REKAYASA PERANGKAT LUNAK DENGAN *MODEL UNIFIED PROCESS* STUDI KASUS: SISTEM INFORMASI JOURNAL

Ibnu Akil
Program Studi Manajemen Administrasi
ASM BSI Jakarta
Jl. Jatiwaringin Raya No.18, Jakarta Timur
Ibnu.ial@bsi.ac.id

Abstract — *In the era of object oriented programming, the analysis models of structured approaches is start to be left behind, because it is no longer relevan with object-oriented paradigm. Nowadays who become the standard of analysis and design of object-oriented system is Unified Modeling Language (UML). However UML is just notations, it doesn't provide a framework to work in software engineering. The Unified Proces which proposed by Rumbaugh and friends, is a framework for object-oriented software engineering.*

Intisari — Di era pemrograman berorientasi objek, model-model analisa dan perancangan system terstruktur (*structured approach*) sudah mulai ditinggalkan karena tidak relevan lagi dengan paradigma berorientasi objek. Sekarang ini yang menjadi standarisasi analisa dan perancangan sistem berorientasi objek adalah bahasa pemodelan UML. Namun UML hanya berisi diagram-diagram dan notasi-notasi, tidak menjelaskan mengenai kerangka kerja dalam proses rekayasa perangkat lunak. Unified Proces yang diprakarsai oleh Rumbaugh dan kawan-kawan, merupakan kerangka kerja yang diperuntukkan untuk rekayasa perangkat lunak berorientasi objek.

Kata Kunci: *Information system, Software engineering, UML, Unified Proces*

PENDAHULUAN

Setiap pengembangan perangkat lunak memerlukan proses. Roger pressman dalam bukunya *Software Engineering 7th Edition* mengatakan bahwa "proses perangkat lunak adalah suatu kerangka kerja untuk aktifitas-aktfitas, tindakan-tindakan, dan tugas-tugas yang dibutuhkan untuk membangun perangkat lunak berkualitas tinggi" (*Pressman, Software Engineering A Practitioner's Approach 7th Edition*, 2010).

Yang menjadi pertanyaan disini adalah apakah model pengembangan sistem yang ada (*waterfall, RAD, prototyping, CBD, dan evolutionary*) bisa digunakan untuk pengembangan berorientasi objek? Jawabnya, menurut Pressman system berorientasi objek cenderung berkembang seiring waktu berjalan. Karenanya model proses evolutionary dipasangkan dengan *component-based development* adalah paradigma terbaik untuk rekayasa perangkat lunak berorientasi objek (Pressman, *Software Engineering 5th Edition*, 2001). Pernyataan Pressman di atas bukan berarti bahwa kita harus menggunakan model proses *evolutionary* atau *component-based development.* Tetapi disini penulis akan mencoba menggunakan model *Unified Pocess*. Uml menyediakan teknologi yang dibutuhkan untuk mendukung rekayasa perangkat lunak berorientasi objek, namun tidak menyediakan process framework (kerangka kerja proses) untuk memandu team proyek dalam pengembangan sistemnya. Setelah selesai dengan UML Jacobson, Rumbaugh, dan Booch, mengembangkan *Unified Process*, sebuah kerangka kerja untuk rekayasa perangkat lunak berorientasi objek.

Sebuah kerangka kerja proses menspesifikasikan siapa melakukan aktifikas apa pada produk-produk kerja apa, termasuk kapan, bagaimana, mengapa, dan dimana aktivitas tersebut dilakukan. Kerangka kerja proses mendeskripsikan kegiatan proses sebagai proses-proses terkait yang lebih fleksibel dan dapat ditingkatkan, dan contoh-contoh proses menjalankan sebagian dari kerangka proses. UP adalah kerangka proses dan kasus-kasus pengembangan adalah contoh-contoh proses.

BAHAN DAN METODE

Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan visual yang digunakan untuk menspesifikasikan, memvisualisasikan, membangun, dan mendokumentasikan rancangan dari suatu sistem perangkat lunak (Rumbaugh, Jacobson, & Booch, 2005).

Pemodelan memberikan gambaran yang jelas mengenai sistem yang akan dibangun baik dari sisi struktural ataupun fungsional. UML

dapat diterapkan pada semua model pengembangan, tingkatan siklus sistem, dan berbagai macam domain aplikasi. Dalam UML terdapat konsep semantik, notasi, dan panduan masing-masing diagram. UML juga memiliki bagian statis, dinamis, ruang lingkup, dan organisasional. UML bertujuan menyatukan teknik-teknik pemodelan berorientasi objek menjadi terstandarisasi.

Sejarah UML

Ada beberapa usaha untuk menyatukan metode-metode berorientasi objek yang bermunculan. Contohnya Fusion oleh Coleman dan kawan-kawan (Coleman-94), yang memasukan konsep dari OMT (*Object Modeling Technique*)(Rumbaugh-91),Booch (Booch-91), dan CRC (Wirfs-Brock-90). Karena hal ini tidak melibatkan penulis asli, maka dianggap sebagai metode baru dan bukan menggantikan berbagai metode-metode yang sudah ada. Usaha penggabungan yang sukses pertamakali dan mengganti metode yang ada adalah ketika Rumbaugh bergabung dengan Booch pada perusahaan Rational Software tahun 1994. Mereka mulai mengkombinasikan konsep OMT dan metode Booch, yang menghasilkan proposal pertama tahun 1995. Pada waktu itu, Jacobson juga bergabung di Rational dan mulai bekerjasama dengan Booch dan Rumbaugh. Kerjasama mereka disebut *Unified Modeling Language* (UML). Mereka merevisi metode masing-masing untuk menghasilkan satu metode lengkap yang harmonis.

Pada tahun 1996 *Object Management Group* mengeluarkan permintaan untuk proposal-proposal untuk pendekatan standar pemodelan berorientasi objek. *Unified Modeling Language* diadopsi oleh anggota OMG sebagai standar pada November 1997. OMG bertanggung jawab untuk pengembangan lebih lanjut dari standar UML.

Artifact UML

A. Structure Diagram

Structure diagram menunjukkan struktur statis dari sistem dan bagian dari abstraksi serta level implementasi yang berbeda dan bagaimana bagian-bagian tersebut saling berelasi satu sama lain. Elemen-elemen dari *structure diagram* merepresentasikan konsep sistem yang memiliki arti. Termasuk abstraksi dunia nyata dan konsep implementasi. *Structure diagram* tidak menggunakan konsep hubungan waktu, tidak menunjukkan detail-detail dari tingkah laku yang dinamis. Namun mereka mungkin menunjukkan relasi tingkah laku dari *classifiers* yang ditunjukkan dalam *structure diagram*.

1. ***Class Diagram***
2. ***Object Diagram***
3. ***Package Diagram***
4. ***Model Diagram***
5. ***Composite Structure Diagram*, meliputi:**
   a) ***Internal Structure Diagram***
   b) ***Collaboration Diagram***
6. ***Component Diagram***
7. ***Deployment Diagram***
8. ***Profile Diagram***

B. Behavior Diagram

Behavior Diagram menunjukkan tingkah laku dinamis dari objek-objek dalam sistem, yang mana bisa dijelaskan sebagai sederet perubahan-perubahan dalam sistem sepanjang waktu.

1. ***Use Case Diagram***
2. ***Activity Diagram***
3. ***State Machine Diagram***
4. ***Interaction Diagram* meliputi:**
   a) ***Sequence Diagram***
   b) ***Communication Diagram***
   c) ***Timing Diagram***
   d) ***Interaction Overview Diagram***

C. Unified Process

Kerangka kerja proses UP meliputi *collaboration*, *context*, dan *interaction*. *Collaboration* menekankan kepada elemen-elemen dari proyek, *context* menekankan kepada kerangka proses dari framework, dan *interaction* menekankan kepada eksekusi dari proyek. (Alhir, 2002)

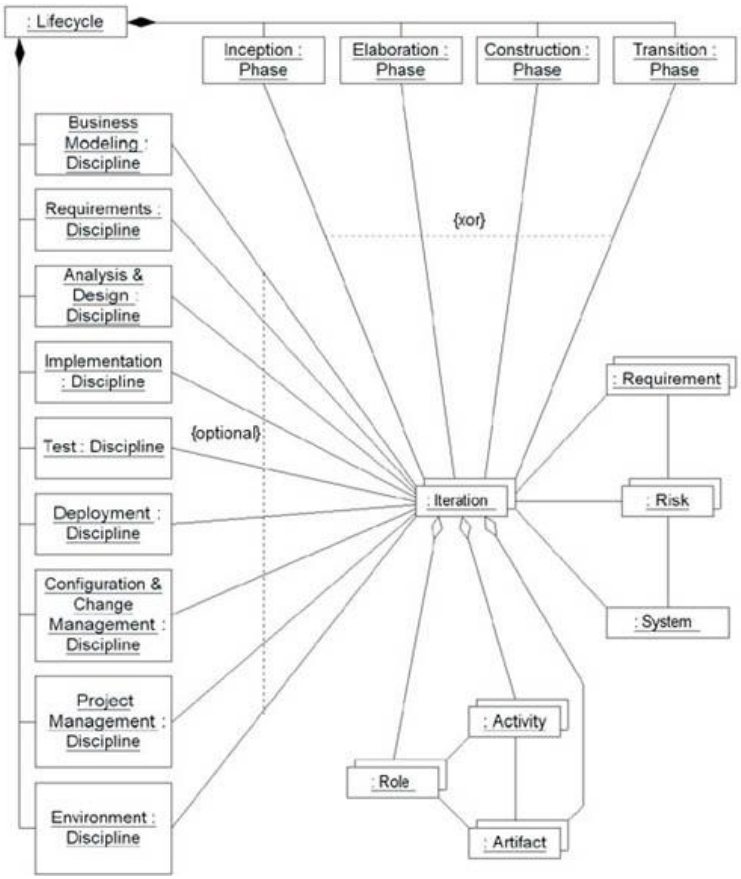
D. Collaboration

Collaboration (baca: kolaborasi) melibatkan interaksi di dalam context. Sebuah kolaborasi menggambarkan siapa melakukan aktivitas apa dan pada produk kerja apa. Sehingga kolaborasi menetapkan elemen-elemen dari proyek.

Sebuah role adalah sebuah individu atau team yang bertanggung jawab terhadap aktivitas-aktivitas dan artifact. Sebuah aktifitas adalah sebuah unit kerja, terdiri dari tahapan-tahapan, yang dilaksanakan oleh sebuah role. Sebuah artifact adalah sebuah elemen dari informasi yang merupakan tanggung jawab dari pemegang role, dan yang dihasilkan atau digunakan oleh aktivitas-aktivitas.

E. Context

Sebuah *context* (baca: konteks) berisi struktur atau aspek statis dari sebuah kolaborasi. Sebuah konteks menggambarkan kapan dan dimana aktivitas-aktivitas tersebut dilaksanakan serta produk kerja yang dihasilkan dan digunakan. Gambar berikut adalah konteks yang ditetapkan dalam UP.



Sumber : (Alhir, 2002)

Gambar 1. Context yang ditetapkan dalam UP.

Siklus hidup dari sebuah proyek terdiri dari tahapan-tahapan (*phases*) dimana di dalam iterasi melibatkan disiplin-disiplin. Sebuah pengembangan terdiri dari tahapan-tahapan yang berurutan yang menghasilkan rilis utama dari system yang disebut generasi sistem (misal: versi 1.0, 2.0). sebuah tahapan (*phase*) adalah pencapaian utama, sebuah keputusan manajemen yang difokuskan untuk menangani resiko bisnis. Tahapan-tahapan berisi proses pemecahan masalah pada level yang kecil. Sebuah perulangan (*iteration*) adalah suatu pencapaian kecil, sebuah keputusan teknis yang difokuskan kepada mengelola resiko teknis, yang menghasilkan rilis yang kecil dari system yang disebut *system increment*. Sebuah disiplin adalah satu bidang perhatian dimana rincian alur kerja meliputi kumpulan dari aktivitas-aktivitas.

UP mendefinisikan empat tahapan sebagai berikut:

a. ***Inception phase***, memfokuskan pada menetapkan batasan dan visi dari proyek, yaitu menetapkan kelayakan bisnis dari usaha dan memantapkan tujuan dari proyek. Tahapan inception menghasilkan capaian tujuan.

b. ***Elaboration phase***, memfokuskan pada kebutuhan system dan arsitekturnya, yaitu menetapkan kelayakan teknis dari usaha dan memantapkan arsitektur system. Tahapan *elaboration* menghasilkan capaian arsitektur.

c. ***Construction phase***, memfokuskan pada pembangunan system. Tahapan construction menghasilkan capaian awal operasional dari system.

d. ***Transition phase***, memfokuskan pada menyempurnakan transisi atau instalasi dari system kepada *user*. Tahapan *transition* menghasilkan capaian rilis produk.

UP mendefinisikan beberapa disiplin-disiplin sebagai berikut:

a. *Business Modeling*, memfokuskan pada pemahaman bisnis yang sedang dikembangkan dan menggambarkan pengetahuan bisnis tersebut dalam bisnis model.

b. *Requirement*, memfokuskan pada kebutuhan sistem yang mengotomatiskan bisnis dan menggambarkannya dengan use case model.

c. *Analysis and Design*, memfokuskan pada menganalisa kebutuhan dan merancang

system dan menggambarkan pengetahuan tersebut dalam design model.

d. *Implementation*, memfokuskan pada meng-implementasikan system berdasarkan pada implementation model.
e. *Test*, memfokuskan pada pengujian system berdasarkan pada test model.
f. *Deployment*, memfokuskan pada instalasi system berdasarkan deployment model.
g. *Configuration and Change Management*, memfokuskan pada mengelola konfigurasi dari system dan permintaan perubahan.
h. *Project Management*, memfokuskan pada mengelola proyek.
i. *Environment*, memfokuskan pada lingkungan dari proyek termasuk proses-proses dan perangkat kerja.

F. Interaction

Sebuah *interaction* (baca: interaksi) menekankan pada tingkah laku atau aspek dinamis dari kolaborasi, elemen-elemen yang berkolaborasi dan kerjasama mereka dalam suatu komunikasi. Interaksi menggambarkan kapan dan mengapa aktivitas-aktivitas tersebut dilakukan dan produk kerja yang dihasilkan dan digunakan.

Sebuah perulangan (*iteration*) adalah langkah-langkah sepanjang jalan untuk mencapai tujuan. Sebuah perulangan bersifat mengulang dan melibatkan *work* dan *re-work* serta bersifat *incremental*. Sebuah use case adalah kebutuhan fungsional. Karena UP merupakan model berbasis use-case driven, maka use case – use case mengendalikan jalannya perulangan seiring dengan timbulnya permintaan-permintaan fungsionalitas baru.

Sebuah arsitektur (*architecture*) dari system melibatkan kumpulan dari elemen-elemen dan bagaimana mereka berkolaborasi dan berinteraksi. Resiko adalah hambatan untuk mencapai tujuan, termasuk di dalamnya manusia, bisnis, dan kendala teknis.

G. Iteration

Sebuah perulangan adalah direncanakan, dieksekusi, dan dievaluasi. Use case – use case dan resiko diutamakan. Ketika merencanakan perulangan, use case – use case yang memiliki resiko tertinggi dan bisa memberikan akomodasi dari faktor-faktor yang membatasi perulangan (dana, sumberdaya, waktu, dll), dipilih untuk mengendalikan perulangan. Ketika mengeksekusi perulangan, use case – use case berkembang melalui disiplin-disiplin, begitu juga dengan system dan arsitekturnya.

Studi Kasus Sistem Informasi Journal

Sistem informasi journal disini adalah sistem untuk mempublikasi artikel ilmiah. Sistem ini dimulai ketika seorang penulis mengirimkan artikel ilmiahnya kepada suatu penerbit jurnal. Artikel-artikel yang masuk sebelum dipublikasi harus direview terlebih dahulu oleh reviewer seseuai bidang keilmuannya untuk menjamin bahwa artikel tersebut layak dan memenuhi syarat keilmuan. Bagian administrasi jurnal akan mendistribusikan artikel-artikel yang masuk kepada reviewer, kemudian reviewer akan menetapkan apakah artikel tersebut layak untuk dipublikasi atau perlu direvisi terlebih dahulu. Jika perlu direvisi artikel akan dikembalikan kepada penulis. Apabila artikel-artikel yang sudah direview dan layak untuk dipublikasi sudah memenuhi kuota publikasi, maka administrasi jurnal akan mulai menyusun artikel-artikel dalam satu volume terbitan, baru kemudian di publikasi.

A. Inception Phase

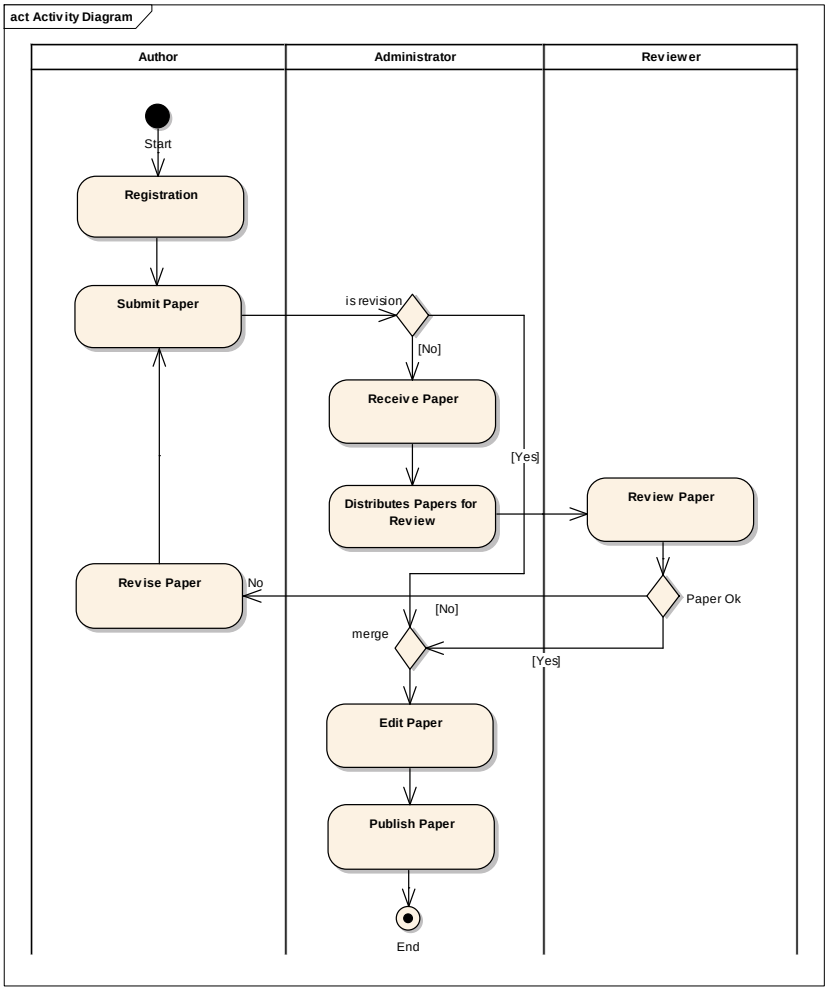
Business Modeling

Tujuan utama dari business modeling adalah memahami proses bisnis dan segala pengetahuan yang terkait operasional proses yang ada.

Proses Bisnis

1. Registrasi
2. Mengirim Artikel
3. Mengelola Artikel
4. Mereview Artikel
5. Mengedit Artikel
6. Publikasi Artikel

Untuk menggambarkan bisnis proses atau work flow dalam UML anda dapat menggunakan activity diagram.



Sumber : (Hasil Penelitian, 2015)

Gambar 2. Rancangan Activity Diagram Proses Bisnis

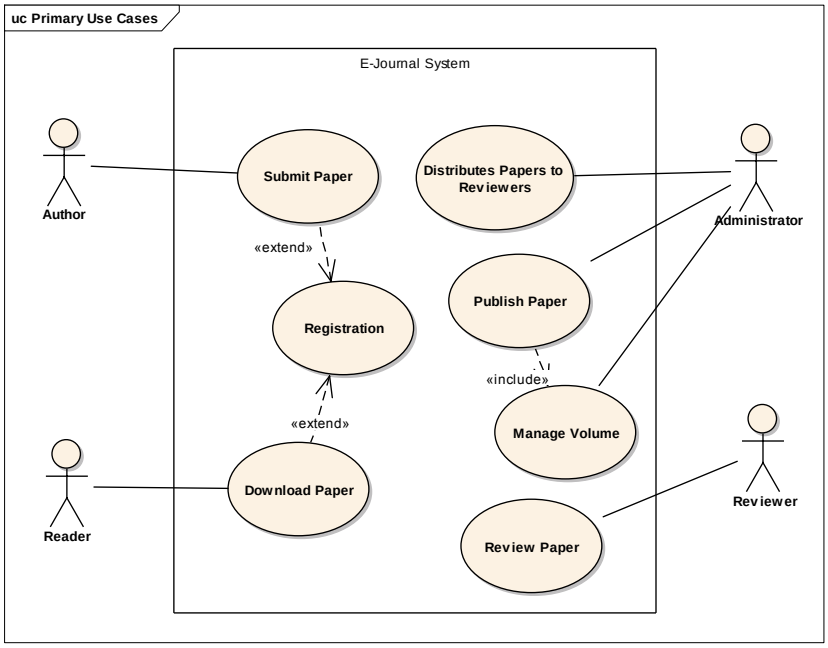
Requirement

Untuk mendefinisikan kebutuhan system anda dapat menggunaka use case diagram, dimana pada tahapan ini anda melihat system sebagai satu konteks. Jangan terburu-buru menggambarkan use case diagram sampai detail, identifikasi terlebih dahulu use case – use case utama, karena use case – use case tersebut akan berkembang pada setiap perulangan tahapan.

Aktor

Ada beberapa aktor yang akan berperan di dalam sistem ini yaitu; *reader, author, administrator*, dan *reviewer.*

a) Reader
b) Author
c) Administrator
d) Reviewer

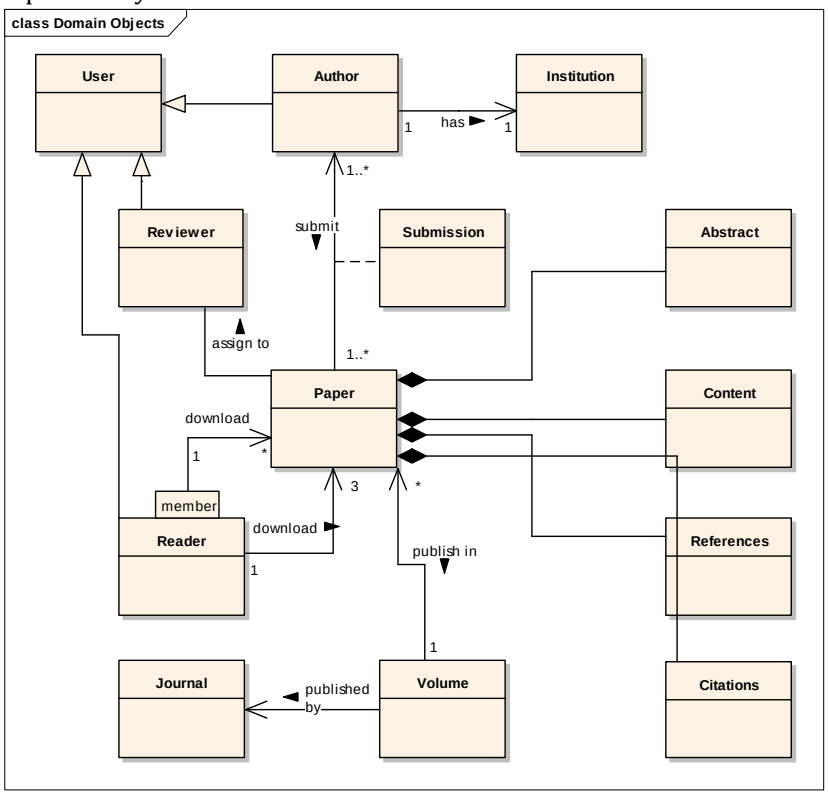


Sumber : (Hasil Penelitian, 2015)

Gambar 3. Rancangan Use Case Diagram Konteks Sistem

Analysis & Design

Pada tahapan ini anda masih mengidentifikasi objek objek yang terlibat dalam system beserta peranannya. Anda tidak harus menggambarkan dalam bentuk class diagram secara lengkap, cukup kelas-kelas tanpa atribut dan operasi sebagai *domain problem* seperti berikut:



Sumber : (Hasil Penelitian, 2015)

Gambar 4. Rancangan Class Diagram Domain Problem

Elaboration Phase

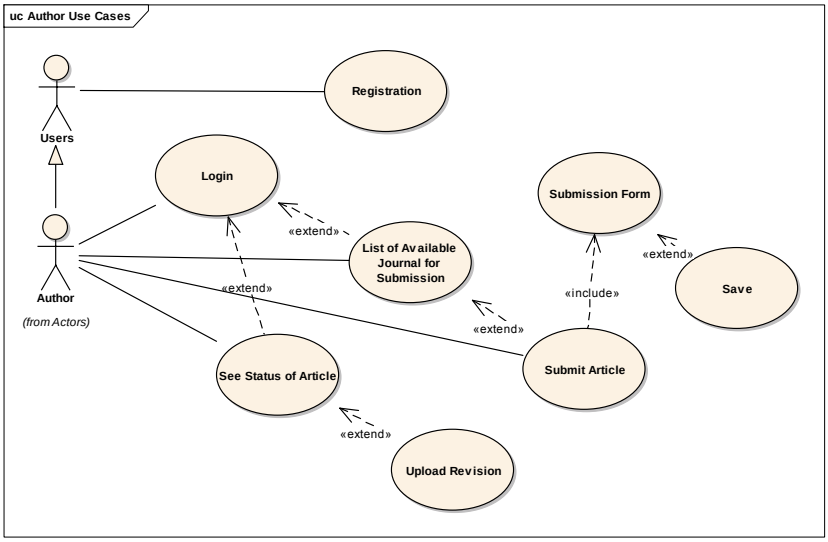
Pada tahapan ini anda mulai menggali lebih dalam kebutuhan-kebutuhan sistem, arsitektur sistem, desain sistem sampai mendapatkan gambaran sistem keseluruhan secara terinci.

Milestone-nya adalah arsitektur sistem yang lengkap.

Requirement

Anda dapat melakukan perbaikan-perbaikan kebutuhan system pada putaran ini. Identifikasi lebih detail kebutuhan sistem, termasuk fitur-fitur dan batasan sistem. Serta pengecualian-pengecualian yang mungkin ada dalam sistem. Gambarkan dalam use case diagram system (*sea level*) dan sub system (*fish level*) jika memungkinkan.

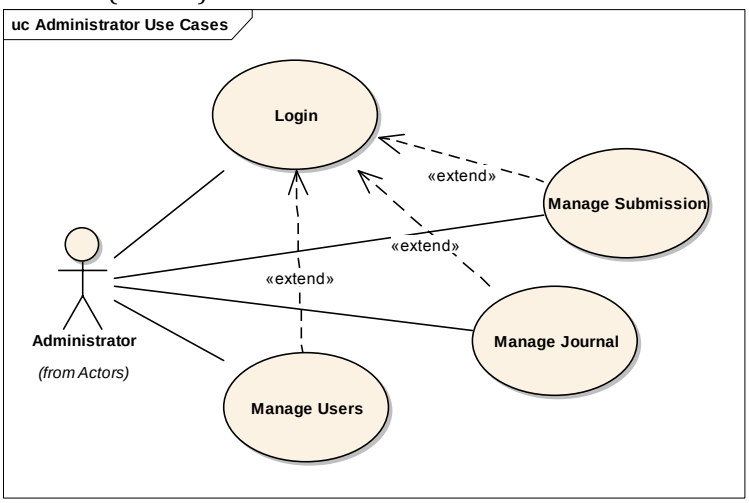
Ada baiknya sebelum menggambarkan use case yang lebih detail, anda organisir use case – use case tersebut ke dalam package-package yang relevan, sehingga anda dapat menentukan fungsionalitas system yang dibutuhkan masing-masing jenis user. Kemudian gambarkan detail use case dari masing-masing package.



Sumber : (Hasil Penelitian, 2015)

Gambar 5. Rancangan Use Case Package Author.

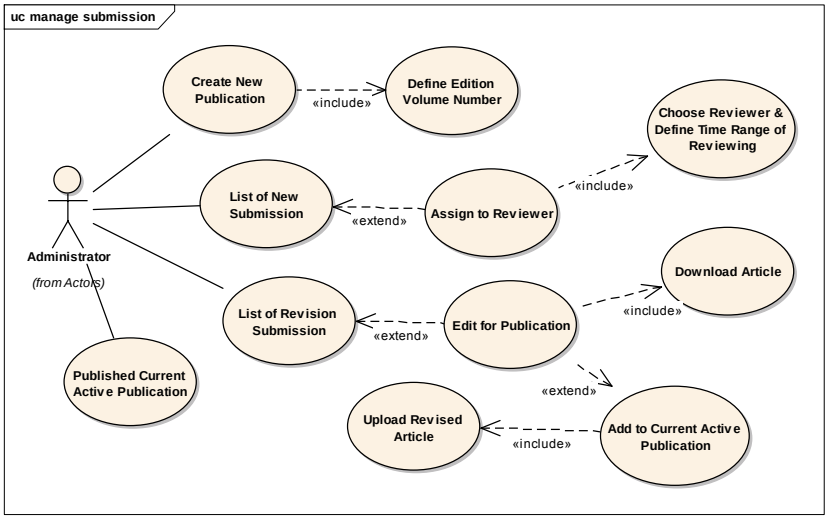
Perhatikan dari gambar di atas use case *"List of Available Journal for Submission"*, "*See Status of Article*" merupakan extensi dari use case login, yang berarti use case – use case tersebut dapat dieksekusi setelah user (Author) melakukan login. Perhatikan bahwa setiap notasi ini memiliki arti dan implementasi yang sesuai dengan fungsinya dalam system. Jangan sampai anda salah menggambarkan.



Sumber : (Hasil Penelitian, 2015)

Gambar 6. Rancangan Use Case Diagram Package Administrator.

Use case diagram Package Administrator di atas adalah sea level, yang berarti jika memungkinkan dapat anda uraikan detailnya menjadi fish level.



Sumber : (Hasil Penelitian, 2015)

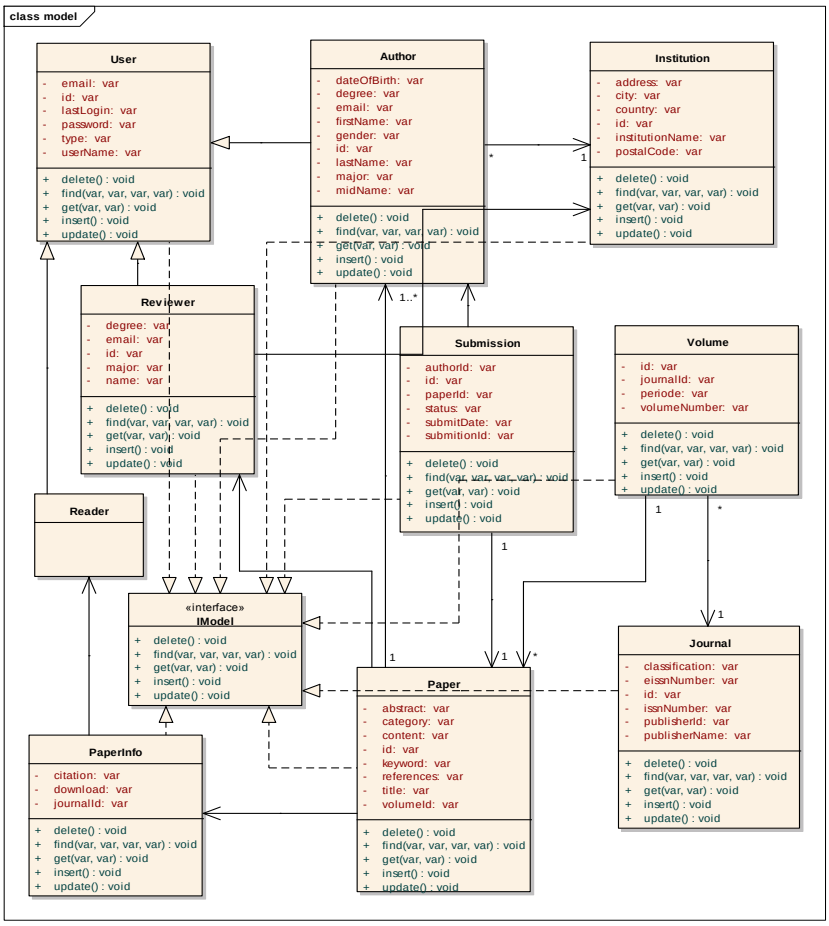
Gambar 7. Rancangan Use Case Diagram Fish Level dari Manage Submission.

Diagram-diagram use case di atas adalah beberapa contoh saja dari studi kasus ini. Karena tidak memungkinkan penulis gambarkan secara keseluruhan di buku ini. Intinya, anda dapat memahami konsep dan implementasi diagram-diagram UML dalam praktek sesungguhnya.

Analysis & Design

a) Rancangan Arsitektural Sistem

Arsitektur system dapat anda gambarkan dengan beberapa diagram statis dari UML diantaranya; Class Diagram, Component Diagram, Deployment Diagram, dan Artifact Diagram.



Sumber : (Hasil Penelitian, 2015)

Gambar 8. Rancangan Class Diagram Model

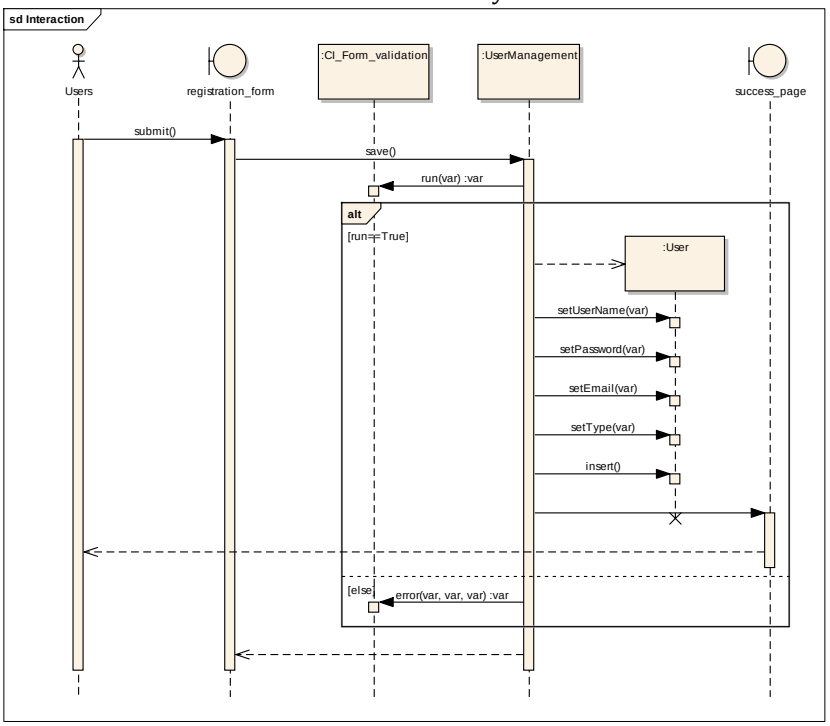
Class Diagram

Rancangan class diagram diatas adalah *refinement* (perbaikan) dari rancangan class domain problem pada tahapan inception. Pada phase ini anda sudah menentukan atribut-atribut dan operasi-operasi pada masing-masing kelas. Anda juga sudah menetapkan bahasa pemrograman yang akan digunakan, dimana dalam rancangan ini system akan diimplementasikan dengan bahasa PHP dan menggunakan framework CodeIgniter.

Operasi-operasi dasar manipulasi data yang ada pada kelas-kelas di atas adalah implementasi dari interface IModel. Anda juga dapat menentukan operasi dengan bantuan sequence diagram, acitivity diagram, dan state machine diagram.

b) Rancangan Tingkah Laku Sistem

Tingkah laku sistem adalah sisi dinamis dari system. Sisi dinamis ini menggambarkan interaksi antar komponen-komponen statis (mis. Kelas) dalam system sebagai response dari event yang diterima atau realisasi dari suatu use case sub system.



Sumber : (Hasil Penelitian, 2015)

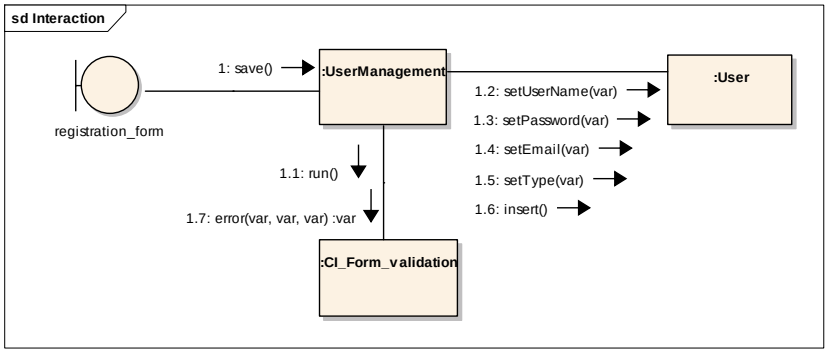
Gambar 9. Rancangan Sequence Diagram Registration

Sequence Diagram

Kadang kala ada beberapa use case yang dapat digambarkan dalam satu sequence diagram. Yaitu use case – use case extensi dan use case – use case inklusi. Hal ini akan membantu anda lebih mudah untuk memahaminya.

Communication Diagram

Berikut ini adalah rancangan communication diagram dari use case registration. Objek-objek yang terlibat di dalamnya sama dengan yang terlibat di dalam sequence diagram registration.



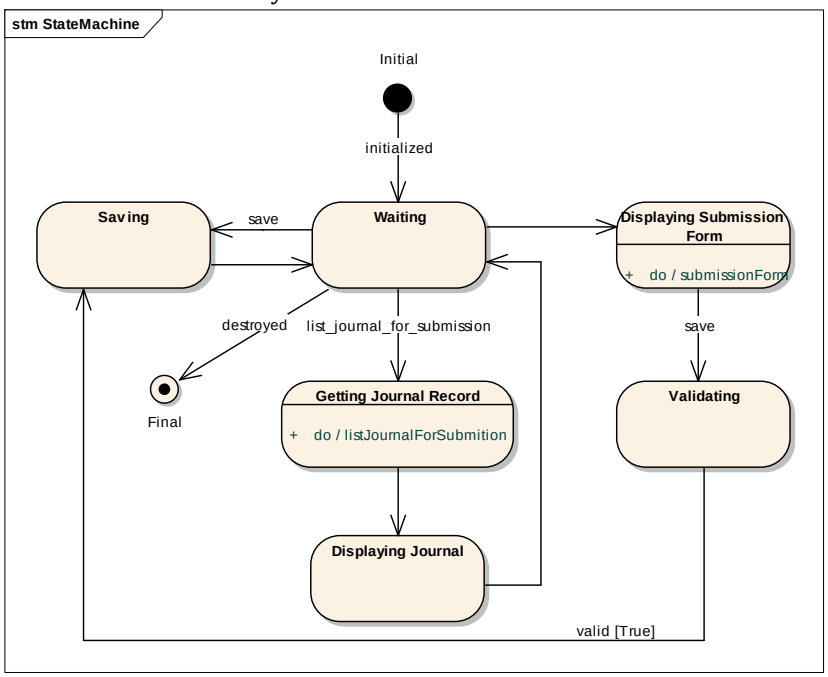
Sumber : (Hasil Penelitian, 2015)

Gambar 10. Rancangan Communication Diagram Use Case Registration

State Machine Diagram

State machine menggambarkan status-status dari system dari waktu ke waktu selama masa hidup system. Anda dapat menggambarkan state dari suatu use case, sebuah kelas, atau system secara keseluruhan.

Object AuthorManagement ini adalah kelas yang akan kita gambarkan state machine-nya. Namun untuk menggambarkan state machine anda harus menangkap event-event yang terkait dengan objek tersebut bukan hanya dari satu sequence diagram, tapi dari banyak sequence diagram yang melibatkan objek tersebut. Atau bisa juga dari communication diagram.



Sumber : (Hasil Penelitian, 2015)

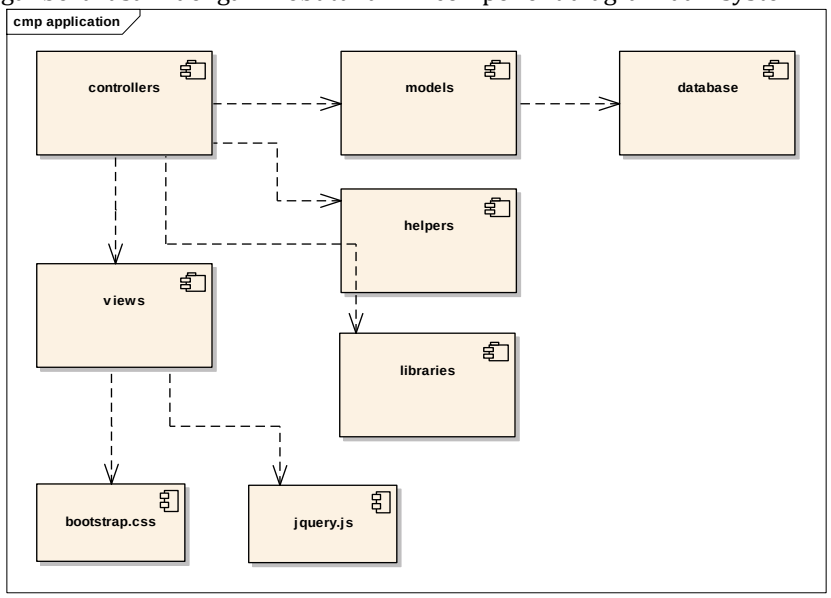
Gambar 11. Rancangan State Machine Kelas *AuthorManagement*

Construction Phase

Tahapan ini menekankan pada meng-implementasikan rancangan yang dihasilkan pada akativitas analisis dan desain. Selain itu tahapan ini juga berurusan dengan kebutuhan yang bersifat non-fungsional dan *deployment* modul-modul "*executable*".

Rancangan Component

Gambar berikut adalah rancangan component diagram dari system.

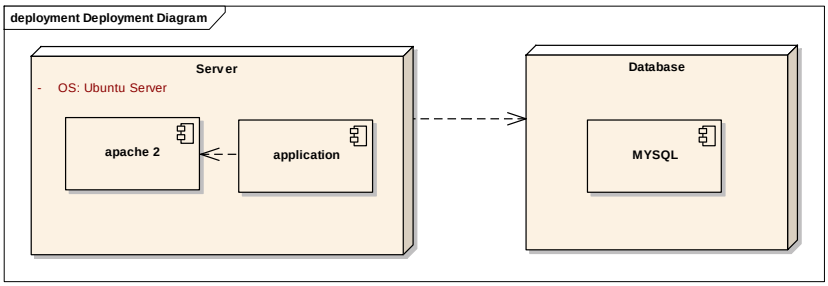


Sumber : (Hasil Penelitian, 2015)

Gambar 12. Rancangan Component Diagram.

Rancangan Deployment

Rancangan deployment diagram meliputi menentukan komponen-komponen yang akan di deploy (install) pada node-node termasuk menentukan perangkat lunak dan perangkat keras yang dibutuhkan dalam node-node tersebut.



Sumber : (Hasil Penelitian, 2015)

Gambar 13. Rancangan Deployment Diagram

Transition Phase

Tahapan transisi adalah di release-nya sistem versi beta. Tahapan ini menekankan kepada instalasi system versi beta, memantau umpan balik dari user dan menangani modifikasi-modifikasi atau update-update yang diperlukan. Hal ini bisa melibatkan desain lebih lanjut dan bahkan use case – use case baru. Tahapan ini selesai dengan di release-nya system versi produksi.

HASIL DAN PEMBAHASAN

Unified Process menyediakan kerangka kerja yang cukup lengkap. Kerangka kerja tersebut terbagi menjadi empat tahapan (*inception, elaboration, construction* dan *transition*). Dalam setiap tahapan yang penulis tekankan adalah rancangan diagram-diagram UML yang sesuai pada tahapan dan proses kerja-proses kerja tahapan tersebut, agar anda dapat menangkap rancangan sistem secara keseluruhan pada setiap tahapan, sesuai dengan milestone tahapan tersebut.

KESIMPULAN

Unified process sebagai kerangka kerja proses rekayasa perangkat lunak terasa cukup ringan dan tidak terlalu membebani pengembang dengan proses-proses yang tidak terlalu penting. Kekurangannya *Unified process* tidak menekan kepada hal-hal yang bersifat manajerial seperti pengelolaan sumberdaya, pengelolaan waktu dan pengelolaan keuangan. *Unified process* hanya menekankan pada pengembangan model dan desain yang esensial bagi aplikasi. Jika anda menginginkan kerangka kerja yang ringan dan tujuan akhir (*system*) tercapai tanpa usaha yang keras, maka unified process cocok untuk anda. Tetapi jika anda menginginkan kerangka kerja yang lebih lengkap dengan segala fitur proses yang banyak, anda dapat mencoba kerangka kerja yang lain.

REFERENSI

Alhir, S. S. 2002. Understanding the Unified Process (UP). *Methods & Tools*.

Hunt, J. 2000. *The unified process for practitioners : object-oriented design, UML and Java.* London: Springer.

Pressman, R. 2001. *Software Engineering 5th Edition.* New York: McGraw Hill.

Pressman, R. 2010. *Software Engineering A Practitioner's Approach 7th Edition.* New York: McGraw-Hill.

Rumbaugh, J., Jacobson, I., & Booch, G. 2005. *The Unified Modeling Language Reference Manual Second Edition.* Canada: Pearson Education.

BIODATA PENULIS

Ibnu Akil, M.Kom. Jakarta 15 Januari 1980. Magister Ilmu Komputer Program Pasca Sarjana Nusamandiri. Bekerja sebagai Dosen di AMIK BSI dan Konsultan IT. Beberapa karya ilmiah yang telah di hasilkan adalah : Analisis dan Desain Sistem Berbasis Web dengan Web Application Extension-UML dan Framework MVC, Rancang Bangun Sistem Informasi Perpustakaan Berbasis Web Menggunakan MVC (*Model View Controller*), ImplementasiPersistence Dengan Framework Hibernate untuk Meningkatkan Efektifitas Pemrograman, Metode Pembelajaran *Object Oriented Programming* (OOP) dengan Pendekatan *Hemispheric Cognitive Style Collaboration.*