

Pengembangan *Dashboard* Universitas Ibnu Sina Batam Berbasis *Web Mobile*

Lastri Yanti

^{1,2}Universitas Ibnu Sina; Jalan Teuku Umar - Lubuk Baja, Batam, Kepulauan Riau, Telp. 0778 – 408 3113

³Program Studi Teknik Informatika, Fakultas Teknik – Universitas Ibnu Sina, Batam
e-mail: *¹ Lastri@uis.ac.id

Abstrak

Penelitian ini bertujuan untuk mendukung pengembangan kampus digital Universitas Ibnu Sina Batam dengan merancang dan mengembangkan *Dashboard* Universitas Ibnu Sina. Objek pada penelitian ini dilakukan di kampus Universitas Ibnu Sina Batam di bagian Biro Administrasi dan Akademik dan Kemahasiswaan (BAAK). Metode penelitian yang digunakan dalam ini adalah metode *Object Oriented Analysis and Design (OOAD)* yang meliputi kebutuhan, analisa dan desain, implementasi, pengujian, dan deployment. Dengan menggunakan permodelan UML. Sistem informasi yang dirancang dapat memberikan informasi yang real time dan akurat. Penelitian ini dimaksudkan mampu merancang aplikasi berbasis web mobile. Sehingga dapat diakses oleh seluruh civitas akademika di kampus Universitas Ibnu Sina dan mempermudah dalam pengaksesan informasi Universitas Ibnu Sina melalui internet.

Kata kunci— *Dashboard*, *Object Oriented Analysis and Design (OOAD)*, dan *Unified Modeling Language (UML)*.

Abstract

This study aims to support the development of University Ibnu Sina digital campus by designing and developing the University Ibnu Sina *Dashboard* Batam. Objects in this study were conducted at University Ibnu Sina campus in Batam in the Administrative and Academic Student Affairs Bureau (BAAK). The research method used in this thesis is the method of *Object Oriented Analysis and Design (OOAD)* which includes needs, analysis and design, implementation, testing, and deployment. Using UML modeling. The information system that is designed can provide real time and accurate information. This study is intended to be able to design mobile web-based applications. So that it can be accessed by the entire academic community on the Ibnu Sina University campus in Batam and makes it easier to access University Ibnu Sina information via the internet.

Keywords— *Dashboard*, *Object Oriented Analysis and Design (OOAD)*, and *Unified Modeling Language (UML)*.

PENDAHULUAN

Pengelolaan dan penyajian informasi bukanlah hal yang mudah, mengingat kompleksitas dan banyaknya informasi yang dimiliki organisasi. Organisasi memerlukan sebuah alat untuk mengelola informasi dan menyajikannya dalam bentuk yang efisien dan efektif, yakni *dashboard* (Sistem Panel). Menurut Few dalam jurnal (Ropianto, 2017) Istilah *Dashboard*

(Sistem Panel) merupakan alat untuk menyajikan informasi secara sekilas, solusi bagi kebutuhan informasi organisasi.

Pada penelitian terdahulu telah ada perancangan *Dashboard digital* untuk kampus Universitas Ibnu Sina Batam. *Dashboard digital* adalah sebuah sistem informasi yang menyediakan layanan satu pintu yang menggabungkan beberapa informasi penting yang dibutuhkan oleh civitas akademik (Staff, Dosen dan Mahasiswa). Aplikasi *Dashboard* dengan potensinya dapat dioptimalkan dengan beberapa langkah, salah satunya dengan proses digitalisasi dan penayangan melalui *Smart TV*. *Dashboard* juga dapat diakses melalui berbagai peralatan komunikasi seperti komputer, *tablet*, *smartphone*, sehingga kreasi ini bisa disebut sebagai *dashboard digital* (Prasetya, 2017).

Penerapan sistem informasi *dashboard* (Sistem Panel) ini diharapkan dapat memudahkan dalam memberikan berbagai informasi penting dan berkualitas sekitar kegiatan perkuliahan dan kegiatan non-akademik yang dibutuhkan civitas akademik dalam satu layar yang tertata rapi, aktual, selalu *update* dan tentunya mudah dipahami bagi seluruh kalangan di Universitas Ibnu Sina.

Informasi akademik dan perkuliahan menjadi kebutuhan yang penting bagi mahasiswa. Seperti informasi tentang dosen berhalangan hadir, informasi seputar kampus, dan sebagainya. Dengan informasi yang terdistribusi dengan baik, maka tidak akan terjadi kesalahpahaman bahkan buntunya penyebaran informasi yang bersifat penting. Informasi biasanya disebarakan secara *online* melalui beberapa media seperti *e-mail*, *classroom*, dan *website*. Ada kelemahan yang timbul dengan cara tersebut, seperti *classroom* yaitu kita harus memiliki *email* kampus, *download* aplikasi *classroom* dan *login* terlebih dahulu, lalu kita harus masuk *join class* untuk mendapatkan informasi yang di sebarakan di *classroom* sehingga tidak semua orang bisa melihat informasi tersebut.

METODE PENELITIAN

2.1 *Dashboard*

Menurut Few pada jurnal (Ropianto, 2017) Istilah *information dashboard* didefinisikan sebagai tampilan visual dari informasi penting yang diperlukan untuk mencapai satu atau beberapa tujuan, digabungkan dan diatur pada sebuah layar, menjadi informasi yang dibutuhkan dan dapat dilihat secara sekilas. Tampilan visual disini mengandung pengertian bahwa penyajian informasi harus dirancang sebaik mungkin, sehingga mata manusia dapat menangkap informasi secara dan otak manusia dapat memahami maknanya secara benar. *Dashboard* itu sebuah tampilan pada satu monitor komputer penuh, yang berisi informasi yang bersifat kritis, agar kita dapat melihatnya dengan segera, sehingga dengan melihat *dashboard* itu saja, kita dapat mengetahui hal-hal yang perlu diketahui. Biasanya kombinasi dari teks dan grafik, tetapi lebih ditekankan pada grafik.

2.2 Metode Pengumpulan Data

Walsham (1995) yang dikutip oleh (Gunawan, 2016) menjelaskan bahwa metode pengumpulan data penelitian kualitatif menggunakan metode wawancara secara mendalam.

Tahap-tahan pengumpulan data dalam penelitian ini adalah sebagai berikut:

a. Studi Pustaka

Metode studi pustaka adalah suatu teknik atau cara pengumpulan data atau informasi yang dilakukan dengan cara membaca buku-buku, laporan-laporan yang berhubungan dengan masalah terkait.

b. Wawancara

Wawancara adalah metode pengumpulan data yang dilakukan dengan cara tanya jawab secara langsung dengan Biro Administrasi Akademik dan Kemahasiswaan (BAAK).

c. Observasi

Metode observasi adalah salah satu teknik pengumpulan data atau informasi yang dilakukan dengan cara mengadakan pengamatan langsung pada objek permasalahan (Kampus STT Ibnu Sina Batam) dan kemudian dari pengamatan tersebut diambil suatu kesimpulan.

d. Dokumentasi

Metode dokumentasi adalah metode pengumpulan data yang berupa catatan, transkrip, buku, surat kabar, majalah, notulen rapat, *legger*, agenda dan sebagainya yang ada di Kampus STT Ibnu Sina Batam.

2.3 Metode Analisa Data

Data yang telah diolah kemudian dianalisis dengan menggunakan analisis deskriptif kualitatif. Analisis deskriptif kualitatif adalah analisis yang diwujudkan dengan cara menggambarkan kenyataan atau keadaan-keadaan atas suatu obyek dalam bentuk uraian kalimat berdasarkan keterangan-keterangan dari pihak-pihak yang berhubungan langsung dengan penelitian ini. Hasil analisis tersebut kemudian diinterpretasikan guna memberikan gambaran yang jelas terhadap permasalahan yang diajukan mengenai Pengembangan *Dashboard* STT Ibnu Sina Batam Berbasis *Web Mobile*.

2.4 Metode Pengolahan Data

Metode Pengolahan Data yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Analisa Sistem
2. Pengumpulan Data
3. Perancangan Sistem *Unified Modelling Language* (UML)
4. Perancangan Basis Data
5. Pengujian dan Perbaikan Aplikasi
6. Implementasi

2.5 Metode Perancangan Sistem

Menurut Booch's yang dikutip dalam Prasetyotomo (2015) Metodologi Booch's *Object Oriented Analysis & Design*, selanjutnya disingkat OOAD, merupakan penggabungan 3 (tiga) pendekatan, yaitu: *Object Oriented Design* (OOD), *Object Oriented Analysis* (OOA) dan *Object Oriented Programming* (OOP).

OOA merupakan metode analisis yang memeriksa kebutuhan (*requirement*) berdasarkan prespektif pengumpulan obyek-obyek dan kelas-kelas dalam sebuah domain problem, OOD merupakan sebuah metode mendesain yang mencakup proses pendekomposisian obyek dan digambarkan dalam notasi sehingga bisa menggambarkan static dan dynamic model sistem baik secara *logical* dan/atau *physical*, sedangkan OOP merupakan sebuah metode untuk mengimplementasikan program yang diorganisasikan sebagai kumpulan obyek dimana tiap-tiap obyek merupakan instan dari sebuah kelas dan kelas merupakan salah satu dari kumpulan kelas yang saling berhubungan secara hirarki menggunakan *inheritance relationship*.

Hubungan antara OOA, OOD dan OOP adalah: hasil pemodelan atau pengumpulan obyek dari OOA akan digunakan oleh OOD dan hasil dari OOD akan digunakan sebagai blueprint untuk membangun sistem dengan menggunakan OOP.

Tahapan dalam proses OOAD adalah sebagai berikut:

a. *Requirement*

Pada tahap *requirement* menjelaskan tentang bagaimana membuat dan menjaga sebuah perjanjian kerjasama dengan *customer* dan stakeholder mengenai apa yang harus dilakukan pada pembuatan sistem, misalnya kebutuhan-kebutuhan dari suatu sistem seperti fungsi-fungsi apa saja yang akan digunakan oleh sistem, fitur-fitur apa saja yang akan dipakai dalam pembuatan sistem, siapa saja user yang terlibat atau yang memakai sistem tersebut dan lain sebagainya.

b. *Analysis and Design*

Tahapan ini menjelaskan bagaimana mengkonversi kebutuhan-kebutuhan sistem yang telah dibuat sebelumnya menjadi sebuah bentuk rancangan sistem. Yang mana rancangan tersebut disajikan sebagai spesifikasi dari implementasi sistem dalam memilih lingkungan penerapannya. Selain itu, tahapan analisis dan desain juga mencakup dalam mengembangkan sebuah arsitektur/bentuk rancangan yang kokoh.

c. *Implementation*

Setelah membuat suatu kebutuhan sistem, menganalisis dan merancang sistem yang akan dibuat. Kemudian pada tahap ini dilakukan implementasi unit sistem dan menggabungkan rancangan dengan sistem, maksudnya adalah mengimplementasikan rancangan tersebut dalam sebuah sistem atau program dalam bentuk *coding* program. Selain itu juga menghasilkan sebuah sistem yang sudah dapat dijalankan.

d. *Testing*

Tes/pengujian dilakukan untuk meyakinkan bahwa sistem telah sesuai dengan kebutuhan-kebutuhan yang telah dibuat sebelumnya (kebutuhan-kebutuhan tersebut adalah menerapkan yang sewajarnya). Memvalidasi fungsi sistem yang telah konkrit kemudian didemonstrasikan bahwa produk *software* tersebut sesuai dengan kebutuhan dan rancangan yang telah dibuat sebelumnya.

e. *Deployment*

Meyakinkan bahwa produk *software* tersebut (termasuk implementasi dan pengujian) telah tersedia untuk *end user* atau sudah dapat digunakan oleh pengguna akhir.

2.6 Metode Pemodelan Sistem

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual (Braun,et.al,2001) yang dikutip dalam Ropianto (2016). Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek (Whitten,et.al,2004) yang dikutip dalam Ropianto (2016).

Jenis-jenis Diagram Pada UML

a. *Use Case Diagram*

Use Case diagram merupakan uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. Use Case atau diagram use case merupakan pemodelan untuk melakukan (behavior) sistem informasi yang akan dibuat (Rossa M.Salahuddin, 2011) yang dikutip dalam Nuddin & Fithri (2015).

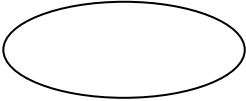
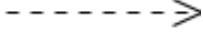
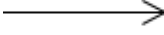
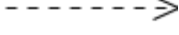
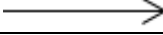
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, membuat sebuah daftar belanja, dan sebagainya. Sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

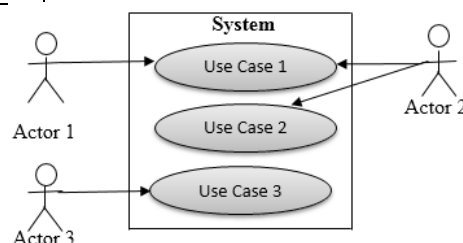
Use case diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua fitur yang ada pada sistem.

Sebuah *use case* dapat meng-include fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-include akan dipanggil setiap kali *use case* yang meng-include dieksekusi secara normal. Sebuah *use case* dapat di-include oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah use case juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Tabel 1 Simbol Use case Diagram (A.S & Shalahuddin 2011:130) yang dikutip oleh Aprianti & Maliha (2017)

GAMBAR	DESKRIPSI
<i>Use case</i> 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal-awal frase nama <i>use case</i> .
Aktor/ <i>Actor</i> 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari actor adalah gambar orang, tapi actor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Asosiasi/ <i>Association</i> 	Komunikasi actor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan actor.
Ekstensi/ <i>Extend</i> <<Extend>> 	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek.
Generalisasi/ <i>Generalization</i> 	Hubungan generalisasi dan spesialisasi (umum- khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari yang lain.
Menggunakan/ <i>Include/Uses</i> <<include>>  <<uses>> 	Relasi <i>use case</i> tambahan kesebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini.



Gambar 1 Contoh Use Case Diagram (Ropianto, 2016)

b. Activity diagram

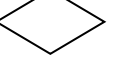
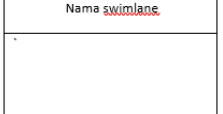
Activity Diagram menggambarkan workflow (aliran kerja) dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. (Aprianti & Maliha, 2017) *Activity diagram*

merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik inkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

Tabel 2 Simbol *Activity diagram* (A.S & Shalahuddin 2011:134)

yang dikutip oleh Aprianti & Maliha (2017)

GAMBAR	DESKRIPSI
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan/ <i>decision</i> 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan/ <i>join</i> 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
<i>Swimlane</i> 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas

c. *Class Diagram*

Diagram kelas atau *Class diagram* menggambarkan sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. (Rossa M.Salahuddin, 2011) pada jurnal (Nuddin & Fithri, 2015).

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok:

1. Nama (dan *stereotype*)

2. Atribut
3. Metode

Atribut dan metoda dapat memiliki salah satu sifat berikut:

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan

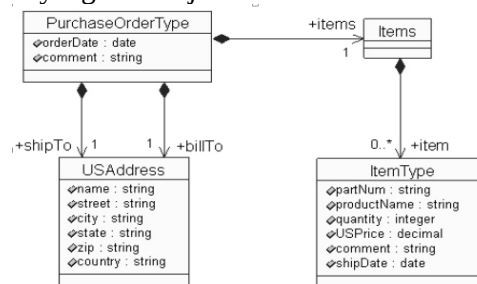
dan anak-anak yang mewarisinya.

3. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Berikut ini merupakan hubungan antar *class*:

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas”).
3. Pewarisan, yaitu hubungan hirarki antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

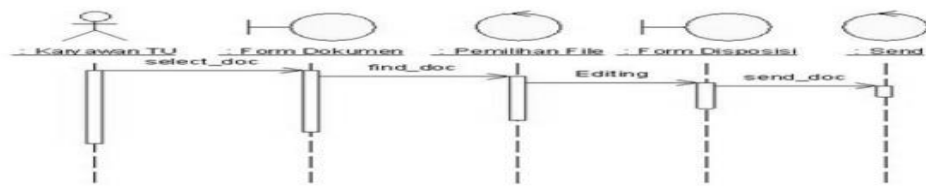


Gambar 2. Contoh Class Diagram (Ropianto, 2016)

d. Sequence Diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan use case diagram (Ropianto, 2016). *Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.



Gambar 3 Notasi *Sequence Diagram* (Ropianto, 2016)

e. *User Interface*

Merupakan komponen penting dari setiap sistem baru yang akan digunakan. *User Interface* merupakan bagian dari sistem informasi yang memerlukan interaksi langsung pada sistem *user* dalam membuat *input* dan *output*. Dalam membuat *user interface* terdapat aturan yang disebut *eight golden rules*. *Eight golden rules* bermanfaat dalam merancang *user interface* yang interaktif.

2.1 *Dashboard*

Menurut Few pada jurnal (Ropianto, 2017) Istilah *information dashboard* didefinisikan sebagai tampilan visual dari informasi penting yang diperlukan untuk mencapai satu atau beberapa tujuan, digabungkan dan diatur pada sebuah layar, menjadi informasi yang dibutuhkan dan dapat dilihat secara sekilas. Tampilan visual disini mengandung pengertian bahwa penyajian informasi harus dirancang sebaik mungkin, sehingga mata manusia dapat menangkap informasi secara dan otak manusia dapat memahami maknanya secara benar. *Dashboard* itu sebuah tampilan pada satu monitor komputer penuh, yang berisi informasi yang bersifat kritis, agar kita dapat melihatnya dengan segera, sehingga dengan melihat *dashboard* itu saja, kita dapat mengetahui hal-hal yang perlu diketahui. Biasanya kombinasi dari teks dan grafik, tetapi lebih ditekankan pada grafik.

2.2 *Metode Pengumpulan Data*

Walsham (1995) yang dikutip oleh (Gunawan, 2016) menjelaskan bahwa metode pengumpulan data penelitian kualitatif menggunakan metode wawancara secara mendalam. Tahap-tahan pengumpulan data dalam penelitian ini adalah sebagai berikut:

e. Studi Pustaka

Metode studi pustaka adalah suatu teknik atau cara pengumpulan data atau informasi yang dilakukan dengan cara membaca buku-buku, laporan-laporan yang berhubungan dengan masalah terkait.

f. Wawancara

Wawancara adalah metode pengumpulan data yang dilakukan dengan cara tanya jawab secara langsung dengan Biro Administrasi Akademik dan Kemahasiswaan (BAAK).

g. Observasi

Metode observasi adalah salah satu teknik pengumpulan data atau informasi yang dilakukan dengan cara mengadakan pengamatan langsung pada objek permasalahan (Kampus STT Ibnu Sina Batam) dan kemudian dari pengamatan tersebut diambil suatu kesimpulan.

h. Dokumentasi

Metode dokumentasi adalah metode pengumpulan data yang berupa catatan, transkrip, buku, surat kabar, majalah, notulen rapat, *legger*, agenda dan sebagainya yang ada di Kampus STT Ibnu Sina Batam.

2.3 *Metode Analisa Data*

Data yang telah diolah kemudian dianalisis dengan menggunakan analisis deskriptif kualitatif. Analisis deskriptif kualitatif adalah analisis yang diwujudkan dengan cara menggambarkan kenyataan atau keadaan-keadaan atas suatu obyek dalam bentuk uraian kalimat

berdasarkan keterangan-keterangan dari pihak-pihak yang berhubungan langsung dengan penelitian ini. Hasil analisis tersebut kemudian diinterpretasikan guna memberikan gambaran yang jelas terhadap permasalahan yang diajukan mengenai Pengembangan *Dashboard STT Ibnu Sina Batam Berbasis Web Mobile*.

2.4 Metode Pengolahan Data

Metode Pengolahan Data yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Analisa Sistem
2. Pengumpulan Data
3. Perancangan Sistem *Unified Modelling Language* (UML)
4. Perancangan Basis Data
5. Pengujian dan Perbaikan Aplikasi
6. Implementasi

2.7 Metode Perancangan Sistem

Menurut Booch's yang dikutip dalam Prasetyotomo (2015) Metodologi Booch's *Object Oriented Analysis & Design*, selanjutnya disingkat OOAD, merupakan penggabungan 3 (tiga) pendekatan, yaitu: *Object Oriented Design* (OOD), *Object Oriented Analysis* (OOA) dan *Object Oriented Programming* (OOP).

OOA merupakan metode analisis yang memeriksa kebutuhan (*requirement*) berdasarkan perspektif pengumpulan obyek-obyek dan kelas-kelas dalam sebuah domain problem, OOD merupakan sebuah metode mendesain yang mencakup proses pendekomposisi obyek dan digambarkan dalam notasi sehingga bisa menggambarkan static dan dynamic model sistem baik secara *logical* dan/atau *physical*, sedangkan OOP merupakan sebuah metode untuk mengimplementasikan program yang diorganisasikan sebagai kumpulan obyek dimana tiap-tiap obyek merupakan instan dari sebuah kelas dan kelas merupakan salah satu dari kumpulan kelas yang saling berhubungan secara hirarki menggunakan *inheritance relationship*.

Hubungan antara OOA, OOD dan OOP adalah: hasil pemodelan atau pengumpulan obyek dari OOA akan digunakan oleh OOD dan hasil dari OOD akan digunakan sebagai blueprint untuk membangun sistem dengan menggunakan OOP.

Tahapan dalam proses OOAD adalah sebagai berikut:

c. Requirement

Pada tahap *requirement* menjelaskan tentang bagaimana membuat dan menjaga sebuah perjanjian kerjasama dengan *customer* dan stakeholder mengenai apa yang harus dilakukan pada pembuatan sistem, misalnya kebutuhan-kebutuhan dari suatu sistem seperti fungsi-fungsi apa saja yang akan digunakan oleh sistem, fitur-fitur apa saja yang akan dipakai dalam pembuatan sistem, siapa saja user yang terlibat atau yang memakai sistem tersebut dan lain sebagainya.

d. Analysis and Design

Tahapan ini menjelaskan bagaimana mengkonversi kebutuhan-kebutuhan sistem yang telah dibuat sebelumnya menjadi sebuah bentuk rancangan sistem. Yang mana rancangan tersebut disajikan sebagai spesifikasi dari implementasi sistem dalam memilih lingkungan penerapannya. Selain itu, tahapan analisis dan desain juga mencakup dalam mengembangkan sebuah arsitektur/bentuk rancangan yang kokoh.

e. Implementation

Setelah membuat suatu kebutuhan sistem, menganalisis dan merancang sistem yang akan dibuat. Kemudian pada tahap ini dilakukan implementasi unit sistem dan menggabungkan rancangan dengan sistem, maksudnya adalah mengimplementasikan rancangan tersebut dalam sebuah sistem atau program dalam bentuk *coding* program. Selain itu juga menghasilkan sebuah sistem yang sudah dapat dijalankan.

f. Testing

Tes/pengujian dilakukan untuk meyakinkan bahwa sistem telah sesuai dengan kebutuhan-kebutuhan yang telah dibuat sebelumnya (kebutuhan-kebutuhan tersebut adalah menerapkan

yang sewajarnya). Memvalidasi fungsi sistem yang telah konkrit kemudian didemonstrasikan bahwa produk *software* tersebut sesuai dengan kebutuhan dan rancangan yang telah dibuat sebelumnya.

g. *Deployment*

Meyakinkan bahwa produk *software* tersebut (termasuk implementasi dan pengujian) telah tersedia untuk *end user* atau sudah dapat digunakan oleh pengguna akhir.

2.8 Metode Pemodelan Sistem

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual (Braun,et.al,2001) yang dikutip dalam Ropianto (2016). Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek (Whitten,et.al,2004) yang dikutip dalam Ropianto (2016).

Jenis-jenis Diagram Pada UML

f. *Use Case Diagram*

Use Case diagram merupakan uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. *Use Case* atau diagram *use case* merupakan pemodelan untuk melakukan (behavior) sistem informasi yang akan dibuat (Rossa M.Salahuddin, 2011) yang dikutip dalam Nuddin & Fithri (2015).

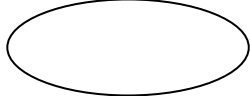
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, membuat sebuah daftar belanja, dan sebagainya. Sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

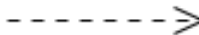
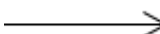
Use case diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua fitur yang ada pada sistem.

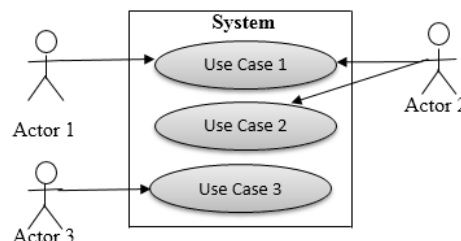
Sebuah *use case* dapat meng-include fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-include akan dipanggil setiap kali *use case* yang meng-include dieksekusi secara normal. Sebuah *use case* dapat di-include oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-extend *use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Tabel 1 Simbol *Use case Diagram* (A.S & Shalahuddin 2011:130) yang dikutip oleh Aprianti & Maliha (2017)

GAMBAR	DESKRIPSI
	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal-awal frase nama <i>use case</i> .

GAMBAR	DESKRIPSI
Aktor/Actor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari actor adalah gambar orang, tapi actor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Asosiasi/Association 	Komunikasi actor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan actor.
Ekstensi/Extend <<Extend>> 	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek.
Generalisasi/Generalization 	Hubungan generalisasi dan spesialisasi (umum- khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari yang lain.
Menggunakan/Include/Uses <<include>>  <<uses>> 	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini.



Gambar 1 Contoh Use Case Diagram (Ropianto, 2016)

g. Activity diagram

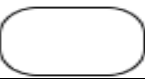
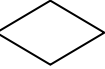
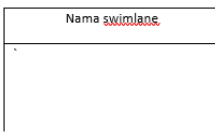
Activity Diagram menggambarkan workflow (aliran kerja) dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. (Aprianti & Maliha, 2017) *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-trigger oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik inkronisasi yang dapat berupa titik, garis

horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

Tabel 2 Simbol *Activity diagram* (A.S & Shalahuddin 2011:134)

yang dikutip oleh Aprianti & Maliha (2017)

GAMBAR	DESKRIPSI
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan/decision 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan/join 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
<i>Swimlane</i> 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas

h. *Class Diagram*

Diagram kelas atau *Class diagram* menggambarkan sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. (Rossa M.Salahuddin, 2011) pada jurnal (Nuddin & Fithri, 2015).

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok:

4. Nama (dan *stereotype*)
5. Atribut
6. Metode

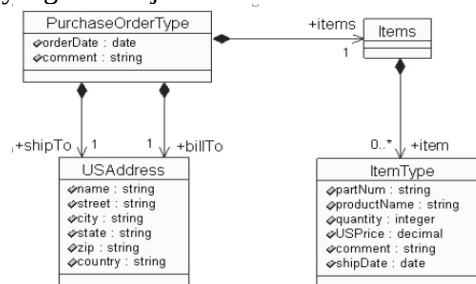
Atribut dan metoda dapat memiliki salah satu sifat berikut:

3. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
4. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Berikut ini merupakan hubungan antar *class*:

5. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
6. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas”).
7. Pewarisan, yaitu hubungan hirarki antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
8. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

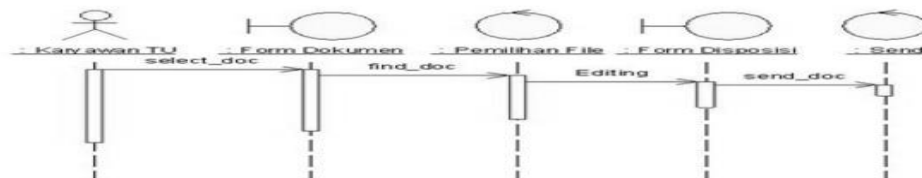


Gambar 2. Contoh Class Diagram (Ropianto, 2016)

i. Sequence Diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan use case diagram (Ropianto, 2016). *Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.



Gambar 3 Notasi Sequence Diagram (Ropianto, 2016)

j. User Interface

Merupakan komponen penting dari setiap sistem baru yang akan digunakan. *User Interface* merupakan bagian dari sistem informasi yang memerlukan interaksi langsung pada sistem *user* dalam membuat *input* dan *output*. Dalam membuat *user interface* terdapat aturan yang disebut *eight golden rules*. *Eight golden rules* bermanfaat dalam merancang *user interface* yang interaktif.

SIMPULAN

Berdasarkan latar belakang dan penelitian serta pembahasan yang dilakukan terhadap fungsi *dashboard* STT Ibnu Sina Batam dengan metode *Object Oriented Analysis and Design*, maka dapat ditarik kesimpulan sebagai berikut:

1. Untuk menerapkan metode *Object Oriented Analysis and Design* (OOAD) terlebih dahulu penulis melakukan wawancara dengan manajemen Ketua STT Ibnu Sina Batam dan Ketua Biro Administrasi Akademik dan Kemahasiswaan (BAAK) untuk menjelaskan *design interface system* yang akan penulis rancang.
2. Untuk merancang sistem *Dashboard* STT Ibnu Sina Batam penulis terlebih dahulu melakukan analisa terhadap sistem berjalan, merancang sistem baru, dan kemudian melakukan implementasi kedalam bahasa pemrograman.

SARAN

Berdasarkan hasil penelitian ini ada beberapa saran yang penulis usulkan untuk mengembangkan sistem agar menjadi lebih baik, diantaranya sebagai berikut:

1. Sistem ini dibangun diharapkan dapat menjadi sebagai bahan dalam peneliti lebih lanjut sehingga menghasilkan sistem baru yang lebih bermanfaat.

UCAPAN TERIMA KASIH

Ucapan terima kasih saya sampaikan kepada keluarga, pembimbing, penguji serta seluruh teman-teman seluruh manajemen Sekolah Tinggi Teknik Ibnu Sina Batam yang telah memberikan semangat, motivasi, dan dorongan dalam mengapai cita-cita.

DAFTAR PUSTAKA

- Abdurahman, M (2016). Sistem Informasi Jadwal Perkuliahan Berbasis *Web Mobile* pada Politeknik Sains Dan Teknologi Wiratama Maluku Utara.
- Braund D., Silvis J. Shapiro A. Versteegh J. (2001). *Object Oriented Analysis and Design* Team. Kennesaw State University CSIS 4650 Spring 2001.
- Bruce E. Wample, Ph.D (2003). *The Essence of Object Oriented Programming with Java and UML*. Addison-Wesley.
- Dharwiyanti, Sri., dan Wahono, Romi Satrio. (2003). “Pengantar Unified Modeling Language (UML)”, Ilmukomputer 2003.
- Eckerson, Wayne, 2006. *Perfomance Dashboard*. John Wiley and Sons. Canada.
- Few, Stephen. (2006). *Information Dashboard Design*. USA: O'Reilly Media. ISBN: 0-596-10016-7.
- Gordon.B Davis. (1998). Kerangka dasar sistem informasi Manajemn: Pengantar Seri Manajemen No: 90 A, PT. Pustaka Binaman Pressindo.
- Gunawan.(2016). I Polink: Indonesia *Police Link Model Knowledge Management* Untuk Membantu *Investigasi Forensika Digital*, Ilmu Teknik Universitas Islam Makasar, Universitas Islam Makasar.
- Hariyati, Eva. 2008. Metodologi Pembangunan Dashboard sebagai alat monitoring kinerja organisasi studi kasus intitut Teknologi Bandung. Bandung.
- Kusuma, T. A. H. (2012). *Dashboard Monitoring System* Penjualan Dan *Reward Mobile* Kios PT Telekomunikasi Seluler. *Semantik* 2012, 1-7.
- Prasetyotomo, Azis Senoaji. (2015). Pengembangan sistem informasi geospasial berbasis mobile pada perlindungan tenaga kerja Indonesia: studi kasus BNP2TKI.

Ropianto, M. (2016). Pemahaman Penggunaan *Unified Modelling Language*. Jurnal Teknik Ibnu Sina JT-IBSI, 1(01).

Ropianto, M. (2017). Pemanfaatan Sistem Dashboard Pada Data Akademik Di Sekolah Tinggi Teknik (STT) Ibnu Sina Batam. *Jurnal Teknik Ibnu Sina JT-IBSI*, 2(2).

Satzinger, J.W., Dkk. (2012). *Systems Analysis and Design in A Changing World*. (6th Edition). USA: Course Technology. ISBN-13: 978-1-111-53415-8 ISBN-10: 1-111-53415-2.

Scott W. Ambler (2003). The Element of UML style,” Cambridge University Press 2003.

Suhartini, (2017) Daerah, P. S. K. P. Analisis dan Pengembangan Aplikasi Pembayaran Studi Kasus Perusahaan Daerah Air Minum (PDAM) Tirta Prabujaya Kota Prabumulih.

Sungkar, I. I., Mustafid, I. W., & Widiyanto, I. (2011). *Performance Dashboard* pada Rumah Sakit Islam. *proc. Jurnal Sistem Informasi Bisnis*, 3, 123-128.