVC dengan cara berikut:

- **Model**
  Model diimplementasikan dengan menciptakan objek-objek pada model ranah aplikasi xCMS menjadi paket-paket dan kelas-kelas
- ***View***
  View diimplementasikan dengan penggunaan templat. Templat ini adalah berkas halaman *web* (*web pages*) dengan tambahan *tag* khusus yang akan diterjemahkan oleh aplikasi xCMS menjadi tampilan halaman klien (*client pages*) yang diinginkan. Lebih lanjut tentang tag khusus ini, dapat dilihat dalam dokumen xCMS

Administration yang disertakan pada *source code* aplikasi xCMS.

- ***Controller***
  Controller diimplementasikan dengan mengubah objek-objek manager (*Category Manager, Content Manager, File Explorer*, dsb) menjadi halaman-halaman *server* (*server pages*).

6. Pemodelan Interaksi

Setelah tahap pemodelan ranah dan analisa kehandalan dilalui, maka hamper semua objek-objek dalam ruang permasalahan telah berhasil diidentifikasi lengkap dengan sebagian atribut-atributnya.

Jika analisa kehandalan berkutat dengan penemuan objek-objek, maka pemodelan interaksi berkutat dengan alokasi perilaku-perilaku yaitu alokasi fungsi-fungsi perangkat lunak kedalam objek-objek yang berhasil diidentifikasi.

Pemodelan interaksi dilakukan dengan mencermati dokumen *use case* untuk melakukan identifikasi perilaku-perilaku sistem yang kemudian dilanjutkan dengan melakukan alokasi perilaku-perilaku tersebut sebagai operasi-operasi dalam kelas-kelas. Dari proses ini akan dihasilkan *sequence diagram* sebagai detil desain dari sistem yang dibangun.

Setelah melakukan pemodelan ranah dan melakukan analisa kehandalan maka hampir semua objek dalam ruang permasalahan berikut atribut-atributnya berhasil diidentifikasi. Relasi statis ditunjukkan dalam *class diagram* aras tinggi, sedangkan sedikit relasi dinamis ditunjukkan dalam *robustness diagram*. Pemodelan interaksi diperlukan untuk melengkapi relasi dinamis tersebut.

Pemodelan interaksi (*interaction modeling*) menunjukkan bagaimana objek-objek saling berkolaborasi untuk membentuk fungsi-fungsi yang harus disediakan oleh sistem. Tujuan utama dari pemodelan interaksi adalah [4]:

- Alokasi perilaku terhadap *boundary*, *entity*, dan *control objects*. Pada saat analisa kehandalan, sekelompok objek yang membentuk perilaku sistem berhasil diidentifikasi. Pada saat pemodelan interaksi inilah ditentukan objek mana saja yang bertanggung jawab membentuk potongan perilaku sistem yang diinginkan.
- Menunjukkan detil interaksi yang terjadi antar objek pada setiap *use case*. Objek berinteraksi dengan mengirimkan pesan-pesan (*messages*) satu dengan lainnya. Sebuah pesan akan membangkitkan suatu objek untuk melakukan aksi tertentu. Pada saat pemodelan interaksi inilah dilakukan identifikasi pesan-pesan atau fungsi-fungsi apa saja yang dibutuhkan untuk membentuk potongan perilaku sistem tersebut.
- Melengkapi operasi-operasi pada kelas-kelas. Analisa kehandalan harus berhasil melakukan identifikasi semaksimal mungkin atribut-atribut pada model statis dan menghasilkan seminimal mungkin operasi-operasi. Pada fase pemodelan interaksi inilah dilakukan identifikasi sebagian besar operasi-operasi yang dilakukan oleh sistem.

7. Tinjauan Desain Kritis

Tinjauan desain kritis (*critical design review*) digunakan untuk meyakinkan bagaimana (*how*) detil desain seperti yang tampak pada *sequence diagram* dan *class diagram*, sesuai dengan apa (*what*) yang ditentukan pada *use case*.

Proses ini berguna untuk meyakinkan bahwa detil desain cukup mendalam dan cukup mudah untuk diwujudkan menjadi kode-kode. Berkaitan dengan hal ini, penentuan arsitektur sistem yang dipakai dan bahasa pemrograman yang dipakai untuk melakukan implementasi menjadi sesuatu yang sangat penting..

8. Implementasi

Fase terakhir pada ICONIX *Process* adalah implementasi. Setelah tahap desain dilalui, maka model statis yang dihasilkan telah lengkap. Paket-paket telah dirancang, kelas-kelas telah lengkap dengan atribut-atribut dan operasi-operasi. Model statis ini selanjutnya diserahkan kepada programmer untuk diwujudkan menjadi kode program.

9. Penutup

ICONIX *Process* menawarkan metodologi singkat namun lengkap untuk membangun perangkat lunak termasuk aplikasi *web*. Tujuan utama dari ICONIX adalah mewujudkan *use case* yang telah disusun menjadi kode. Proses –proses yang terlibat di dalamnya:

- *Requirement Analysis*
- *Analysis & Preliminary Design*
- *Design*
- *Implementation*

UML merupakan notasi yang tepat untuk memodelkan dan mendokumentasikan sistem termasuk aplikasi *web*. ICONIX menggunakan UML secara elegan, tidak berlebihan. Hal ini pula yang menyebabkan ICONIX lebih ringkas dibanding metodologi lainnya. Diagram-diagram UML yang digunakan dalam ICONIX:

- *Use case diagram*

- *Class Diagram*
- *Robustness Diagram*
- *Sequence Diagram*

Keunggulan utama ICONIX (dan membedakannya dengan metodologi lain) adalah pemanfaatan *robustness diagram* untuk melakukan analisa kehandalan. Dengan analisa kehandalan, akan ditemukan objek-objek baru yang sebelumnya tidak teridentifikasi.

Untuk menentukan model dinamis dari sistem yang dibangun, ICONIX menitikberatkan pada penggunaan *sequence diagram*. Dari sequence diagram yang dihasilkan akan ditemukan operasi-operasi yang harus dimiliki suatu kelas.

Dengan ICONIX, model aplikasi sistem manajemen isi berbasis *web* yang disebut xCMS berhasil dibangun. Penerapan metodologi ICONIX *Process* sangat membantu dalam menemukan objek-objek yang terlibat di dalamnya

Beberapa saran-saran yang dapat disampaikan berkaitan makalah dan aplikasi xCMS yang dibangun antara lain:

1. Manfaatkan kotak dokumentasi pada *Rational Rose* untuk mendokumentasikan semua objek pada model yang dirancang. Dokumen ini dapat dengan mudah disertakan pada kode yang dibangkitkan sebagai dokumentasi kode.
2. Analisa kehandalan perlu dilakukan dengan hati-hati agar semua objek teridentifikasi dengan benar, tetapi jangan terlalu cemas (terlalu berhati-hati) yang berakibat pada lambatnya proses analisa.
3. Templat merupakan hal yang sangat penting bagi aplikasi xCMS karena dengan templat inilah tampilan situs *web* disusun. Penamaan tag-tag templat yang baik, konsisten, dan mudah diingat akan sangat membantu pengguna. Penggunaan tema (*theme*) pada templat akan menjadi nilai tambah bagi aplikasi xCMS versi berikutnya.

REFERENSI

[1] McIntosh, Michael. *Content Management Using the Rational Unified Process*, Rational Software White Paper.

[2] Booch, Grady. *Object–Solutions: Managing the Object-Oriented Project*, Addison Wesley Longman, 1996.

[3] RosenBerg, Dooug. *Inside the ICONIX Process*, Addison-Wesley, 2001.

[4] RosenBerg, Dooug. and Kendall Scott, *Applying Use case Driven Object Modelling with UML: An Annotated E-Commerce Example*, Addison Wesley, 2001.

[5] RosenBerg, Dooug, and Kendall Scott, *Use case Driven Object Modelling with UML: A Practical Approach*, Addison Wesley, 2001.

[6] Fueck, Harry. "Model View Controller Pattern", http://www.phppatterns.com/ index.php/article/ view/11/1/1.