PETUNJUK PRAKTIKUM

ANALISIS DAN PERANCANGAN PERANGKAT LUNAK (1865331)

Versi 1.0
Februari 2018

Disusun oleh:
Herman Yuliansyah, S.T., M.Eng.
Sri Handayaningsih, S.T., M.T.
Murein Miksa Mardhia, S.T., M.T.
Arfiani Nurkhusna, S.T., M.Kom.

Laboratorium Basis Data
Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Ahmad Dahlan

Tahun Akademik 2017/2018

KATA PENGANTAR

Puji syukur kita panjatkan kehadirat Allah SWT, yang telah memberi hidayah-NYA sehingga Modul Praktikum Analisis Perancangan Perangkat Lunak ini dapat terwujud. Modul ini dimaksudkan untuk membantu mahasiswa dalam melaksanakan praktikum sehingga dapat memahami teori yang telah diberikan di kelas.

Modul praktikum ini terdiri dari 8 topik, yaitu: Proses Bisnis, Functional & Non Functional Requirement, Use Case Diagram, Activity Diagram, User Interface, Class Diagram I dan II, Sequence Diagram, Use Case Points. Topik-topik tersebut diuraikan dalam 9 lembar kerja. Topik-topik tersebut dilaksanakan dalam 10 kegiatan pertemuan.

Penyusun sangat berterimakasih bila pembaca berkenan memberi masukan, kritik, maupun saran untuk memperbaiki Modul Praktikum ini untuk meningkatkan kualitas proses belajar mengajar.

Semoga Modul Praktikum ini dapat bermanfaat dalam meningkatkan kualitas penyampaian ilmu pengetahuan yang dibutuhkan sebagai kompetensi mahasiswa Teknik Informatika di masa depan dan membantu mahasiswa dalam melaksanakan praktikum.

Yogyakarta Februari 2018,

Tim Penyusun APPL

PROSES BISNIS

Pertemuan ke : I

Alokasi Waktu: 1 Minggu & 1,5 jam

Tempat : Laboratorium

Kompetensi Dasar : 1. Mampu menggambarkan proses bisnis yang dijalankan sesuai studi kasus yang sedang diproyekkan.

2. Mampu menganalisis proses bisnis yang bermasalah

3. Mampu menggambarkan proses bisnis yang diharapkan untuk menyelesaikan permasalahan di kondisi saat ini.

Indikator : 1. Tergambar proses bisnis saat ini;

2. Hasil analisis dari permasalahan proses bisnis saat ini;

3. Tergambar proses bisnis yang diharapkan untuk menyelesaikan permasalahan.

A. Teori Pendukung

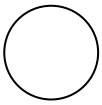
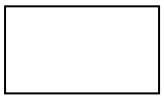
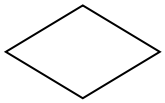
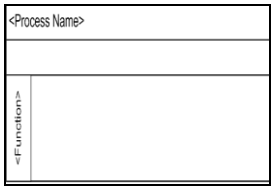
BPMN merupakan suatu teknik untuk memodelkan dan manajemen proses bisnis. Sebelumnya, organisasi telah mempergunakan berbagai teknik dan tools untuk memodelkan dan mengelola proses bisnis. Tetapi teknik-teknik tersebut tidak memiliki standarisasi yang lengkap dan siklus hidup yang lengkap untuk mengontrol dan memandu perancangan dan eksekusi proses bisnis. Ada beberapa teknik yang dipergunakan dalam memodelkan proses, yang dibedakan menjadi beberapa tingkatan pemodelan, yaitu :

1. Process maps, yaitu pemodelan proses yang ditampilkan melalui flowchart sederhana atau grafik sederhana dari aktifitas-aktifitas.
2. Process descriptions, yaitu pemodelan proses dengan penggunaan flowchart yang diperluas. Pemodelan proses pada tingkatan ini sudah dilengkapi dengan penambahan

informasi tetapi masih belum memadai untuk mendeskripsikan kinerja aktifitas yang sesungguhnya.

3. Process models, yaitu pemodelan proses bisnis melalui flowchart dilengkapi dengan informasi yang memadai sehingga proses yang dimodelkan dianalisis, disimulasi dan/atau dieksekusi.

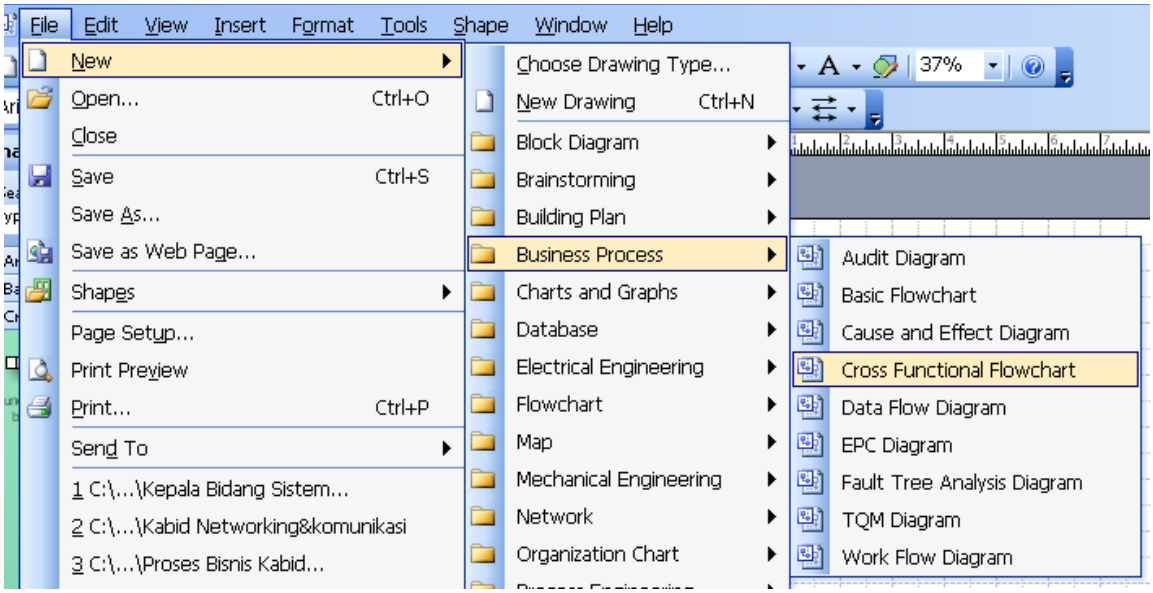
Elemen utama dalam *Business Process Diagram*

Elemen	Deskripsi	Notasi
Event	Sesuatu yang terjadi selama proses bisnis berlangsung. *Event* mempengaruhi aliran proses bisnis dan biasanya memiliki suatu akibat atau dampak. *Event* dibedakan menjadi tiga, yaitu *Start, Intermediate* dan *End*	
Activity	Pekerjaan yang dilaksanakan oleh perusahaan. Jenis *Activity* yang merupakan bagian dari model proses, yaitu *process, subprocess* dan *task*	
Gateway	Untuk mengontrol percabangan dan pertemuan dari *sequence flow*	
Sequence flow	Menunjukkan perintah yang akan dilaksanakan oleh suatu aktifitas dalam suatu proses	
Pool	suatu "*swimlane*" dan sebuah kontainer grafis untuk membagi serangkaian aktifitas dari *pool* lainnya	

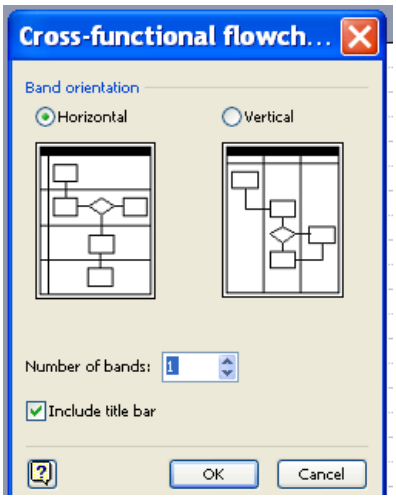
B. Langkah Praktikum

1. Berdasarkan studi kasus pengembangan aplikasi yang dipilih di kelas, buatkan proses bisnis kondisi saat ini.
2. Buatkan analisis permasalahan pada proses bisnis yang berjalan saat ini, berdasarkan hasil wawancara dengan customer.

3. Buat Proses bisnis yang diharapkan untuk memperbaiki proses bisnis saat ini, dengan memberikan solusi aplikasi yang Anda ambil.
4. Menggambarkan proses bisnis menggunakan alat bantu visio
   a. Buka visio
   b. Buka file/new/Business process/cross Functional Flowchart

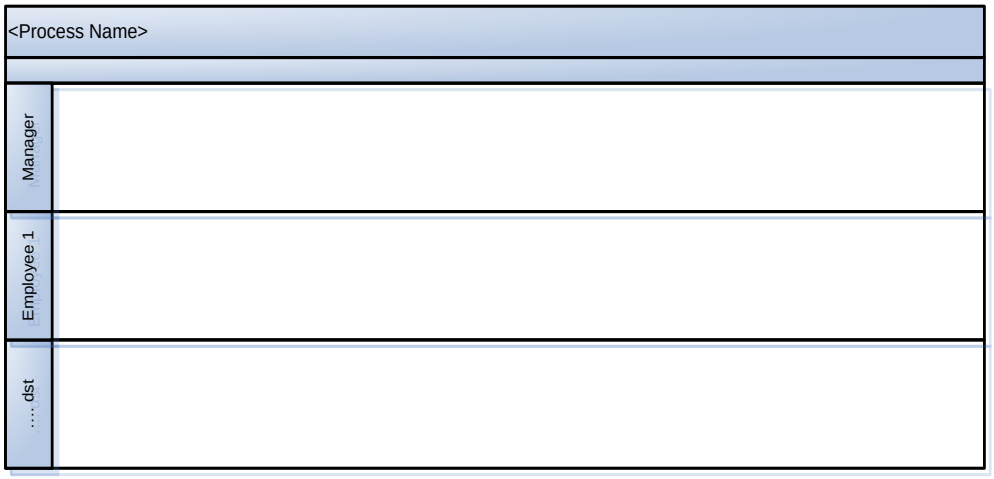


   c. Pilih yang horizontal, number of bands diisi dengan berapa jumlah line yang akan dibuat



d. Setelah itu akan muncul

Nama proses dituliskan proses bisnis yang akan digambarkan. Function menunjukkan pelaku bisnis yang bertanggungjawab menjalankan proses bisnis.



e. Buat gambar proses bisnisnya.

f. Lakukan verifikasi dengan asisten yang berlaku sebagai asisten.

C. Evaluasi

Buat laporan sesuai pada langkah-langkah praktikum

Nilai	Yogyakarta, ……. Paraf asisten (……)

jika diperlukan lembar jawaban lebih mahasiswa bisa menyisipkan kertas sendiri

D. Referensi

White, S. A., (2003). *Business Process Modelling Notation*,

http://www.bpmi.org 060610

KEBUTUHAN FUNGSIONAL DAN NON FUNGSIONAL

Pertemuan ke : **II**

Alokasi Waktu: 1 Minggu & 1,5 jam

Tempat : **Laboratorium**

Kompetensi Dasar : **1.** Mampu menganalisis kebutuhan fungsional dan kebutuhan non fungsional.

2. Mampu menyusun kebutuhan fungsional dan non fungsional yang dibutuhkan sistem

Indikator : **1.** Hasil analisis kebutuhan fungsional dan non fungsional;

2. Hasil pemetaan kebutuhan fungsional dan non fungsional yang dibutuhkan sistem.

A. Teori Pendukung

Requirements engineering adalah fase terdepan dari proses rekayasa perangkat lunak, di mana *software requirements* (kebutuhan) dari *user* (pengguna) dan *customer* (pelanggan) dikumpulkan, dipahami dan ditetapkan. Para pakar *software engineering* sepakat bahwa *requirements engineering* adalah suatu pekerjaan yang sangat penting. Fakta membuktikan bahwa kebanyakan kegagalan pengembangan *software* disebabkan karena adaya ketidakkonsistenan *(inconsistent)*, ketidaklengkapan *(incomplete)*, maupun ketidakbenaran *(incorrect)* dari *requirements specification* (spesifikasi kebutuhan).

Hasil dari fase *requirements engineering* terdokumentasi dalam SRS *(software requirements specification)* atau SKPL (spesifikasi kebutuhan perangkat lunak). SKPL berisi kesepakatan bersama tentang permasalahan yang ingin dipecahkan antara pengembang dan *customer*, dan merupakan titik start menuju proses berikutnya yaitu *software design.*

Tipe Requirements

Kebutuhan *(requirements)* perangkat lunak seringkali diklasifikasikan ke dalam dua kategori :

1. *Functional Requirements* (Kebutuhan Fungsional)

   Merupakan pernyataan tentang sekumpulan layanan/fitur yang harus tersedia dalam perangkat lunak.

2. *Non Functional Requirements* (Kebutuhan Non Fungsional)

   Terkait dengan kendala *(constraint)* dan kualitas dari perangkat lunak. Kualitas perangkat lunak adalah sifat atau karakteristik dari sistem yang stakeholders peduli dan karenanya akan mempengaruhi tingkat kepuasan terhadap sistem.

Tabel Parameter Kebutuhan Non Fungsional

SKPL-Id	Keterangan Parameter
SKPL-NF1	*Availability – Ketersediaan Aplikasi Untuk Dapat Diakses Oleh Pengguna.*
SKPL-NF2	Reliability – kehandalan aplikasi, termasuk aspek teknis seperti koneksi, kebutuhan hardware.
SKPL-NF3	Ergonomy – Desain Aplikasi harus disesuaikan dengan kenyamanan pengguna.
SKPL-NF4	Portability – Keberpindahan Aplikasi, sehingga dapat diakses oleh berbagai device.
SKPL-NF5	Memory – Kebutuhan Aplikasi akan media penyimpanan.
SKPL-NF6	Response time – Waktu Aplikasi untuk merespon request dari user.
SKPL-NF7	Safety – Keamanan data dari aplikasi, serta penggunaan aplikasi.
SKPL-NF8	Security – Keamanan aplikasi untuk melindungi data di dalamnya.
SKPL-NF9	Bahasa komunikasi – Media Bahasa yang digunakan oleh aplikasi.

B. Langkah Praktikum

1. Klasifikasikan daftar kebutuhan pelanggan ke dalam kategori kebutuhan fungsional dan kebutuhan non fungsional.
2. Berikan deskripsi dari masing-masing kebutuhan tersebut.

C. Evaluasi

Buat daftar kebutuhan fungsional dan kebutuhan non fungsional menggunakan tabel berikut!

Tabel Kebutuhan Fungsional

No	Kode	Deskripsi
1	SKPL-F1	
2		
3		
4		
dst		

Tabel Kebutuhan Non Fungsional

No	Kode	Parameter	Deskripsi
1	*SKPL-NF1*		
2			
3			
4			
5			
6			
7			
dst			

jika diperlukan lembar jawaban lebih mahasiswa bisa menyisipkan kertas sendiri

D. Referensi

Software Modeling and Design UML, Use Cases, Patterns, and Software Architectures - Gomaa – 2011

USE CASE

Pertemuan ke **: III**

Alokasi Waktu: 1 Minggu & 1,5 jam

Tempat **: Laboratorium**

Kompetensi Dasar **: 1.** Mampu memahami Unified Modelling Langguage (UML) sebagai suatu aktifitas dan modeling untuk perangkat lunak

2. Mampu memahami analisis dan notasi Use Case

3. Mampu membuat analisa Use Case yang diharapkan untuk menyelesaikan permasalahan di kondisi saat ini.

Indikator **: 1.** Hasil analisis tertuang dalam gambar use case dari studi kasus yang ditentukan saat ini;

2. Tergambar use case yang diharapkan untuk menyelesaikan permasalahan.

A. Teori Pendukung

Unified Modeling Language ™ (UML®) adalah bahasa pemodelan visual standar dimaksudkan untuk digunakan untuk

a. pemodelan bisnis dan sejenis proses,
b. analisis, desain, dan implementasi sistem berbasis software

UML merupakan bahasa umum untuk analis bisnis, arsitek dan pengembang perangkat lunak yang digunakan untuk menggambarkan, menentukan, desain, dan dokumen yang sudah ada atau proses bisnis baru, struktur dan perilaku artefak dari sistem perangkat lunak. UML adalah bahasa pemodelan standar, bukan proses pengembangan perangkat lunak. UML menjelaskan proses yang:

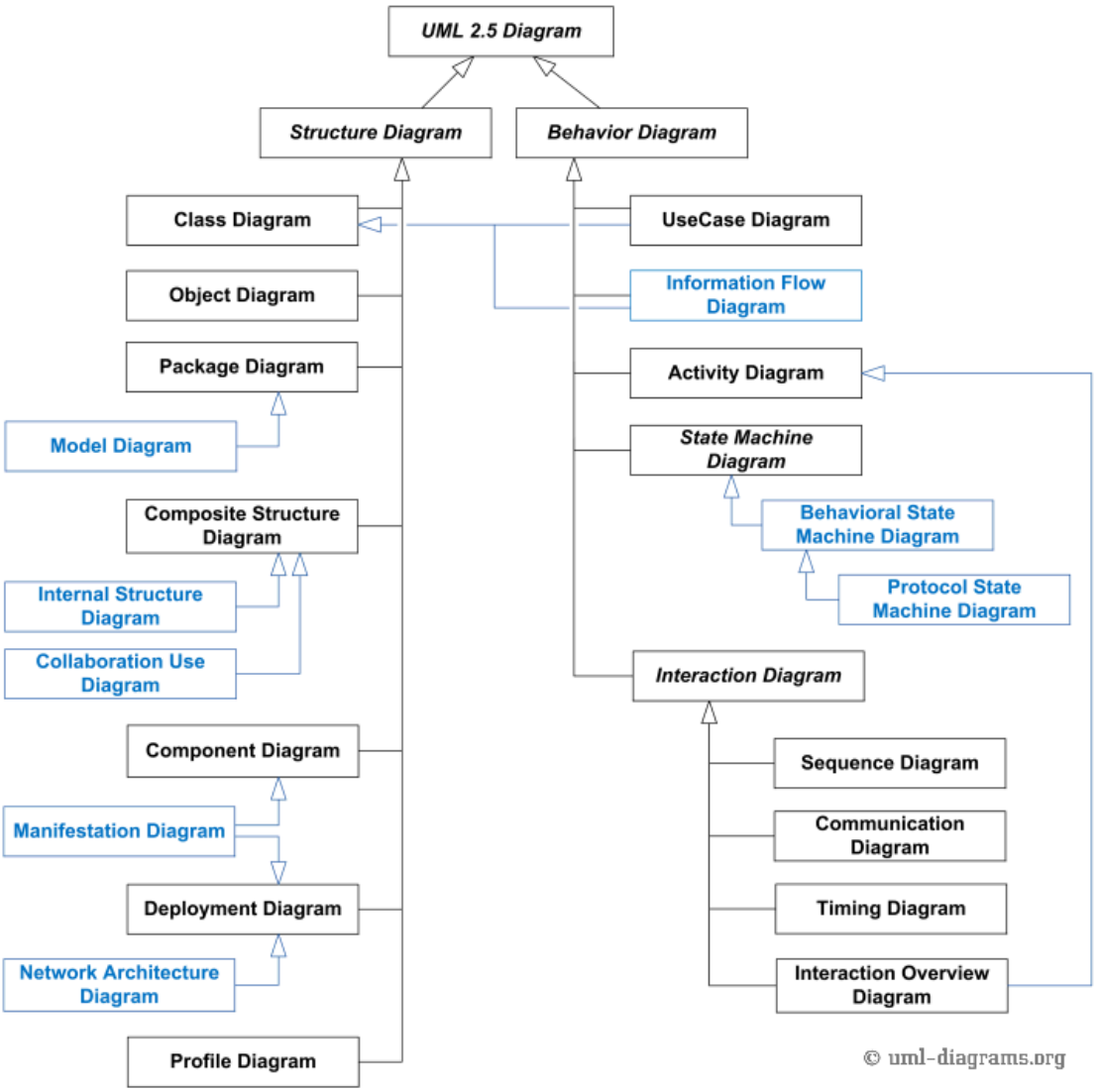
a. memberikan panduan untuk urutan kegiatan tim,
b. menentukan apa yang harus dikembangkan artefak,
c. mengarahkan tugas pengembang individu dan tim secara keseluruhan, dan
d. menawarkan kriteria untuk memantau dan mengukur produk dan kegiatan proyek.

UML spesifikasi mendefinisikan dua jenis utama dari diagram UML: **diagram struktur** dan **diagram perilaku**.

Diagram struktur menunjukkan struktur statis dari sistem dan bagian-bagiannya pada abstraksi yang berbeda dan tingkat pelaksanaan dan bagaimana mereka berhubungan satu sama lain. Unsur-unsur dalam diagram struktur mewakili konsep yang bermakna dari suatu sistem, dan mungkin termasuk abstrak, dunia nyata dan konsep implementasi.

Diagram perilaku menunjukkan perilaku dinamis dari objek dalam suatu sistem, yang dapat digambarkan sebagai serangkaian perubahan ke sistem dari waktu ke waktu.

Diagram UML 2.5 bisa dikategorikan hirarki seperti yang ditunjukkan di bawah ini. Catatan, item ditampilkan dalam warna biru bukan bagian resmi diagram taksonomi UML 2,5.



Gambar 3.1 Diagram UML 2.5

Keterangan beberapa diagram UML 2.5 sebagai berikut:

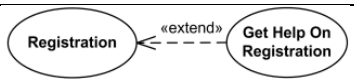
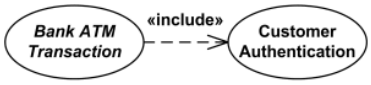
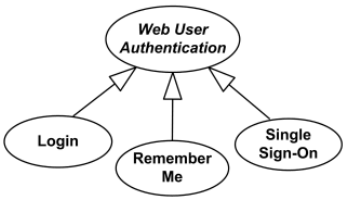
a. Class Diagram/Diagram Kelas: Menunjukkan struktur sistem yang dirancang, subsistem atau komponen sebagai kelas terkait dan interface, dengan fitur mereka, kendala dan hubungan - asosiasi, generalisasi, dependensi, dll
b. Object Diagram/Diagram Object: Instansiasi tingkat dari diagram kelas yang menunjukkan instansiasi spesifikasi kelas dan interface (objek), slot dengan spesifikasi nilai, dan link (instansiasi asosiasi).
c. Package Diagram/Diagram Paket: Menunjukkan paket dan hubungan antara paket.
d. Use Case: Menjelaskan serangkaian tindakan (Use Case) bahwa beberapa sistem atau sistem (subjek) harus atau dapat melakukan bekerjasama dengan satu atau lebih pengguna eksternal dari sistem (aktor) untuk memberikan beberapa hasil yang dapat diamati dan berharga untuk para aktor atau pemangku kepentingan lainnya dari sistem.
e. Activity Diagram/Diagram Aktifitas: Menunjukkan urutan dan kondisi untuk mengkoordinasikan perilaku-tingkat yang lebih rendah, daripada yang pengklasifikasi memiliki perilaku mereka. Ini biasa disebut aliran kontrol dan model aliran objek.
f. Sequence Diagram: merupakan jenis umum sebagian besar diagram interaksi yang berfokus pada pertukaran pesan

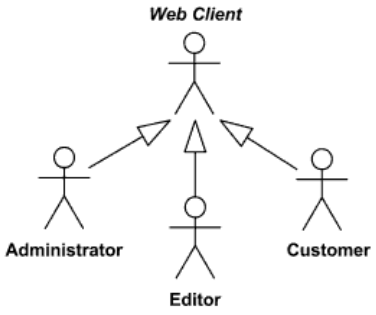
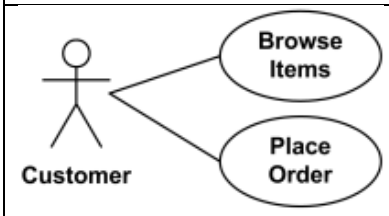
Use Case Diagram

Diagram use case biasanya disebut sebagai diagram perilaku yang digunakan untuk menggambarkan serangkaian tindakan (Use Case) bahwa beberapa sistem atau sistem (subjek) harus atau dapat melakukan bekerjasama dengan satu atau lebih pengguna eksternal dari sistem (aktor). Setiap penggunaan use case harus menyediakan beberapa hasil diamati dan berharga untuk para aktor atau pemangku kepentingan lain dari sistem.

Tabel 3.1 Notasi Use Case

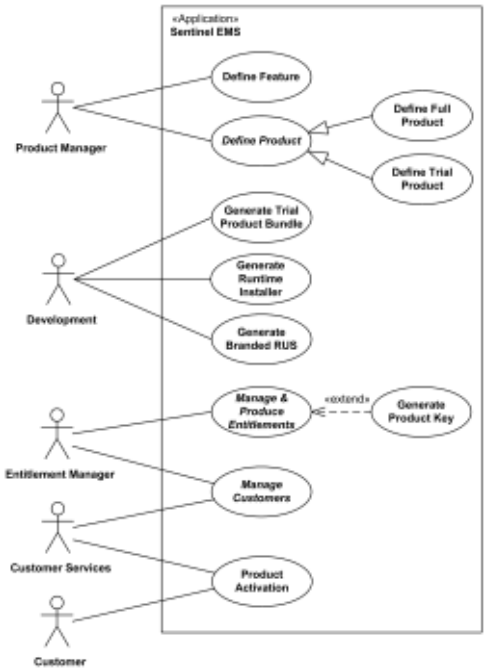
Notasi	Penjelasan
	Seorang aktor adalah **perilaku** yang menentukan peran yang dimainkan oleh entitas eksternal yang berinteraksi dengan subjek (misalnya, dengan bertukar sinyal dan data), pengguna manusia dari sistem yang dirancang, beberapa layanan sistem atau perangkat keras menggunakan lain dari

	subjek.
	Sebuah pelaku usaha merupakan peran yang dimainkan oleh beberapa **orang atau sistem eksternal** untuk bisnis dimodelkan dan berinteraksi dengan bisnis.
	Gunakan use case untuk menangkap persyaratan sistem, menggambarkan **fungsionalitas** yang disediakan oleh sistem-sistem, dan menentukan persyaratan sistem berpose di lingkungan mereka.
	Nama kasus penggunaan juga bisa ditempatkan di bawah elips.
	**Business use case** penggunaan bisnis use case untuk mendukung Pemodelan untuk mewakili fungsi bisnis, proses, atau kegiatan yang dilakukan dalam bisnis model. penggunaan bisnis kasus harus menghasilkan hasil nilai diamati untuk business actor.
	**Relasi Extend** adalah hubungan berarah yang menentukan bagaimana dan kapan perilaku didefinisikan dalam biasanya **tambahan (opsional)** penggunaan *extend use case* dapat dimasukkan ke dalam perilaku yang ditetapkan dalam kasus penggunaan yang berkepanjangan. Relasi antar use case ini merupakan relasi yang **optional, sebagai pelengkap**.
	**Relasi Include** merupakan hubungan berarah antara dua use case yang mana digunakan untuk menunjukkan bahwa tingkah laku dari use case include adalah ditambahkan dalam tingkah laku use case dasar. Relasi antar use case ini merupakan relasi yang **diperlukan**, tidak opsional.
	**Relasi Generalisasi** merupakan Generalisasi antara use case anak use case **mewarisi** sifat dan perilaku induk use case dan mungkin menimpa induk use case.

	
	**Relasi Asosiasi** menghubungkan antara aktor dan use case
	**Notasi Subject.** Subjek adalah bisnis, perangkat lunak sistem, subsistem, komponen, perangkat, dll. Hal ini sangat penting untuk menentukan jenis sistem itu, dan apa yang ruang lingkup atau batas.

Contoh sebuah use case:

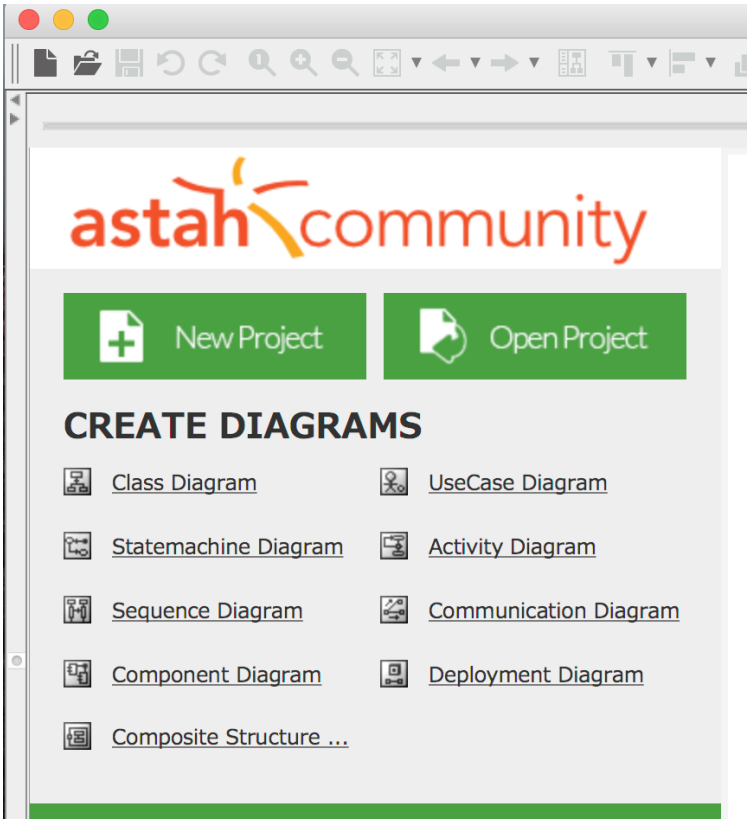
contoh diagram use case menunjukkan beberapa pandangan sederhana dari perangkat lunak lisensi use case yang didukung oleh Sentinel EMS Aplikasi.



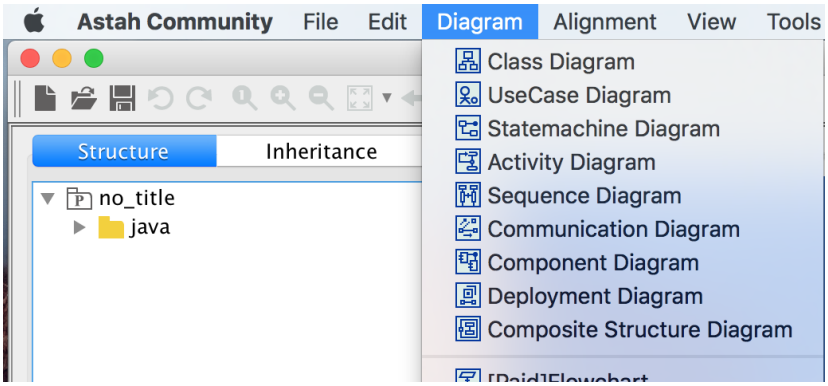
Gambar 3.2 Contoh Use Case

B. Langkah Praktikum

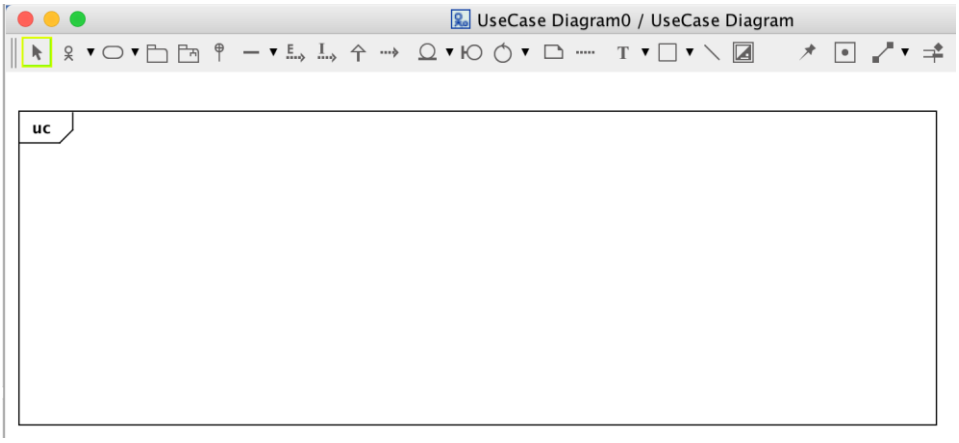
1. Berdasarkan studi kasus pengembangan aplikasi yang dipilih di kelas, buatkan use case kondisi saat ini.
2. Buatkan analisis use case, berdasarkan hasil wawancara dengan customer.
3. Menggambarkan use case menggunakan alat bantu astah community
   a. Buat Project Baru



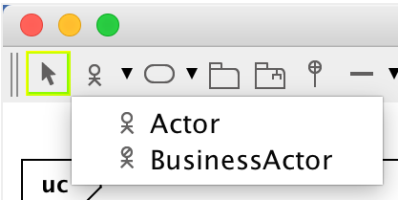
   b. Pilih Menu Diagram dan pilih UseCase Diagram

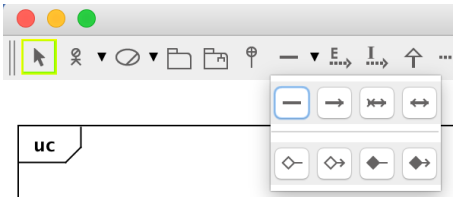


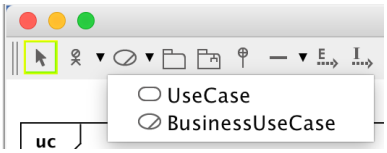
c. Notasi UseCase Diagram dapat ditemui pada toolbar jendela UseCase Diagram



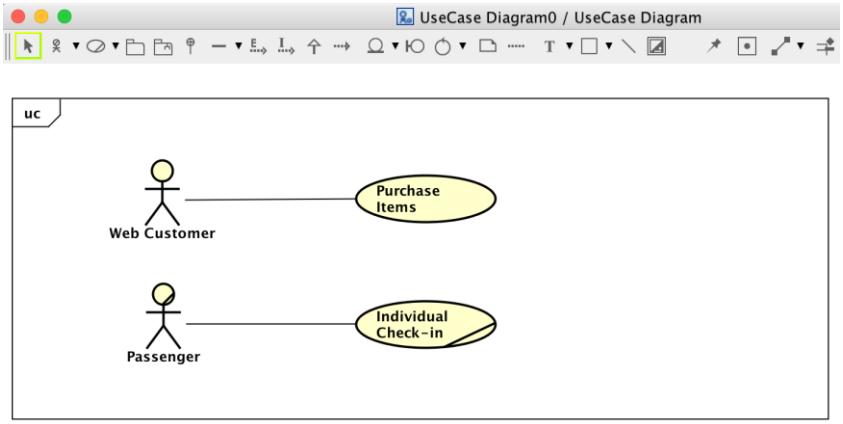
d. Untuk menggambar aktor, usecase atau relasi pilih ikon pada toolbar kemudian klik drag ke area subject







e. Buat gambar use casenya.



C. Evaluasi

Buat laporan diagram use case sesuai dengan tugas proyek anda.

Nilai	Yogyakarta, ……. Paraf asisten (……)

jika diperlukan lembar jawaban lebih mahasiswa bisa menyisipkan kertas sendiri

D. Referensi

Dennis, Alan, Barbara Haley Wixom, and David Tegarden. *Systems analysis and design: An object-oriented approach with UML*. John Wiley & Sons, 2015.

Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John Wiley & Sons, 2014.

http://www.uml-diagrams.org/

Activity Diagram

Pertemuan ke **: IV**

Alokasi Waktu: 1 Minggu & 1,5 jam

Tempat **: Laboratorium**

Kompetensi Dasar **: 1.** Mampu memahami Activity Diagram

2. Mampu membuat rancangan activity diagram

Indikator **: 1.** Hasil analisis tertuang dalam gambar Activity Diagram dari studi kasus yang ditentukan saat ini;

2. Tergambar Activity Diagram yang diharapkan untuk menyelesaikan permasalahan.

A. Teori Pendukung

Activity Diagram/Diagram aktivitas adalah UML behavior diagram /diagram perilaku UML yang menunjukkan flow of control/aliran kontrol atau arus objek/object flow dengan penekanan pada urutan dan kondisi arus. Tindakan yang dikoordinasikan oleh model aktivitas dapat dimulai karena tindakan lain selesai dijalankan, karena objek dan data tersedia, atau karena beberapa kejadian di luar arus terjadi.

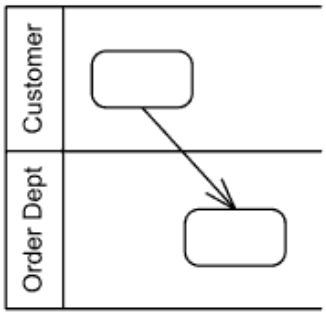
Activity

Activity/aktivitas adalah perilaku parameter yang ditunjukkan sebagai arus tindakan terkoordinasi. Aliran eksekusi dimodelkan sebagai node aktivitas yang dihubungkan oleh tepi aktivitas. Sebuah simpul bisa menjadi eksekusi dari perilaku bawahan, seperti perhitungan aritmatika, panggilan ke operasi, atau manipulasi isi objek. Activity nodes juga mencakup flow of control constructs, seperti synchronization, decision, dan concurrency control. Kegiatan dapat membentuk hierarki pemanggilan yang meminta aktivitas lain, pada akhirnya menyelesaikan tindakan individual. Dalam model berorientasi objek, aktivitas biasanya dipanggil secara tidak langsung sebagai metode yang terikat pada operasi yang dipanggil secara langsung.

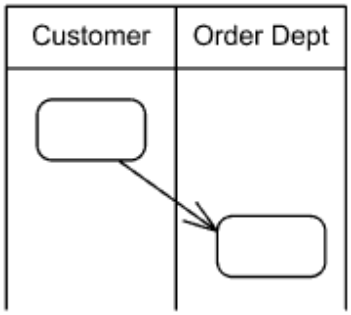
Activity Partition

Activity Partition/Partisi aktivitas adalah kelompok kegiatan untuk tindakan yang memiliki karakteristik umum. Partisi sering sesuai dengan **unit organisasi** atau **pelaku bisnis** dalam **model bisnis**.

Partisi aktivitas dapat ditunjukkan dengan menggunakan notasi **swimlane** - dengan dua, biasanya garis sejajar, baik horizontal atau vertikal, dan nama yang melabeli partisi dalam sebuah kotak di salah satu ujungnya. Setiap simpul aktivitas, mis. Tindakan dan tepi yang ditempatkan di antara garis-garis ini dianggap terkandung di dalam partisi (Gambar 4.1 dan Gambar 4.2).

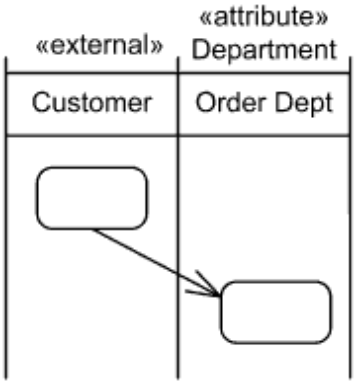


Gambar 4.1 Partisi kegiatan Customer and Order Dept sebagai swimlanes horizontal



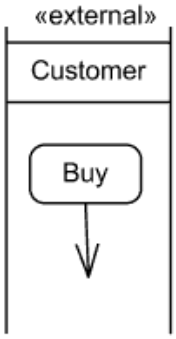
Gambar 4.2 Partisi kegiatan Customer and Order Dept sebagai swimlanes vertikal

Partisi dapat mewakili beberapa atribut dan subpartisinya - nilai spesifik atribut itu. Misalnya, partisi mungkin mewakili lokasi di mana perilaku dilakukan, dan sub-partisi akan mewakili nilai spesifik untuk atribut itu, seperti New York. Partisi hirarkis diwakili dengan menggunakan swimlanes untuk sub-partisi seperti diilustrasikan di Gambar 4.3.



Gambar 4.3 Partisi hierarkis dengan sub-partisi

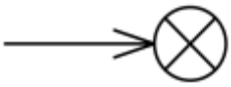
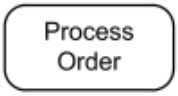
Partisi bisa mewakili entitas eksternal yang struktur partisinya tidak berlaku. Partisi eksternal adalah pengecualian yang disengaja terhadap peraturan untuk struktur partisi. Misalnya, dimensi mungkin memiliki partisi yang menunjukkan bagian pengklasifikasi terstruktur. Ini dapat memiliki partisi eksternal yang tidak mewakili salah satu bagian, namun merupakan klasifikasi yang benar-benar terpisah. Dalam pemodelan bisnis, partisi eksternal dapat digunakan untuk model entitas di luar bisnis. Bila aktivitas dianggap terjadi di luar domain model tertentu, partisi dapat diberi label dengan kata kunci «eksternal» seperti Gambar 4.5. Kapan pun sebuah aktivitas di swimlane ditandai «eksternal», ini menimpa perilakunya dan penunjukan dimensi.

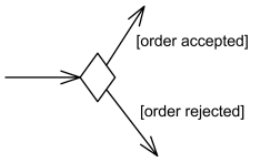
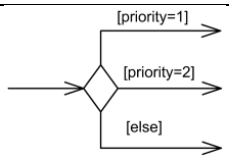
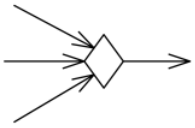
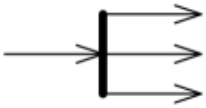
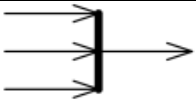


Gambar 4.5 Buy action terjadi pada partisi eksternal Customer

Untuk lebih lanjut, Tabel 4.1 akan menjelaskan tentang notasi-notasi dalam activity diagram.

Tabel 4.1 Notasi Use Case

Notasi	Penjelasan
 Activity initial node.	**Initial node/**Simpul awal adalah node kontrol dimana arus dimulai saat aktivitas dipanggil. Aktivitas mungkin memiliki lebih dari satu simpul awal. Simpul awal ditampilkan sebagai lingkaran padat kecil.
 Flow final node.	**Flow final node/**Arus simpul akhir adalah node akhir kontrol yang mengakhiri aliran. Notasi untuk node arus akhir adalah lingkaran kecil dengan X di dalamnya.
 Activity final node.	**Activity final/**Aktivitas simpul terakhir adalah node akhir kontrol yang menghentikan semua arus dalam suatu kegiatan. Kegiatan akhir baru di UML 2.0. Aktivitas simpul terakhir ditampilkan sebagai lingkaran padat dengan lingkaran berongga di dalamnya.
 Action	Action/tindakan digambarkan sebagai persegi panjang dengan ujung melingkar. Nama tindakan atau deskripsi lainnya mungkin muncul dalam simbol.

Notasi	Penjelasan
 Decision	**Decision/**Keputusan node dengan dua tepi keluar. Decision node adalah node kontrol yang menerima token pada satu atau dua sisi yang masuk dan memilih satu tepi keluar dari satu atau lebih arus keluar.
 Decision	Keputusan node dengan tiga tepi keluar dan [else] Untuk poin keputusan, panah "ELSE" yang telah ditentukan dapat didefinisikan paling banyak satu tepi keluar.
 Merge	Merge/Gabungan simpul dengan tiga sisi yang masuk dan tepi keluar tunggal. Merge node adalah node kontrol yang menyatukan beberapa arus masuk untuk menerima arus keluar tunggal. Tidak ada penggabungan token. Gabung seharusnya tidak digunakan untuk menyinkronkan arus bersamaan.
 Fork	Simpul fork/garpu dengan tepi aktivitas tunggal yang memasukinya, dan tiga sisi meninggalkannya. Fork node adalah node kontrol yang memiliki satu edge yang masuk dan beberapa tepi keluar dan digunakan untuk membagi arus masuk menjadi beberapa aliran bersamaan. Notasi untuk simpul garpu adalah segmen garis dengan tepi aktivitas tunggal yang memasukinya, dan dua atau lebih ujungnya meninggalkannya.
 Join	Join node/Node gabungan dengan tiga sisi aktivitas yang masuk, dan satu sisi meninggalkannya. Gabung simpul adalah node kontrol yang memiliki beberapa tepi masuk dan satu tepi keluar dan digunakan untuk menyinkronkan arus masuk bersamaan. Notasi untuk node join adalah segmen garis dengan beberapa tepi aktivitas yang memasukinya, dan hanya satu sisi yang meninggalkannya.

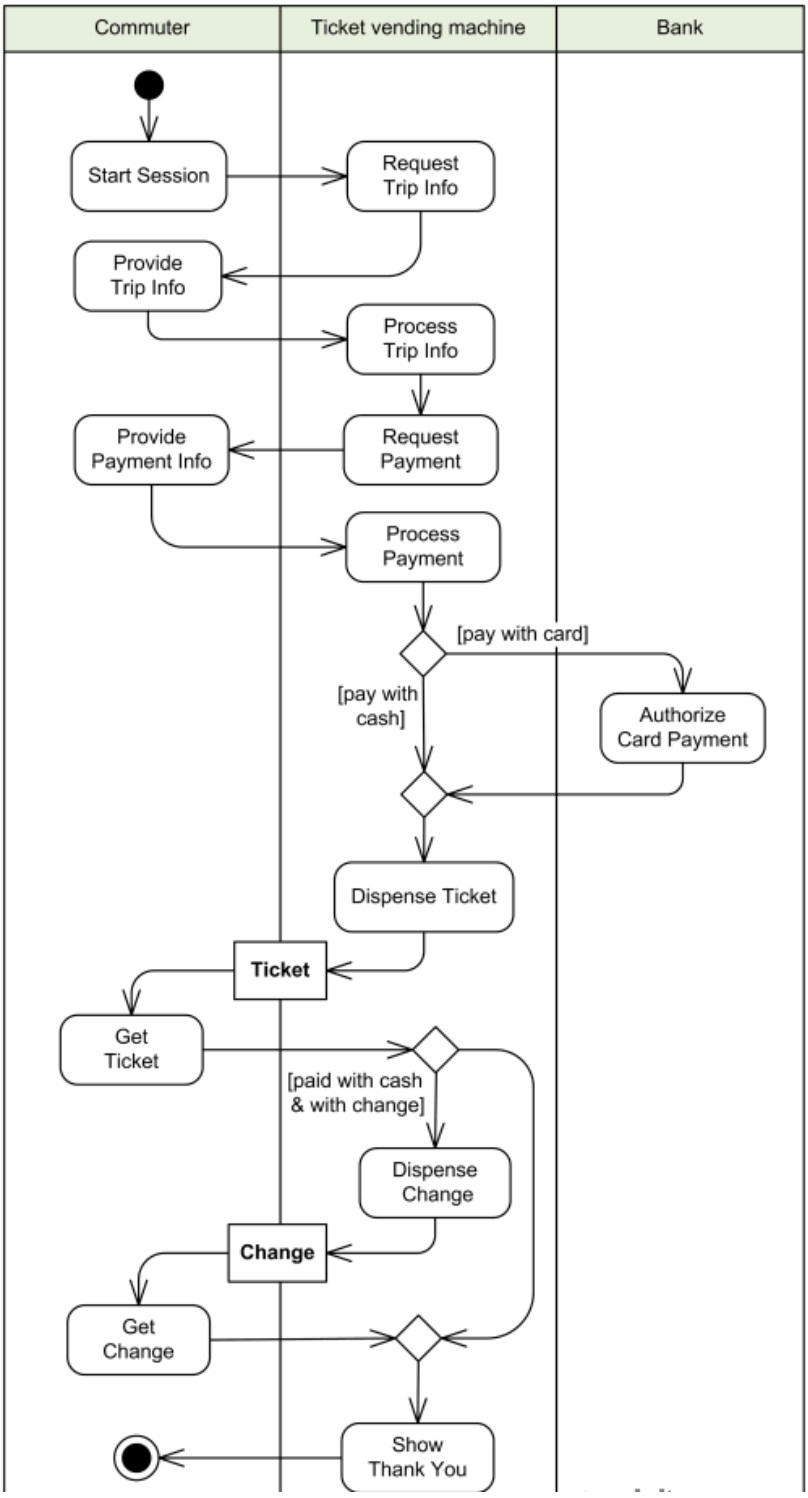
Contoh Ticket Vending Machine

Gambar 4.6 adalah contoh diagram aktivitas UML yang menggambarkan perilaku kasus penggunaan Tiket Pembelian.

Kegiatan dimulai oleh pelaku komuter yang perlu membeli tiket. Mesin penjual tiket akan meminta informasi perjalanan dari Commuter. Informasi ini akan mencakup nomor dan jenis tiket, mis. Apakah itu tiket bulanan, tiket satu arah atau bulat, nomor rute, nomor tujuan atau zona, dll.

Berdasarkan info yang disediakan info tiket mesin penjual akan menghitung pembayaran jatuh tempo dan meminta opsi pembayaran. Pilihan tersebut meliputi pembayaran secara tunai, atau dengan kartu kredit atau debit. Jika pembayaran dengan kartu dipilih oleh Commuter, aktor lain, Bank akan berpartisipasi dalam kegiatan tersebut dengan memberi otorisasi pembayaran.

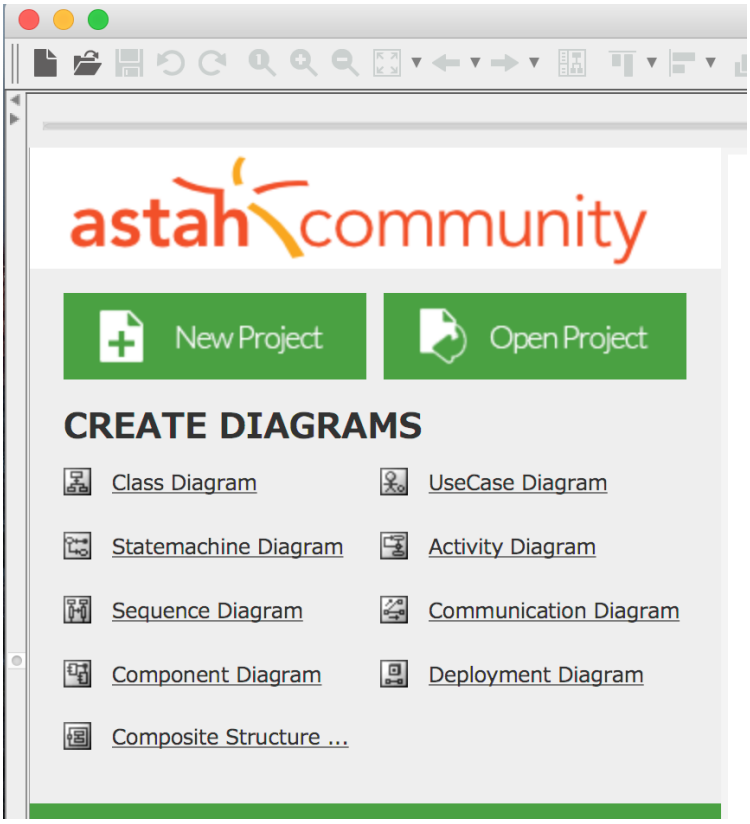
Setelah pembayaran selesai, tiket dibagikan kepada Commuter. Pembayaran tunai bisa mengakibatkan beberapa perubahan karena, sehingga perubahan tersebut dibagikan kepada Commuter dalam kasus ini. Mesin penjual tiket akan menampilkan beberapa layar "Terima Kasih" di akhir aktivitas.



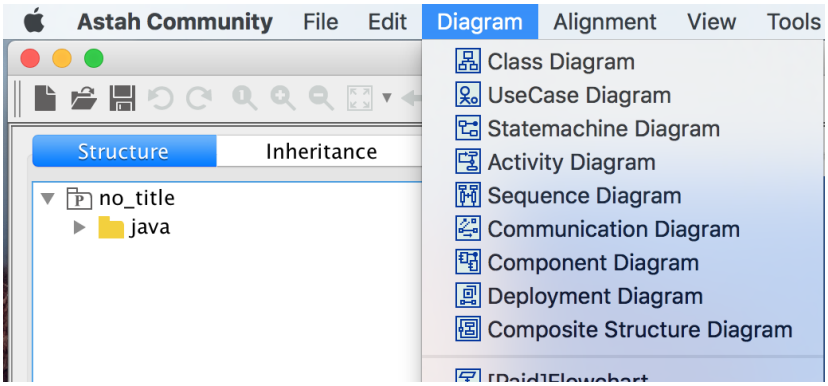
Gambar 4.6 Contoh perilaku penggunaan Tiket Pembelian menggunakan diagram aktivitas UML.

B. Langkah Praktikum

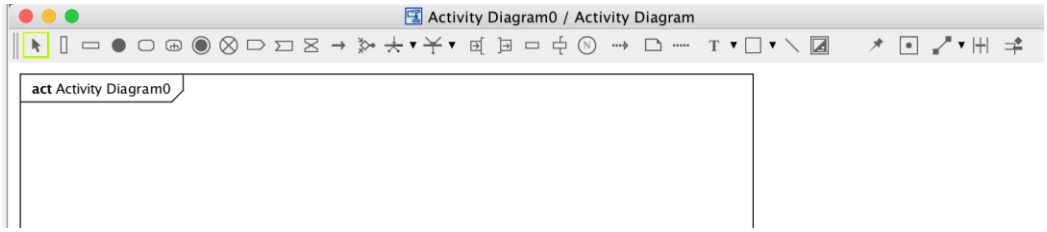
1. Berdasarkan studi kasus pengembangan aplikasi yang dipilih di kelas, buatkan activity diagram kondisi saat ini.
2. Buatkan analisis activity diagram, berdasarkan hasil wawancara dengan customer.
3. Menggambarkan activity diagram menggunakan alat bantu astah community
   a. Buat Project Baru



   b. Pilih Menu Diagram dan pilih Activity Diagram



c. Notasi activity diagram dapat ditemui pada toolbar jendela activity diagram



d. Buat gambar activity diagramnya.

C. Evaluasi

Buat laporan activity diagram sesuai dengan tugas proyek anda.

Nilai	Yogyakarta, ……. Paraf asisten (……)

jika diperlukan lembar jawaban lebih mahasiswa bisa menyisipkan kertas sendiri

D. Referensi

Dennis, Alan, Barbara Haley Wixom, and David Tegarden. *Systems analysis and design: An object-oriented approach with UML*. John Wiley & Sons, 2015.

Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John Wiley & Sons, 2014.

http://www.uml-diagrams.org/

PERANCANGAN USER *INTERFACE*

Pertemuan ke **: V**

Alokasi Waktu **: 1 Minggu & 1,5 jam**

Tempat **: Laboratorium**

Kompetensi Dasar **:** 1. Memahami konsep dasar *User Interface* (UI)

2. Merancang *user interface* dari rancangan diagram aktifitas

3. Mengimplementasikan diagram aktifitas ke dalam alur *interface*

Indikator **: 1. Fungsionalitas yang akan membutuhkan interaksi lewat UI telah diidentifikasi**

2. Desain UI yang diturunkan dari diagram aktifitas telah dirancang dan diimplementasikan

A. Teori Pendukung

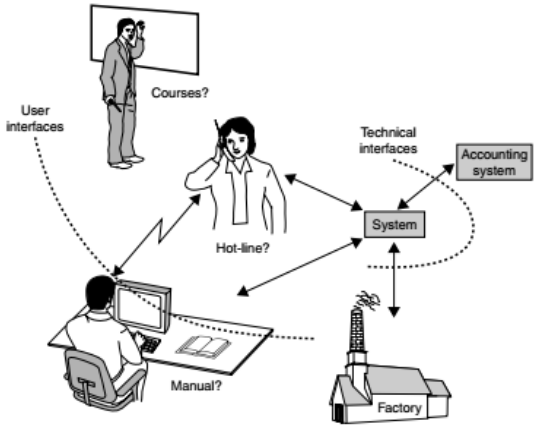
User *Interface* (UI) adalah bagian dari sistem yang dapat dilihat, didengar ataupun dirasakan oleh pengguna. Sedangkan bagian lainnya tersembunyi, misalnya adalah dimana basis data disimpan. Meskipun pengguna tidak melihat bagian yang tersembunyi, mereka membayangkan apa yang terjadi di 'belakang layar'. Apa yang mereka bayangkan mengenai apa yang terjadi ini seringkali mempunyai arti penting ketika pengguna menemui situasi dimana sistem tidak bekerja dengan seharusnya[2].

Saat kita menggunakan komputer, kita menjalankannya dengan memberikan perintah, umumnya melalui *mouse* dan *keyboard*. Komputer akan merespon perintah kita, biasanya dengan memperlihatkan sesuatu di layer atau membuat suatu suara. Terkadang situasi tersebut terbalik, komputer yang memberikan perintah dan kita yang memberikan respon.

Sebuah bioskop adalah sebuah contoh – penonton tidak berinteraksi langsung dengan filmnya. Contoh lain adalah *smoke detector* – alat ini bekerja dengan terus menerus mendeteksi

adanya asap, tetapi pengguna seolah hanya melihat *smoke detector* bekerja saat asapnya benar-benar muncul.

Interaksi dengan komputer ini terjadi melalui adanya User *Interface* (UI). Dalam sebuah komputer PC pada umumnya, aplikasi UI yang ada terdiri dari layar, *keyboard*, *mouse* dan *speaker* (Gambar 1).



Gambar 1. Sistem *Interface* [2]

Pada sistem yang lebih canggih, *interface* yang ditampilkan mungkin meliputi suara melalui mikrofon; tombol khusus, lampu dan tampilan layar; sarung tangan elektrik yang bisa mendeteksi gerakan jari tangan; dan sensor mata yang dapat mendeteksi arah gerak bola mata terhadap layar.

Tujuan dari aktifitas perancangan *interface* ini antara lain [1]:

1. Untuk memperoleh desain dan tampilan *interface* dengan dasar fundamental yang kuat, yang langsung diturunkan sejak dari fase pertama analisis perancangan sistem perangkat lunak, khususnya dari fase diagram aktifitas.
2. Untuk memberikan ruang bagi pengembang untuk mengetahui bagaimana pandangan pengguna terhadap desain *interface* yang diusulkan, dan untuk memperbaiki error (jika ada).
3. Untuk menciptakan dukungan kerjasama / kolaborasi yang baik antara perancang *User Interface* dan programmer (*software developer*).

Best Practices Membangun *Interface*

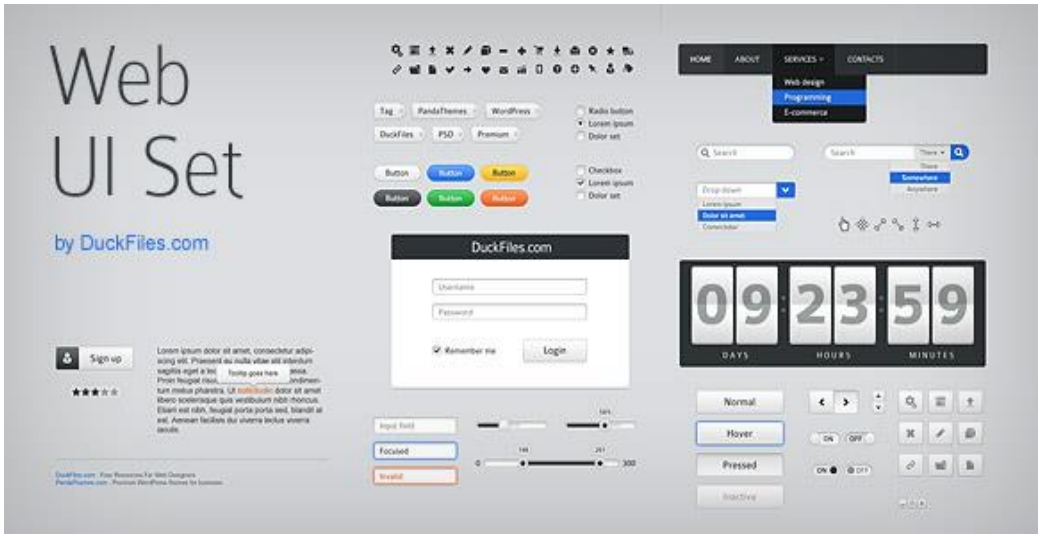
Ketika membangun sebuah sistem telah sampai pada tahapan perancangan *interface*, beberapa hal yang harus diketahui antara lain [3][5]:

- **Buat *interface* yang sederhana.**

  Tidak ada *interface* yang paling baik untuk semua pengguna, yang penting hindari elemen yang tidak perlu dan menggunakan Bahasa yang jelas ketika menggunakan label dan menuliskan pesan (message).

- **Ciptakan elemen UI yang konsisten dan umum diketahui.**

  Pengguna akan merasa lebih nyaman dan dapat berinteraksi dengan sistem secara lebih cepat. Selain itu, penting juga untuk menciptakan pola dalam kalimat Bahasa yang digunakan, dalam layout dan keseluruhan perancangan untuk alasan efisiensi. Apabila pengguna sudah memahami bagaimana caranya melakukan sesuatu melalui *interface* tersebut, mereka akan jadi terbiasa dalam mengoperasikan sistem / PL tersebut. Gambar 1 menyajikan contoh elemen-elemen yang umum diketahui :



Gambar 2. Contoh Elemen-Elemen UI [6]

- **Buatlah layout halaman yang mempunyai tujuan**

  Pertimbangkan hubungan antara hal-hal yang ingin ditampilkan di sebuah halaman dan bagaimana struktur halaman berdasarkan bagian mana yang penting. Penempatan

informasi tersebut harus dilakukan dengan hati-hati dengan tujuan untuk menarik perhatian pengguna terhadap bagian informasi yang paling penting.

- **Gunakan warna dan tekstur yang taktis.**

  Pengembang *interface* dapat mengarahkan atau mengembalikan perhatian terhadap suatu hal melalui warna, pencahayaan, dan tekstur, bergantung kepada tujuan yang ingin dicapai.

- **Gunakan tipografi untuk menciptakan kemampuan pemahaman pengguna.**

  Tipografi adalah seni mengatur atau menggunakan huruf, kata atau paragraf pada ruang yang tersedia. Berhati-hatilah dalam menggunakan jenis font, ukuran font dan pengaturan tata letak teks untuk meningkatkan kemampuan memahami pengguna.

- **Pastikan bahwa sistem / PL selalu mengkomunikasikan apa yang terjadi.**

  Selalu berikan informasi kepada pengguna mengenai lokasi, aksi, perubahan (bila ada) atau mengenai adanya error. Penggunaan berbagai macam elemen UI untuk mengkomunikasikan sebuah status aktifitas, dan mungkin saja langkah selanjutnya sebagai respon atas status yang terjadi.

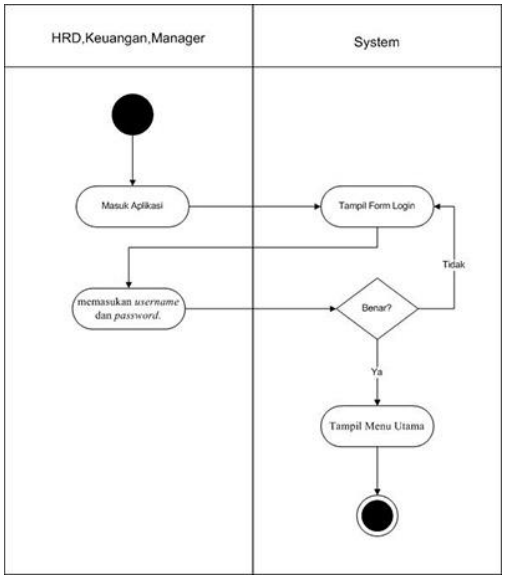
- **Pikirkan mengenai pengaturan awal (default).**

  Dengan mengantisipasi tujuan yang ingin dicapai pengguna dari sistem / PL, pengembang *interface* dapat membuat sebuah pengaturan awal (default) yang bisa mengurangi beban pengguna. Hal ini menjadi penting ketika ada rancangan form, dan ada isian default yang akan otomatis tertulis saat pengguna tidak mengisi form tersebut.

Memetakan Aktifitas ke Dalam Rancangan UI

Di praktikum ini, kita akan belajar merancang *interface* yang diturunkan dari diagram aktifitas. Langkah-langkah yang dapat dilakukan antara lain:

1. Siapkan diagram aktifitas dari sistem perangkat lunak yang sudah dirancang sebelumnya. Contoh di bawah ini adalah diagram aktifitas untuk aktifitas login, seperti yang ditunjukkan pada Gambar 2.



Gambar 3. Contoh Diagram Aktifitas Login [4]

2. Dari setiap diagram aktifitas yang dibuat, buat dokumen aktifitas yang mendata tentang fungsionalitas apa saja yang dimiliki. Dari masing-masing fungsionalitas tersebut, selanjutnya diturunkan ke dalam elemen-elemen UI. Dokumen aktifitas dibuat berdasarkan siapa saja yang menjadi subjek pelaku (aktor). Contoh dokumen aktifitas bisa dilihat seperti pada Gambar 3.

Judul Dokumen : .....
Jumlah aktifitas : M
Jumlah aktifitas yang perlu UI : N
1. Nama aktifitas : .....
Fungsionalitas : .....
Elemen UI : .....
Catatan : .....
2. Nama aktifitas : .....
Fungsionalitas : .....
Elemen UI : .....
Catatan : .....
.....
.....
.....
N. Nama aktifitas : .....
Fungsionalitas : .....
Elemen UI : .....
Catatan : .....

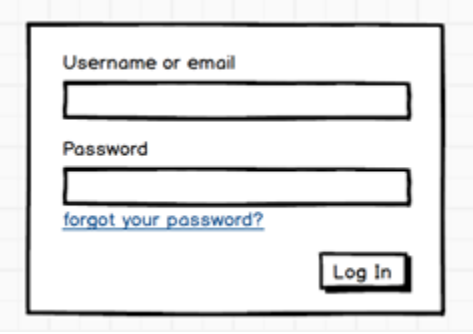
Gambar 4. Dokumen Aktifitas [1]

3. Jumlah aktifitas diisi dengan berapa jumlah aktifitas yang ada di setiap actor, misal berjumlah M. sedangkan jumlah aktifitas yang perlu UI diisi dengan berapa jumlah M yang akan berinteraksi dengan pengguna lewat *interface*, misal berjumlah N. M dan N tidak harus selalu sama. Contoh :

Judul Dokumen : User (HRD, Keu, Man)
Jumlah aktifitas : 2
Jumlah aktifitas yang perlu Interaksi UI : 1
1. **Nama aktifitas** : Memasukkan UsName & Pwd
Fungsionalitas : (1) user masuk sistem dengan mengetikkan username & password
(2) user submit data dengan tombol Login
(3) user dapat meminta bantuan bila lupa UsName atau Password
Elemen UI : (1) form textField UsName & Password, (2) Button Submit, (3) Link "Forgot
Catatan : -

Gambar 5. Contoh Dokumen Aktifitas untuk Login

4. Fungsionalitas diisi dengan kemampuan apa saja yang harus dilakukan oleh aktifitas terkait. Misal, fungsionalitas registrasi, menambah atau menghapus data barang. Elemen UI disesuaikan dengan fungsionalitas yang tadi sudah didata.
5. Catatan digunakan untuk memperjelas interaksi yang terjadi antar elemen yang harus dapat dilakukan lewat *interface*.
6. Setiap elemen UI yang merepresentasikan setiap fungsionalitas kemudian digambarkan dalam rancangan *mockup* sederhana.



Gambar 6. Mockup UI untuk Fungsionalitas Login

B. Langkah Praktikum

1. Berdasarkan studi kasus pengembangan aplikasi yang dipilih di kelas, rancanglah antarmuka yang diperlukan sesuai dengan diagram aktifitas yang dibuat sebelumnya.
2. Siapkan diagram aktifitas yang telah dibuat dari masing-masing studi kasus yang dipilih.
3. Buat dokumen aktifitas untuk setiap aktifitas yang memiliki fungsionalitas yang harus ditampilkan melalui *interface*.
4. Dari tabel dokumen yang dibuat di poin (3), buat rancangan / desain *interface (mockup)* nya.
5. Lakukan verifikasi fungsionalitas dan rancangan *interface* yang diajukan dengan asisten praktikum.

C. Evaluasi

Buat laporan perancangan UI untuk studi kasus masing-masing sesuai pada langkah-langkah praktikum !

Nilai	Yogyakarta, ……. Paraf asisten (……)

jika diperlukan lembar jawaban lebih mahasiswa bisa menyisipkan kertas sendiri

D. Referensi

[1] E. K. Elberkawi and M. M. Elammari, "Producing Graphical User *Interface* from Activity Diagrams," *Int. Sci. Index, Comput. Inf. Eng.*, vol. 9, no. No:3, pp. 667–672, 2015.

[2] Soren Lauesen. 2005. *User Interface Design: A Software Engineering Perspective*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[3] Jesse James Garrett's *The Elements of User Experience: User-Centered Design for the Web and Beyond (2nd Edition)*

[4] "Apa Itu Activity Diagram" https://www.dumetschool.com/blog/Apa-Itu-Activity-Diagram diakses 13 April 2017

[5] "User *Interface* Elements" https://www.usability.gov/how-to-and-tools/methods/user-*interface*-elements.html diakses 13 April 2017

[6] "User *Interface* Elements Free" http://clipart.me/24018/70-user-*interface*-elements-free-psd diakses pada 13 April 2017

SEQUENCE DIAGRAM

Pertemuan ke **: VII**

Alokasi Waktu: 1 Minggu & 1,5 jam

Tempat **: Laboratorium**

Kompetensi Dasar **: 1.** Mampu memahami analisis dan notasi sequence diagram

2. Mampu membuat rancangan sequence diagram

Indikator **: 1.** Hasil rancangan tertuang dalam gambar sequence diagram dari studi kasus yang ditentukan saat ini;

2. Tergambar sequence diagram yang diharapkan untuk menyelesaikan permasalahan.

A. Teori Pendukung

Sequence diagram adalah bentuk diagram interaksi yang menunjukkan objek sebagai jalur kehidupan yang mengalir menuruni halaman, dengan interaksi mereka dari waktu ke waktu ditunjukkan sebagai pesan yang ditarik sebagai panah dari sumber lifeline ke garis hidup target. Sequence diagram bagus untuk menunjukkan objek mana yang berkomunikasi dengan objek lain; Dan pesan apa yang memicu komunikasi tersebut. Sequence diagram tidak dimaksudkan untuk menunjukkan logika prosedural yang kompleks.

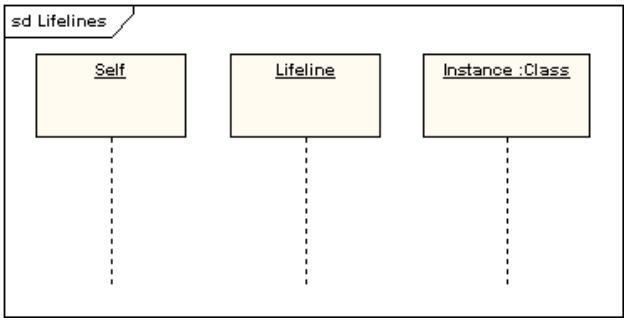
Sequence diagram adalah diagram interaksi yang menunjukkan bagaimana objek beroperasi satu sama lain dan dalam urutan apa. Ini adalah konstruksi dari sebuah bagan urutan pesan. Sequence diagram menunjukkan interaksi objek yang diatur dalam urutan waktu. Ini menggambarkan objek dan kelas yang terlibat dalam skenario dan urutan pesan dipertukarkan antara objek yang dibutuhkan untuk menjalankan fungsi skenario Sequence diagram biasanya dikaitkan dengan realisasi kasus penggunaan dalam Tampilan Logis dari sistem yang sedang dikembangkan. Sequence diagram kadang disebut diagram acara atau skenario acara. Diagram urutan menunjukkan, sebagai garis vertikal paralel (lifelines), proses atau objek yang berbeda yang hidup bersamaan, dan, sebagai panah horisontal, pesan dipertukarkan di antara keduanya, sesuai urutan kemunculannya. Hal ini memungkinkan spesifikasi skenario runtime sederhana secara grafis.

Sequence diagram adalah jenis diagram interaksi yang paling umum, yang berfokus pada pertukaran pesan antara sejumlah lifelines. Sequence diagram menggambarkan interaksi dengan memusatkan perhatian pada urutan pesan yang dipertukarkan, bersamaan dengan spesifikasi kejadian yang sesuai pada jalur kehidupan.

Lifeline

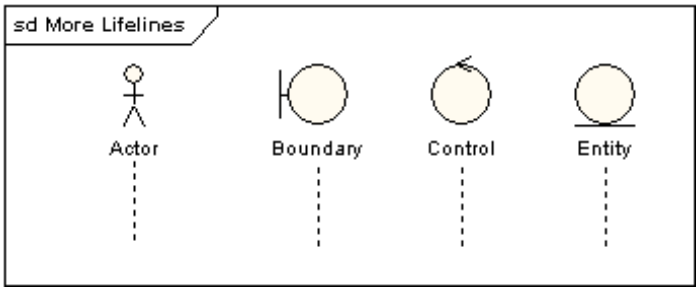
Lifeline adalah elemen bernama yang mewakili peserta individual dalam interaksi. Sementara bagian dan fitur struktural mungkin memiliki multiplisitas lebih besar dari 1, garis hidup hanya mewakili satu entitas yang berinteraksi.

Lifeline mewakili peserta individual dalam sequence diagram. Lifeline biasanya memiliki persegi panjang yang berisi nama objeknya. Jika namanya adalah "Self", itu menunjukkan lifeline mewakili penggolong yang memiliki sequence diagram.



Gambar 7.1 Lifeline

Terkadang sequence diagram akan memiliki lifeline dengan simbol elemen aktor di atasnya. Ini biasanya akan terjadi jika sequence diagram dimiliki oleh use case. Batasan, kontrol dan elemen entitas dari diagram dapat juga memiliki lifeline seperti pada Gambar 7.1 yang menggambarkan lifeline dalam Boundary (Batasan), control (kontrol), entity (entitas).

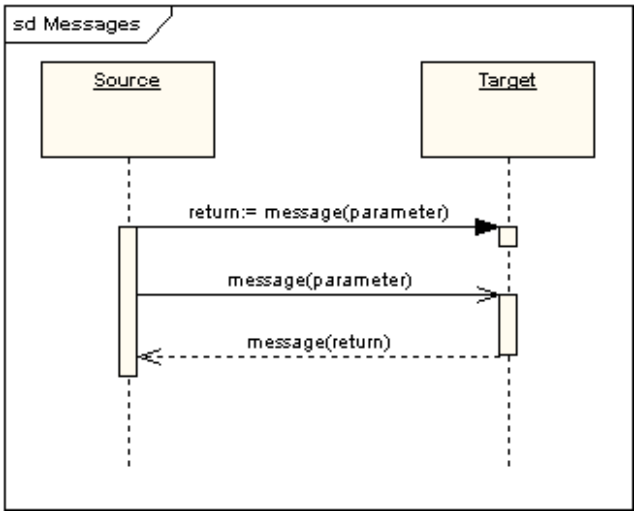


Gambar 7.2 Lifeline

Messages/pesan

Pesan ditampilkan sebagai tanda panah. Pesan bisa lengkap, hilang atau ditemukan; Sinkron atau asinkron; Panggilan atau sinyal Pada diagram berikut, pesan pertama adalah pesan sinkron

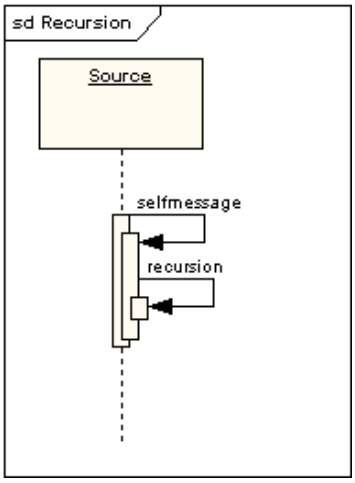
(dilambangkan dengan panah padat) lengkap dengan pesan balik implisit; Pesan kedua adalah asinkron (dilambangkan dengan panah baris), dan yang ketiga adalah pesan kembali asinkron (dilambangkan dengan garis putus-putus) seperti gambar 7.3.



Gambar 7.3 Message

Self Message

Selft Message dapat mewakili panggilan rekursif suatu operasi, atau satu metode memanggil metode lain yang termasuk dalam objek yang sama. Hal ini ditunjukkan pada Gambar 7.4 sebagai menciptakan fokus pengendalian yang terpusat dalam kejadian eksekusi the lifeline.



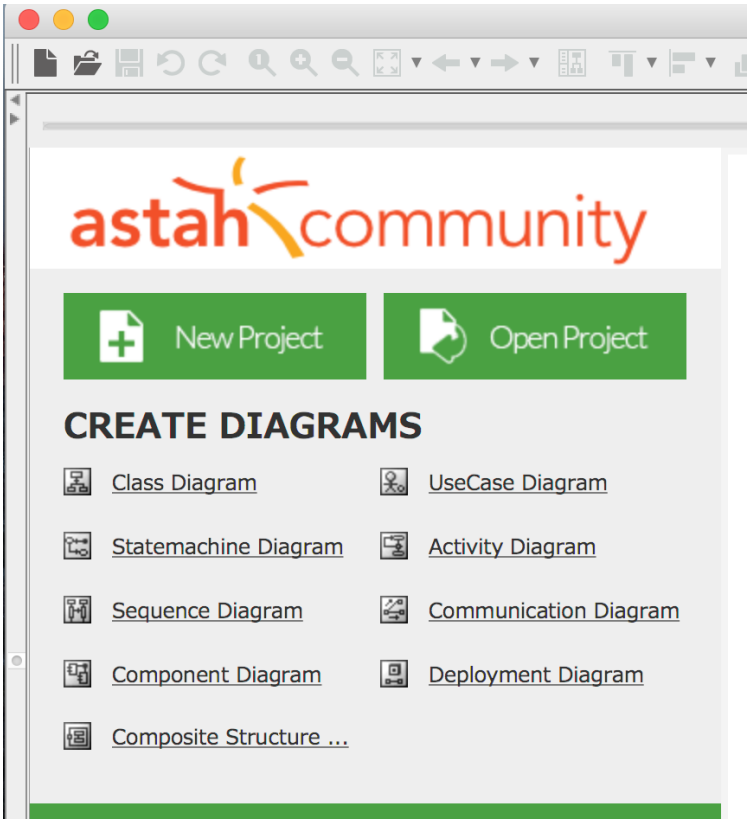
Gambar 7.4 Self Message

B. Langkah Praktikum

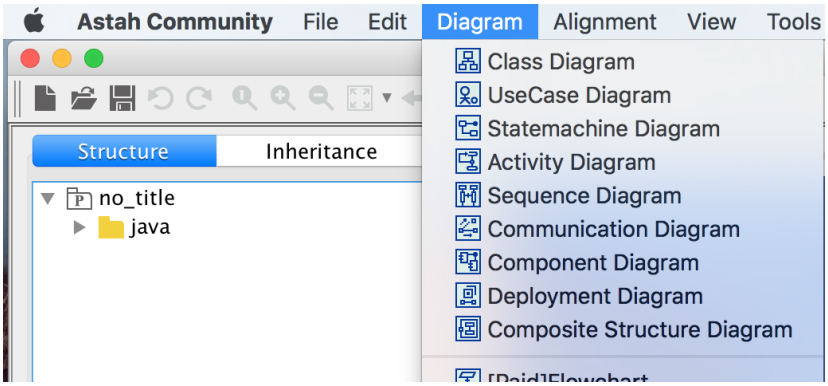
1. Berdasarkan studi kasus pengembangan aplikasi yang dipilih di kelas, buatkan sequence diagram.

2. Menggambarkan sequence diagram menggunakan alat bantu astah community

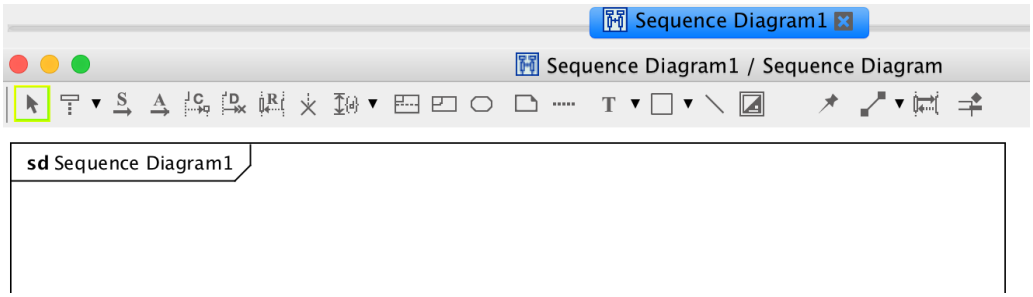
a. Buat Project Baru



b. Pilih Menu Diagram dan pilih sequence diagram



c. Notasi sequence diagram dapat ditemui pada toolbar jendela sequence diagram



d. Buat gambar sequence diagram nya.

C. Evaluasi

Buat laporan sequence diagram sesuai dengan tugas proyek anda.

Nilai	Yogyakarta, ……. Paraf asisten (……)

jika diperlukan lembar jawaban lebih mahasiswa bisa menyisipkan kertas sendiri

D. Referensi

Dennis, Alan, Barbara Haley Wixom, and David Tegarden. *Systems analysis and design: An object-oriented approach with UML*. John Wiley & Sons, 2015.

Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John Wiley & Sons, 2014.

http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_sequencediagram.html

SOFTWARE COSTING DAN ESTIMASI

Pertemuan ke **: VIII**

Alokasi Waktu: 1 Minggu & 1,5 jam

Tempat **: Laboratorium**

Kompetensi Dasar **: 1.** Mampu memahami ragam Software Costing dan Estimasi

2. Mampu menghitung Software Costing dan Estimasi berbasis usce case

Indikator **: 1.** Hasil analisis tertuang dalam perhitungan Software Costing dan Estimasi dari studi kasus yang ditentukan saat ini;

2. Terhitungnya Software Costing dan Estimasi yang diharapkan untuk menyelesaikan permasalahan.

A. Teori Pendukung

Model Use Case Point (UCP) pertama kali dipopulerkan oleh Kerner pada tahun 1993. Model UCP terinspirasi oleh model Function Points(FP) tetapi dengan manfaat dari analisis persyaratan dalam proses Objectory. UCP dimulai dengan mengukur fungsionalitas sistem berdasarkan pada model use case dalam sebuah hitungan yang disebut dengan Unadjusted Use Case Point (UUCP).Faktor teknis yang terlibat dalam mengembangkan fungsi ini dinilai, mirip dengan FP. Langkah terakhir dalam estimasi,namun tidak dari FP dan itu adalah faktor yang disebut Environmental Factorbaru yang diusulkan oleh penulis. Faktor ini tampaknya sangat penting menurut pengalaman pengguna Objectory (Karner 1993).

UUCP - Unadjusted Use Case Point

Untuk menghitung UUCP dilakukan dengan menilai setiap aktor pada Use Case.Hasil penilaian berupa nilai sederhana, rata-rata atau kompleks dengan bantuan dari Tabel 8.1 dan setiap case digunakan dengan bantuan dari Tabel 8.2.

Tabel 8.1 Penilaian terhadap Weighted actors.

Complexity	Definition	Weight
SIMPLE	An actor is simple if it represents another system with a defined application programming interface.	1
AVERAGE	An actor is average if it is: 1. An interaction with another system through a protocol	2

	2. A human interaction with a line terminal.	
COMPLEX	An actor is complex if it interacts through a graphical user interface.	3

Tabel 8.2 Penilaian terhadap Weighted use cases.

Complexity	Definition	Weight
SIMPLE	A use case is simple if it has 3 or less transactions including alternative courses. You should be able to realise the use case with less than 5 analysis objects.	5
A VERAGE	A use case is average if it has 3 to 7 transactions including alternative courses. You should be able to realise the use case with 5 to 10 analysis objects.	10
COMPLEX	A use case is complex if it has more than 7 transactions including alternative courses. The use case should at least need 10 analysis objects to be realised.	15

Nilai UUCP dihitung dengan menjumlahkan hasil dari Tabel 8.1 dan Tabel 8.2 dengan rumus berikut:

$$UUCP = \sum_{i=1}^{6} n_i * W_i$$ .....(1)

dimana ni adalah jumlah item dari berbagai i.

TCF - Technical Complexity Factor

TCF diperoleh dengan memberikan penilaian terhadap Tabel 3 dengan skala 0, 1, 2, 3, 4, dan 5 pada masing-masing itemnya.

Tabel 8.3 Faktor yang berkontribusi terhadap kompleksitas.

Fi	Factors Contributing to Complexity	Wi
F1	Distributed systems.	2
F2	Application performance objectives, in either response or throughput.	1
F3	End user efficiency (on-line).	1
F4	Complex internal processing.	1

F5	Reusability, the code must be able to reuse in other applications.	1
F6	Installation ease.	0.5
F7	Operational ease, usability.	0.5
F8	Portability.	2
F9	Changeability.	1
F10	Concurrency.	1
F11	Special security features.	1
F12	Provide direct access for third parties	1
F13	Special user training facilities	1

Nilai TCF dihitung dengan rumus berikut:

$$TCF = C_1 + C_2 \sum_{i=1}^{13} F_i * W_i$$

.....(2)

Dimana nilai C_1 = 0.6 dan nilai C_2 = 0.01. Nilai C_1 merupakan sebuah konstanta dan bobot yang diusulkan oleh Albrecht pada tahun 1979 tetapi C_1 diturunkan dari 0.65 menjadi 0.6 agar sesuai dengan jumlah faktor (Karner 1993). F_i adalah sebuah faktor yang dinilai pada skala 0, 1, 2, 3, 4 dan 5. 0 berarti tidak relevan dan 5 berarti ini sangat penting. Jika factor tidak penting atau tidak relevan maka akan memiliki nilai 3. Jika semua factor mempunyai nilai 3 maka nilai TCF akan setara dengan 1.

EF - Environmental Factor

EF membantu untuk mengestimasi seberapa efisien proyek tersebut. Faktor ini adalah bentuk yang sama sebagai faktor teknis. EF dihitung berdasarkan Tabel 8.4.

Tabel 8.4 Faktor yang berkontribusi terhadap efisiensi.

Fi	Factors contributing to efficiency	Wi
F1	Familiar with Objectory	1.5
F2	Part time workers	-1
F3	Analyst capability	0.5
F4	Application experience	0.5

F5	Object oriented experience	1
F6	Motivation	1
F7	Difficult programming language	-1
F8	Stable requirements	2

Nilai EF dihitung dengan rumus berikut:

$$EF = C_1 + C_2 \sum_{i=1}^{8} F_i * W_i$$ .....(3)

Dimana nilai C_1 = 1.4 dan nilai C_2 = -0.03. Fi adalah sebuah faktor yang dinilai pada skala 0, 1, 2, 3, 4 dan 5. 0 berarti tidak relevan dan 5 berarti ini sangat penting. Jika factor tidak penting atau tidak relevan maka akan memiliki nilai 3. Jika semua factor mempunyai nilai 3 maka nilai EF akan setara dengan 1.

Hasil dan Analisis

Use Case Point (UCP) dihitung dengan rumus berikut:

UCP = UUCP * TCF * EF. .....(4)

Berdasarkan UCP dihitung dengan melihat statistik dari proyek-proyek sebelumnya untuk melihat berapa banyak sumber daya yang dibutuhkan per UCP. Setelah itu dikalikan jumlah UCP dengan Mean Resources needed per UCP (MR). Nilai ini juga dilihat dengan menggunakan Standard Deviation of the MR (SDMR) untuk melihat seberapa baik estimasinya.

B. Langkah Praktikum

1. Persiapkan Use Case Diagram anda.
2. Lakukan perhitungan Use Case Point dengan alat bantu yang disediakan.

C. Evaluasi

Buat laporan diagram use case sesuai dengan tugas proyek anda.

Nilai	Yogyakarta, ……. Paraf asisten

	(……)

jika diperlukan lembar jawaban lebih mahasiswa bisa menyisipkan kertas sendiri

D. Referensi

Chemuturi, M. (2009). *Software estimation best practices, tools & techniques: A complete guide for software project estimators*. J. Ross Publishing.