PENGEMBANGAN PERANGKAT LUNAK *MONITORING* DATA KONTRAK (Studi Kasus: BAUK Universitas Diponegoro Semarang)

Firhan dan Aris Puji Widodo

Departemen Ilmu Komputer/Informatika, Fakultas Sains dan Matematika
Universitas Diponegoro
[1]Firhan.faisal1995@gmail.com, [2]arispw@gmail.com

Abstract

Biro Administrasi Umum dan Keuangan (BAUK) Diponegoro University was one of the bureau at Diponegoro University which had a task in managing administration and finance, one of them was contract data supervision. Contract data was a document which was created by the unit/ faculty to apply budgetary funds, the created documents would be verified by the BAUK. To solve those problems, a softtware were created to perform contract data monitoring that could facilitate the submission of contract data until the data summary contract payment. The software would be developed by using Unified Process (UP) approach which was a model of software development processes and by using Object Oriented (OO) method. OO was a method of software development based on the abstraction of objects. The result of KARWAS development by using UP was a contract data supervison card which contained unit of work, detailed contract, partnership, and payment session.
Key words : *bauk, data contract, monitoring, unified process.*

Abstrak

Biro Administrasi Umum dan Keuangan(BAUK) Universitas Diponegoro merupakan salah satu badan di Universitas Diponegoro yang memiliki tugas dalam mengelola administrasi dan keuangan, salah satunya adalah pengawasan data kontrak. Data kontrak merupakan dokumen yang dibuat oleh unit/ fakultas untuk mengajukan penganggaran dana, dokumen yang telah dibuat tersebut nantinya akan diverifikasi oleh pihak BAUK. Dalam hal ini dibuat perangkat lunak untuk melakukan *monitoring* data kontrak yang dapat memfasilitasi kegiatan pengajuan data kontrak sampai ke rekapitulasi data pembayarannya. Perangkat lunak ini dikembangkan dengan menggunakan pendekatan *Unified Process*(UP) yang dimana merupakan model proses pengembangan perangkat lunak dan menggunakan metode *Object oriented*(OO). OO merupakan metode pengembangan perangkat lunak yang berdasarkan abstraksi objek-objek. Permodelan KARWAS menggunakan UP menghasilkan kartu pengawasan kontrak yang didalamnya terdapat satuan kerja, detail kontrak, rekanan, dan termin pembayaran.
Kata kunci : bauk, data kontrak, *monitoring*, *unified process*

1. Pendahuluan

Biro Administrasi Umum dan Keuangan (BAUK) Universitas Diponegoro merupakan salah satu badan di Universitas Diponegoro yang memiliki tugas dalam mengelola administrasi dan keuangan, salah satunya adalah pengawasan data kontrak. Data kontrak merupakan dokumen yang dibuat oleh Unit/Fakultas untuk mengajukan penganggaran dana. Dokumen yang telah dibuat tersebut nantinya akan diverifikasi oleh pihak BAUK. BAUK dapat menyetujui ataupun menolak dokumen yang telah dibuat oleh fakultas ketik melakukan verifikasi.

Pihak Fakultas dan Universitas belum menerapkan sistem komputerisasi secara optimal ketika melakukan aktivitas mengelola data, melainkan fakultas masih membuat dokumen proposal secara manual dan memberikannya ke pihak universitas untuk dilakukan verifikasi dan kemudian pihak universitas melengkapi realisasi pembayaran dari kontrak tersebut apabila pagu telah diturunkan. Hal ini akan menimbulkan pemrosesan dan *monitoring* data kontrak relatif cukup lama serta kurang lengkapnya dokumen dan laporan yang telah terkumpul.

Memasuki era modern dan globalisasi pada saat ini, peranan teknologi menjadi semakin penting dalam menangani permasalahan-permasalahan yang ada pada aktifitas kesehariannya. Peranan teknologi ini dapat dimanfaatkan oleh BAUK untuk mengembangkan suatu perangkat lunak yang dapat memfasilitasi kegiatan pengajuan data kontrak sampai ke rekapitulasi data pembayarannya.

Terdapat metode dan model proses yang digunakan sebagai bantuan untuk membuat perangkat lunak ini. Salah satu metode yang digunakan pada proses pengembangan perangkat lunak adalah *Object Oriented*(OO), OO merupakan metode pengembangan berdasarkan abstraksi objek-objek. Sedangkan model proses yang digunakan dalam pengembangan perangkat lunak ini adalah *Unified Process*(UP). UP adalah model proses pengembangan perangkat lunak yang terdefinisi dengan baik. UP biasa di gunakan untuk mengembangkan teknologi berbasis sistem pada

objek dan/atau berbasis komponen. Penggunaan model proses UP dilakukan karena proses pengembangan perangkat lunak *monitoring* data kontrak ini tidak hanya berjalan dalam satu kali proses, melainkan banyak melakukan proses pendekatan sampai mendapatkan *requirements* dan hasil sesuai kebutuhan BAUK Universitas Diponegoro.

Dengan adanya penerapan perangkat lunak *monitoring* data kontrak ini, maka pemrosesan dan *monitoring* data kontrak akan berjalan dengan efektif dan efisien. Pihak Fakultas dapat membuat dan mengajukan proposal data kontrak ke Universitas secara digital dan dengan kurun waktu yang singkat dan pihak Universitas dapat melakukan *monitoring* pemrosesan dan kontrak yang telah berjalan secara digital.

Berdasarkan uraian yang telah dijelaskan, dapat di rumuskan permasalahannya yaitu bagaimana membuat suatu perangkat lunak *monitoring* data kontrak dengan mengimplementasikan metode pengembangan OO dan model proses pengembangan *unified process* pada BAUK Universitas Diponegoro.

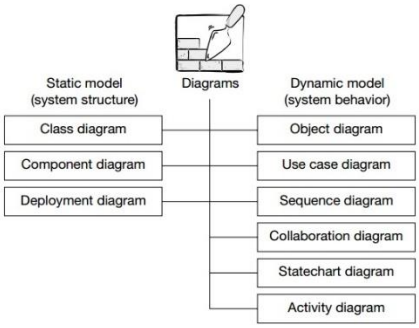
Tujuan yang ingin dicapai dari penelitian ini ialah untuk membangun perangkat lunak *monitoring* data kontrak BAUK Universitas Diponegoro agar kegiatan pengajuan proposal kontrak, realisasi termin pembayaran, penagihan sampai dengan pencetakan laporan dapat dilakukan secara optimal dan juga dapat membangun data *history* yang dapat digunakan di kemudian hari.

Adapun manfaat yang akan dicapai dalam pengembangan perangkat lunak *monitoring* data kontrak BAUK Universitas Diponegoro ini adalah membantu pihak BAUK Universitas Diponegoro dalam melakukan *monitoring* dan pemrosesan terdapat data kontrak. Manfaat lainnya yaitu sebagai sarana rekapitulasi data kontrak, sehingga pihak Fakultas maupun BAUK dapat melihat *history* data kontrak.

2. Tinjauan Pustaka dan Metode

Unified Modeling Language (UML)

UML adalah bahasa permodelan visual untuk sistem atau perangkat lunak. UML memiliki aplikasi yang lebih luas karena mekanisme di dalam pembangunannya diperpanjang, meskipun UML yang paling sering dikaitkan dengan sistem perangkat lunak adalah pemodelan *Object Oriented* (OO). UML tidak menyediakan metodologi permodelan, akan tetapi UML itu sendiri hanya menyediakan sintaksis visual yang dapat kita gunakan untuk membangun model (Arlow & Neustadt, 2002). Tidak ada urutan tertentu dalam menentukan pembuatan diagram pada UML, meskipun biasanya dimulai dengan *use case* diagram untuk menentukan ruang lingkup sistem. Bahkan, pada beberapa diagram dikerjakan secara paralel, sambil memperbaiki diagram masing-masing untuk menemukan informasi yang lebih rinci tentang sistem perangkat lunak yang dirancang. Dengan demikian, diagram-diagram merupakan mekanisme utama untuk memasukkan informasi yang dibutuhkan ke dalam model. Terdapat sembilan tipe diagram pada UML yang terbagi menjadi dua kelompok model yaitu *static model* dan *dynamic model*, seperti yang tertera pada Gambar 2.1.

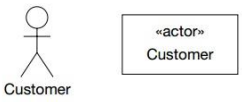


Gambar 2.1 Jenis UML Diagram (Arlow & Neustadt, 2002)

Use Case Diagram

Pemodelan *Use Case* adalah teknik pembentukan dari kebutuhan. Keluaran atau hasil dari aktivitas pemodelan *use case* ini ada 4, antara lain [1]:

1. Aktor – di analogikan dengan orang atau sesuatu yang berhubungan dengan sistem. Representasi aktor dalam UML ditunjukkan pada Gambar 2.2



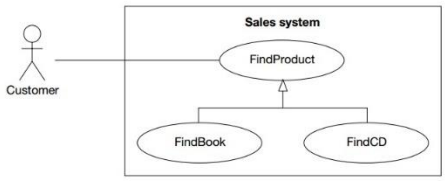
Gambar 2.2 Representasi Aktor dalam UML

Pada Gambar 2.2 dapat dilihat terdapat aktor *Costumer* yang dilambangkan dengan ikon *stickman* atau kotak. Kedua representasi aktor tersebut sah untuk digunakan akan tetapi banyak orang yang menggunakan model *stickman*.

2. *Use case* – sesuatu yang aktor dapat lakukan pada system [1]. Representasi *use case* dapat dilihat pada Gambar 2.3

Gambar 2.3. Representasi use case dalam UML

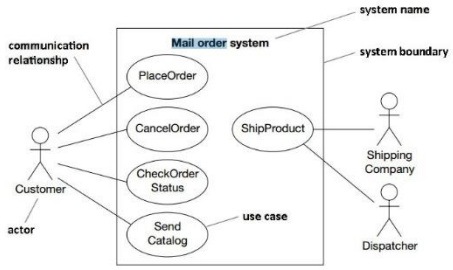
3. *Relationship* – Hubungan antara aktor dan *use case*. Representasi hubungan antara aktor dan *use case* dapat di lihat pada Gambar 2.4



Gambar 2.4. Representasi hubungan aktor dan *use case* dalam UML

Gambar 2.4 menunjukan ekstrak dari *use case* diagram sistem penjualan, dapat di lihat terdapat *parent use case* yaitu *FindProduct,* dan kemudian terdapat 2 *use case* FindBook dan *FindCD* yang merupakan *child use case* dari *FindProduct*.

4. *System boundary* – Kotak di sekitar *use case* untuk menunjukan ujung atau batasan sistem yang telah dimodelkan. Representasi *system boundary* pada pemodelan *use case* dapat di lihat pada Gambar 2.5



Gambar 2.5. Representasi *system boundary* dalam UML

Pada Gambar 2.5 dapat dilihat terdapat 3 aktor yaitu *Costumer*, *Dispatcher* dan *Shipping Company* yang dilambangkan dengan ikon *stickman* dan 5 buah *Use case* yaitu *PlaceOrder, CancelOrder, CheckOrder Status, Send Catalog* dan *ShipProduct*, kelima *use case* tersebut merupakan satu kesatuan sistem dalam mengirimkan surat yaitu *Mail order system*. Oleh karena itu, diberi kotak di sekitar *use case* yang menandakan garis batas sistem atau *system boundary* dari *Mail order system.*

***Sequence* Diagram**

Sequence diagram adalah sebuah *interaction diagram* yang menekankan urutan waktu pertukaran pesan. *Sequence diagram* terdiri atas dua bagian utama, yaitu *lifeline* dan *message*. *Lifeline* digambarkan dengan garis vertikal putus-putus yang digambar di bawah *object*. *Lifeline* menunjukkan masa hidup *object*. *Message* digambarkan dengan anak panah antara dua garis vertikal yang menunjukkan *lifeline object*. Urutan *message* ditunjukkan secara vertikal. *Message* pertama digambarkan paling atas, sedangkan *message* terakhir digambarkan paling bawah dalam diagram [2].

Terdapat 3 jenis panah pada *sequence diagram* antara lain :

1. *Procedure call*

Procedure call menunjukan pengirim pesan atau fungsi menunggu sampai penerima telah selesai menjalankan fungsi atau menerima pesan tersebut [1]. Representasi *procedure call* ditunjukan oleh gambar 2.6.

Gambar 2.6. Representasi *Procedure call*

2. *Asynchronous call*

Asynchronous call menunjukan proses sesegera mungkin dilakukan setelah pengirim mengirimkan pesannya, penerima menerima pesan tanpa membutuhkan waktu tunggu yang lama [1]. Representasi *Asynchronous call* ditunjukan oleh gambar 2.7

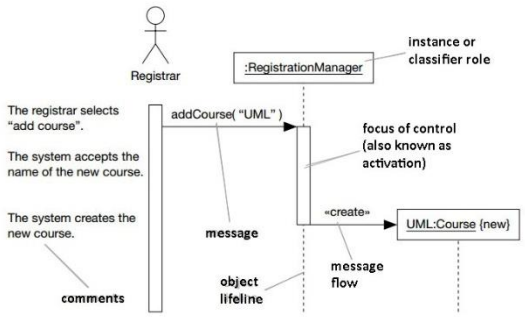
Gambar 2.7. Representasi *Asynchronous call*

3. *Return*

Return merupakan hal yang harus dipenuhi dalam *procedure* call karena menunjukan respon yang diberikan oleh penerima kepada pengirim [1]. Representasi Panah *Return* ditunjukan oleh gambar 2.8

Gambar 2.8. Representasi Panah *Return*

Sequence diagram melayani tujuan yang sedikit berbeda untuk *collaboration* diagram dalam analisis OO. *Collaboration* diagram sangat baik dalam menunjukkan objek-objek aktual dan hubungan struktural mereka (*collaboration*), tetapi mereka lebih lemah ketika menunjukkan interaksi antara objek-objek sebagai urutan waktu *sequence of events*. Di sinilah *sequence* diagram memiliki keuntungan yang signifikan [1]. Representasi *Sequence* diagram dapat di lihat pada Gambar 2.9



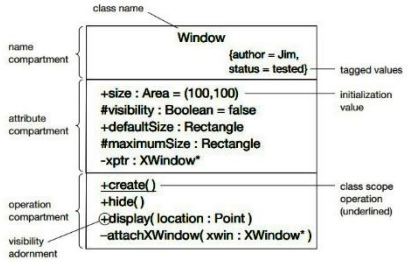
Gambar 2.9. Representasi *Sequence* Diagram dalam UML

Pada Gambar 2.9 aktor *Registrar* mengirim pesan *addCourse("UML")* ke :*RegistrationManager* yang dimana memanggil fungsi *addCourse()* dengan parameter "UML". Pada saat pengeksekusian operasi ini :*RegistrationManager* membuat *instance* baru dengan memanggil fungsi <<*create*>> pada UML:Course {new}

***Class* Diagram**

Sintaks UML visual sangatlah banyak modelnya dan penting untuk menerapkan gagasan UML dari *optional adornments* sehingga dapat dikelola dengan mudah. Satu-satunya bagian wajib dari sintaks visual adalah nama kompartemen dengan nama kelas di dalamnya. Semua kompartemen lain dan atribut pelengkap adalah opsional.

Kompartemen dan atribut pelengkap merupakan bagian pada kelas dalam *class* diagram. Jika hanya untuk menunjukkan hubungan antara berbagai kelas, maka cukup hanya dengan menggunakan kompartemen nama. Sedangkan, diagram yang digunakan untuk menggambarkan perilaku kelas ditambahkan kompartemen operasi dan operasi kunci pada masing-masing kelas [1]. Representasi *class* diagram ditunjukkan pada Gambar 2.10.

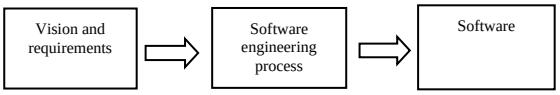


Gambar 2.10. Representasi Class Diagram dalam UML

Pada Gambar 2.10 menunjukan terdapat 3 bagian pada *class* diagram yaitu *name compartment*, *attribute compartment* dan *operation compartment*. *Name compartment* meliputi nama dari kelas itu sendiri yaitu *Window* dan *tagged values*. *Attribute compartment* digunakan untuk mendefinisikan nilai dari atribut-atribut yang ada pada kelas. *Operation compartment* berisi fungsi-fungsi yang ada pada kelas itu sendiri, contohnya pada kelas *Window* terdapat fungsi *create(), hide(), display(location: Point)* dan *attachXWindow(xwin: XWindow*)*.

***Unified Process* (UP)**

Proses pengembangan perangkat lunak atau yang biasa dikenal dengan *software engineering process* (SEP), mendefinisikan siapa, apa, kapan dan bagaimana cara mengembangkan sebuah perangkat lunak. Gambar 2.11 menunjukan bahwa SEP adalah proses yang mengubah *user requirements* menjadi sebuah perangkat lunak [1].

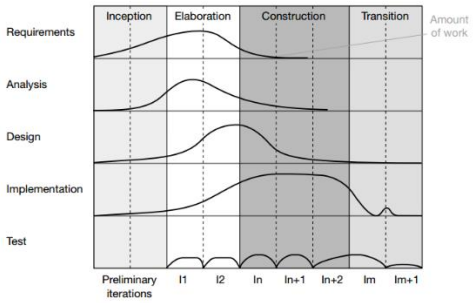


Gambar 2.11. Representasi SEP

Unified Process (UP) merupakan sebuah SEP yang mengubah *user requirements* menjadi sebuah perangkat lunak. Ada 5 alur kerja inti yang menentukan apa saja yang harus diselesaikan dan keahlian yang dibutuhkan untuk menyelesaikannya. Ke 5 alur kerja tersebut antara lain:

1. *Requirements* – memahami apa saja yang harus dapat dilakukan sistem
2. *Analysis* – memperbaiki dan membuat struktur dari *requirements*
3. *Design* – merealisasikan hasil analisis ke dalam desain sistem
4. *Implementation* – membangun perangkat lunak
5. *Testing* – menguji perangkat lunak

Representasi dari UP dapat di lihat pada Gambar 2.12



Gambar 2.12. Representasi *Unified Process*

Dalam fasenya UP terbagi menjadi 4 fase. Fase ini terdapat dalam sumbu horizontal yang menunjukan penyusunan dalam waktu, Fase tersebut antara lain:

1. *Inception Phase*
   Inception phase merupakan fase pertama dalam UP. Tujuan utama dari fase *inception* adalah untuk mendefinisikan lingkup proyek, mengestimasi biaya dan jadwal, mendefinisikan resiko, membuat *business use case* dan mengidentifikasi arsitektur.
2. *Elaboration Phase*
   Selama fase elaborasi dapat dilakukan spesifikasi *requirement* menjadi lebih rinci dan membuat arsitektur sistem. *Requirements* dirincikan untuk memastikan bahwa di sana ada pengertian dari lingkup pada masing-masing *requirements* untuk perencanaan berikutnya.
3. *Construction Phase*
   Fokus dari fase konstruksi adalah membuat kode program dengan menggunakan diagram visual yang didapatkan pada proses *analysis & design*. Produk yang telah selesai dibuat kemudian diuji untuk menjamin kualitas produk dengan harapan telah memenuhi *requirements*.
4. *Transition Phase*

Fase transisi fokus pada pengantaran sistem ke dalam produksi. Akan ada *testing* oleh *system testers* dan *end-users*, dan penyesuaian ulang dan penyetelan.

Setiap fasenya dapat berakhir dengan keputusan maju/tidak maju ke fase berikutnya, *stakeholders* harus memutuskan setuju untuk bergerak maju ke fase berikutnya atau berhenti dan membatalkan proyek. Proyek dapat dibatalkan apabila stakeholders tidak dapat menerima produk karena masalah kualitas [4]

Sebelum memasuki proses inti dari UP sendiri, *stakeholder* dan pengembang perangkat lunak mendefinisikan proses bisnis yang berjalan pada sistem dengan melakukan *business modeling*. Tujuan dari *business modeling* adalah untuk memahami bisnis dari organisasi, biasanya terbatas pada lingkup bisnis yang relevan dengan sistem yang akan dibangun. Bekerja dekat dengan *stakeholders*, diperoleh[3]:

a. Menilai status organisasi, termasuk kemampuan untuk mendukung sistem baru.
b. Menyelidiki proses bisnis, peran, dan tanggung jawab.
c. Mengembangkan domain model yang merefleksikan bisnis.

Untuk merepresentasikan permodelan bisnis serta alur kerja pada sistem dapat dibuat *flow diagram*.

Pada bagian sumbu vertikal pada UP terdapat *workflows* inti yang terbagi menjadi 5 disiplin utama. *Workflows* yang terdapat pada sumbu vertikal merupakan kerangka kerja sekuensial. Proses inti *workflows* tersebut antara lain:

1. Requirements

Tujuan dari *requirements* yaitu untuk memperoleh dokumen dan persetujuan akan lingkup dari apa yang harus dan yang tidak harus di buat pada sistem. Informasi tersebut akan digunakan oleh analis, desainer, dan *programmer* untuk membuat sistem. Aktifitas dari disiplin *requirements* antara lain [3]:

a. Bekerja sama dengan *stakeholders* untuk mengerti kebutuhan mereka
b. Mendefinisikan lingkup sistem
c. Mengeksplorasi kebutuhan, penggunaan, *business rule* serta kebutuhan *non-functional*.

Pada tahapan *requirements* kebutuhan sistem akan dihubungkan dengan aktor yang berinteraksi dengan kebutuhan tersebut, hal ini dapat direpresentasikan dengan diagram UML yaitu *use case diagram*.

2. Analysis and Design

Tujuan dari disiplin ini yaitu untuk melakukan analisis pada *requirements* untuk sistem dan untuk melakukan desain solusi agar dapat diimplementasikan. Aktifitas kritikal pada disiplin ini antara lain [3]:

a. Merencanakan dan membuat arsitektur sistem
b. Memahami (menganalisasi) kebutuhan sistem
c. Membuat desain komponen, *services* dan/atau modul

Representasi yang dibuat pada proses *analysis and design* dapat berupa diagram UML *sequence diagram* dan *class diagram*. Diagram-diagram tersebut akan mendefinisikan modul-modul yang akan dibuat serta alur kerja dari masing-masing modul tersebut.

3. Implementation

Tujuan dari disiplin ini yaitu untuk melakukan transformasi dari desain ke dalam kode program. Aktifitas inti pada disiplin ini antara lain [3]:

a. Memahami desain model
b. Menulis kode program
c. Mengimplementasi komponen, *services* dan/atau modul.

Output dari tahapan *implementation* berupa produk perangkat lunak/sistem yang telah dibuat menggunakan acuan dari dokumen *analysis and design*.

4. Test

Tujuan dari disiplin pengujian adalah untuk melakukan evaluasi dan memastikan kualitas sistem yang telah di buat. Disiplin ini juga termasuk menemukan kerusakan, memastikan sistem telah berjalan sesuai desain, dan memastikan sistem telah mencapai *requirements* yang telah ditentukan. Aktifitas kritikal pada disiplin ini antara lain [3]:

a. Mendefinisikan dan merencanakan *testing.*
b. Membuat *test case*
c. Melaksanakan *testing*
d. Mendokumentasikan *testing*

Hasil akhir dari tahapan *test* ini berupa dokumen pengujian. Dokumen berisi *test-case* atau kasus uji dari masing-masing *requirement* yang telah didefinisikan.

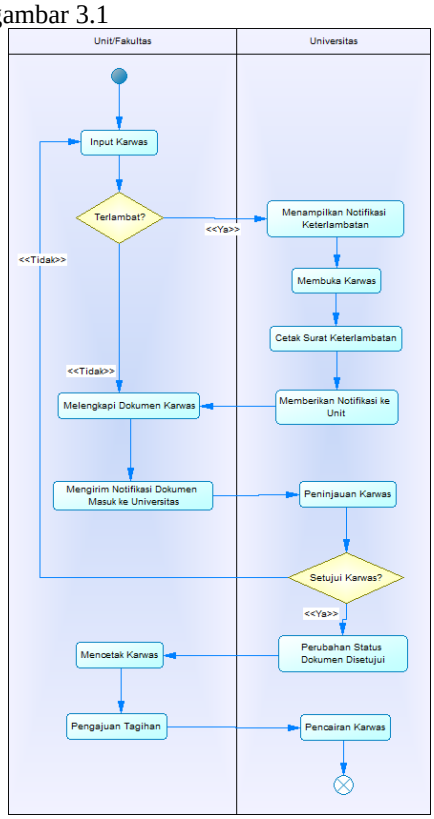
3. Pembahasan

Kartu Pengawasan (KARWAS) adalah nama yang diberikan untuk perangkat lunak *monitoring* data kontrak. Untuk mengembangkan KARWAS maka hal yang harus dilakukan adalah mengerti terlebih dahulu proses bisnis serta *business rule* yang ada pada pengajuan proposal kontrak sampai ke rekapitulasi laporan Pengumpulan Data. Tahapan pada proses bisnis data kontrak adalah:.

1. Unit Kerja/Fakultas membuat/memasukan dokumen KARWAS. Dokumen yang dimasukan ke dalam sistem berupa formulir yang memiliki berbagai atribut yang merupakan ketentuan dalam memasukan data kontrak. Contoh formulir *input* dalam data kontrak KARWAS dapat dilihat pada Lampiran 1.
2. Melakukan pengecekan apakah dokumen yang dibuat terlambat atau tidak. Dokumen KARWAS dikatakan terlambat apabila tanggal kontrak yang ada pada dokumen sudah lebih 2 hari dibandingkan hari pengisian KARWAS pada sistem.
3. Apabila dokumen terlambat maka system akan mengirimkan notifikasi ke universitas dan harus mencetak surat keterlambatan yang menjadi syarat agar dokumen dapat dibuka oleh pihak universitas dan kemudian dilengkapi oleh Unit. Dokumen KARWAS tidak dapat dilengkapi oleh pihak Unit/ fakultas apabila pihak universitas belum membuka dokumen terlambat tersebut.
4. Apabila dokumen tidak terlambat maka Unit melengkapi dokumen KARWAS dengan mengisi detail kegiatan, rekanan dan termin pembayaran. Kemudian sistem akan mengirimkan notifikasi data kontrak baru ke universitas.
5. Dokumen yang telah masuk ditinjau oleh pihak universitas dan menentukan apakah dokumen tersebut disetujui atau ditolak.
6. Apabila dokumen disetujui oleh pihak universitas maka status data kontrak akan berubah menjadi disetujui dan dilakukan pencetakan dokumen KARWAS. Hasil cetak dokumen KARWAS dapat dilihat pada Lampiran 2. Setelah melakukan pencetakan, pihak Unit mengajukan penagihan dan pihak universitas melengkapi realisasi pembayaran.

7. Apabila dokumen ditolak oleh pihak universitas maka dokumen akan dikembalikan ke pihak Unit untuk di revisi.

Representasi proses bisnis KARWAS dapat dilihat pada gambar 3.1



Gambar 3.1 Representasi proses bisnis KARWAS

Requirements

Kebutuhan perangkat lunak merupakan kondisi atau kemampuan yang harus dimiliki oleh perangkat lunak. Dalam sub bab ini dijelaskan definisi kebutuhan perangkat lunak yang meliputi deskripsi umum perangkat lunak, model *use case*, dan kebutuhan non-*functional* perangkat lunak. Perangkat lunak *monitoring* data kontrak ini memiliki tiga aktor, yaitu Admin, Unit, dan Universitas. untuk lebih detailnya bisa dilihat pada Tabel 3.1.

Tabel 3.1. Daftar Aktor

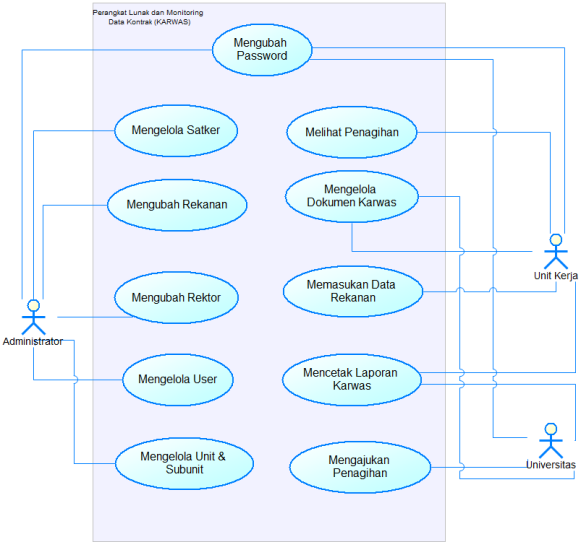
No	Aktor	Deskripsi
1	Admin	Admin merupakan pengelola perangkat lunak *monitoring* data kontrak
2	Unit	Unit merupakan Fakultas yang mengelola data dokumen karwas
3	Universitas	Universitas melakukan peninjauan dan verifikasi terhadap dokumen karwas

Perangkat lunak ini memiliki beberapa *use case*, untuk lebih detailnya dapat dilihat pada Tabel 3.2.

Tabel 3.2. Daftar *Use case*

No	*Use case*	Deskripsi
1	Mengelola Dokumen Karwas	Unit dan Universitas mengelola dokumen karwas
2	Mencetak Laporan Karwas	Unit dan Universitas mencetak laporan karwas
3	Mengajukan Penagihan	Universitas mengajukan penagihan anggaran ke Unit
4	Melihat Penagihan	Unit melihat daftar penagihan yang diajukan oleh Universitas
5	Mengelola Satker	Admin mengelola Satuan Kerja
6	Memasukan Data Rekanan	Unit memasukan data rekanan
7	Mengubah Rekanan	Admin mengubah data Rekanan kontrak
8	Mengubah Rektor	Admin mengubah data Rektor
9	Mengelola Unit & Subunit	Admin mengelola Unit dan Subunit
10	Mengelola *User*	Admin mengelola *User*/Pengguna
11	Mengubah *Password*	Unit, Universitas dan Admin dapat melakukan perubahan *password*

Use case Diagram terbentuk dari serangkaian aktor dan *use case* yang ada pada aplikasi. Dari daftar aktor dan *use case* tersebut diperoleh *use case* Diagram dari perangkat lunak *monitoring* data kontrak studi seperti yang ditunjukkan pada Gambar 3.2



Gambar 3.2 Use case Diagram

4. Eksperimen

KARWAS dibuat berdasarkan rancangan yang telah dilakukan. Hasil implementasi aplikasi ini adalah sebagai berikut :

Spesifikasi Perangkat

Spesifikasi perangkat laptop yang digunakan dalam membangun KARWAS adalah sebagai berikut :

- *Processor*: Intel CORE i3-3217U 1.80GHz
- RAM : 4 GB
- *Hardisk* : 450 GB

Sedangkan perangkat lunak yang digunakan untuk membangun KARWAS adalah sebagai berikut :

- OS Windows 7
- *Web* Server Apache 2.4.16
- PHP Versi 5.6.12
- DBMS MySQL
- *Browser* Chrome
- Sublime *Text Editor*

Tabel 4.1. *Class* Implementasi *web*

No	Nama *Class* Desain	*Class* Implementasi
1	satker_tambah_page	…/…/*Views*/ admin/satker/satker_tambah_page.php
2	satker_list_page	…/…/*Views*/ admin/satker/satker_list_page.php
3	satker_ubah_page	…/…/*Views*/ admin/satker/satker_ubah _page.php
4	subunit_tambah_page	…/…/*Views*/ admin/subunit/ subunit_tambah _page.php
5	subunit_list_page	…/…/*Views*/ admin/subunit/ subunit_list_page.php
6	subunit_ubah_page	…/…/*Views*/ admin/subunit/ subunit_ubah_page.php
7	unit_tambah_page	…/…/*Views*/ admin/unit/ unit_tambah _page.php
8	unit_list_page	…/…/*Views*/ admin/unit/ unit_list_page.php
9	unit_ubah_page	…/…/*Views*/ admin/unit/ unit_ubah_page.php
10	dokumen_tambah_page	…/…/*Views*/ dokumen_tambah_page.php
11	dokumen_list_page	…/…/*Views*/ dokumen_list_page.php
12	dokumen_ubah_page	…/…/*Views*/ dokumen_ubah_page.php
13	user_tambah_page	…/…/*Views*/ admin/user/user_tambah_page.php
14	user_list_page	…/…/*Views*/ admin/user/user_list_page.php
15	user_ubah_page	…/…/*Views*/ admin/user/user_ubah_page.php
16	laporan_page	…/…/*Views*/laporan_page.php
17	penagihan_page	…/…/*Views*/penagihan_page.php
18	rekanan_ubah_page	…/…/*Views*/admin/rekanan/rekanan_ubah_page.php
19	rektor_ubah_page	…/…/*Views*/admin/rektor/rektor_ubah_page.php
20	my_profil_page	…/…/*Views*/my_profil_page.php
21	DokumenController	…/…/*Controllers*/Dokumen.php
22	PenggunaController	…/…/*Controllers*/Pengguna.php
23	TerlambatController	…/…/*Controllers*/Terlambat.php
24	SatkerController	…/…/*Controllers*/Satker.php
25	UnitController	…/…/*Controllers*/Unit.php
26	RekananController	…/…/*Controllers*/Rekanan.php
27	RektorController	…/…/*Controllers*/Rektor.php
28	Mysql_service	…/…/*Models*/Mysql_service.php
29	Karwas	…/…/*Models*/Karwas.php
30	Satker	…/…/*Models*/Satker.php
31	Rekanan	…/…/*Models*/Rekanan.php

No	Nama *Class* Desain	*Class* Implementasi
32	Rektor	…/…/*Models*/Rektor.php
33	Subunit	…/…/*Models*/Subunit.php
34	Unit	…/…/*Models*/Unit.php
35	User	…/…/*Models*/User.php
36	Termin_pembayaran	…/…/*Models*/Termin_pembayaran.php
37	Kegiatan	…/…/*Models*/Kegiatan.php
38	Output	…/…/*Models*/Output.php
39	Subouput	…/…/*Models*/Suboutput.php
40	Komponen_input	…/…/*Models*/Komponen_input.php
41	Subkomponen_input	…/…/*Models*/ Subkomponen_input.php
42	Akun_belanja	…/…/*Models*/ Akun_belanja.php
43	Detail_belanja	…/…/*Models*/ Detail_belanja.php

5. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian mengenai pengembangan perangkat lunak *monitoring* data kontrak studi kasus BAUK Universitas Diponegoro adalah :

1. Permodelan KARWAS menggunakan UP menghasilkan kartu pengawasan kontrak yang didalamnya terdapat satuan kerja, detail kontrak, rekanan, dan termin pembayaran.
2. KARWAS memfasilitasi proses kelola data kontrak sehingga pihak unit tidak kesulitan dalam melakukan proses pengisian *form* data kontrak dan pihak universitas mudah dalam meninjau data kontrak dan melengkapi realisasi termin pembayaran.
3. KARWAS membangun *data history* dari data kontrak yang pernah diajukan sehingga data kontrak dapat ditinjau kembali dan dilakukan rekapitulasi pencetakan laporan.
4. KARWAS membantu pihak BAUK maupun unit Universitas Diponegoro dalam melakukan *monitoring* data kontrak.

DAFTAR PUSTAKA

[1] Arlow, J. & Neustadt, I., 2002. *UML and the Unified Process Practical object-oriented analysis and design*. Great Britain: Pearson Education Limited.

[2] Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide 2nd Edition.* Boston, USA: Addison Wesley Proffesional.

[3] Ambler, S.W., 2005. *A Manager's Introduction to The Rational Unified Process (RUP)*. Ambysoft.

[4] Rational Software, 1998. *Rational Unified Process Best Practices for Software Development Teams*.