POLMANBABEL PRESS

Pemodelan Perangkat Lunak
Behavior Diagram

Use Case

Activity

State Machine

Sidhiq Andriyanto

Pemodelan Perangkat Lunak
Behavior Diagram

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak ekslusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Perlindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;

ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;

iii. Penggandaan Ciptaan dan/atau produk Hak Terkait untuk keperluan pengajaran, kecuali pertunjukkan dan Fonogram yang telah dilakukan Pengumuman seebagai bahan ajar; dan

iv. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100.000.000 (serratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).

PEMODELAN PERANGKAT LUNAK
BEHAVIOR DIAGRAM

SIDHIQ ANDRIYANTO

POLITEKNIK MANUFAKTUR NEGERI
BANGKA BELITUNG
2022

PEMODELAN PERANGKAT LUNAK *BEHAVIOR DIAGRAM*
Sidhiq Andriyanto

Pengarah *: I Made Andik Setiawan*
Editor *: Mardliyah Ayu*
Proofreader *: Subkhan*
Desain Isi *: Sidhiq Andriyanto*
Desain Cover *: Muhammad Zenda Rud*

Jumlah halaman *: xii, 53 halaman*
Ukuran buku *: 15 x 21 cm*
Cetakan Pertama *: 2022*
ISBN *: 978-623-97870-8-0*

Penerbit
Politeknik Manufaktur Negeri Bangka Belitung
Kawasan Industri Air Kantung, Sungailiat, Bangka
polmanBabelpress@polman-Babel.ac.id
Telp/Faks: (0717) 93586

Kata Pengantar

Alhamdulillah, segala puji bagi Allah yang telah melimpahkan rahmat, nikmat dan petunjuk Nya sehingga buku Pemodelan Perangkat Lunak *Behavior Diagram* dapat terselesaikan dengan baik. Penulis juga mengucapkan banyak terimakasih kepada pihak-pihak yang telah membantu dalam penyusunan hingga terbitnya buku ini.

Buku ini mengulas tentang diagram *Unified Modeling Language* dengan fokus pada *behavior diagram*. Dalam kategori ini terdapat tiga diagram yakni *Use Case Diagram*, Activity, dan *State Machine Diagram*. Buku ini berisi tentang teori dasar, langkah penggunaan aplikasi pendukung, penyelesaian studi kasus, latihan teori dan tugas mandiri.

Buku ini dapat dibaca oleh siswa, mahasiswa atau umum yang bertujuan untuk memahami bagaimana membuat model dari perangkat lunak. Model tersebut menggunakan UML dengan aplikasi pendukung StarUML. Setiap tahapan perlu dibaca dengan seksama, sehingga pembaca dapat menerapkan pada kasus yang dimilikinya.

Sungailiat, Juli 2022

Penulis

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1
Pendahuluan

1. Penjabaran Mata Kuliah

Pemodelan Perangkat Lunak adalah salah satu mata kuliah wajib program studi Teknologi Rekayasa Perangkat Lunak di Politeknik Manufaktur Negeri Bangka Belitung. Pembelajaran mata kuliah ini dilaksanakan dengan teori dan praktik.

Pemodelan Perangkat Lunak merupakan suatu upaya untuk membuat gambaran abstrak dari sistem dimana persyaratan sistem tersebut akan atau sedang dianalisa oleh pengembang perangkat lunak. Dalam pemodelan dapat dilakukan dengan menggambar dengan alat tulis atau menggunakan aplikasi pendukung.

Tujuan dari mata kuliah ini adalah mahasiswa dapat menggambarkan konsep pemodelan perilaku, pemodelan data dan pemodelan obyek dari sistem yang akan dianalisa. Selain itu memperkenalkan notasi-notasi Unified Modeling Languages (UML) yang ada dalam pemodelan berbasis obyek.

Pemodelan berorientasi obyek merupakan upaya mengorganisasikan perangkat lunak menjadi kumpulan obyek yg bekerja sama. Kerjasama tersebut melibatkan informasi atau struktur data dengan perilaku yang mengaturnya.

Pemodelan termasuk dalam tahapan yang penting dalam bagian perancangan perangkat lunak. Untuk itu kita harus bisa mentransformasikan hasil analisis menjadi bentuk model yang tepat dan jelas. Sehingga model

tersebut mudah dipahami dan diimplementasikan di tahapan pemrograman. Model-model yang dibuat tentunya menjadi gambaran dari perangkat lunak yang akan dikembangkan.

2. Pemodelan Berorientasi Obyek

Unified Modeling Language merupakan salah satu bahasa yang dapat digunakan untuk menganalisa dan membuat desain berorientasi obyek. Perumusan UML ada berdasarkan kebutuhan pemodelan visual untu mengatur spesifikasi, menggambarkan, membangun, dan perekaman dari sistem perangkat lunak.

Penggunaan UML memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa. UML dapat menyatukan praktek-praktek terbaik dalam pemodelan. Model yang siap pakai, bahasa pemodelan visual yang ekspresif dan mudah dipahami oleh umum.

Berdasarkan literatur, UML muncul pada akhir tahun 1980. Pada saat itu Grady Booch memperkenalkan metode Booch yang digabung dengan teknik pemodelan dari Jame Rumbaugh. Sedangkan Ivan Jacobson turut ikut memperkenalkan prinsip dan metodologi lainnya.

Sehingga pada tahun 1997, UML versi 1 resmi diperkenal kepada umum. Metodologi ini memiliki standar pada setiap diagramnya. Kemudian ada perubahan pada UML versi 2.0 tahun 2005 yang berfokus pada pendekatan berorientasi obyek. Hingga pada tahun 2017, UML dengan versi 2.5 terus disempurnakan agar dapat mengikuti permasalahan dan perkembangan teknologi saat ini.

Dalam UML versi 2.0 terdapat tiga belas diagram yang terbagi menjadi tiga kategori, yaitu;

1) *Behavior Diagram* yang terdiri dari *Use Case Diagram, Activity Diagram dan State Machine Diagram.*
2) *Structure Diagram: Class Diagram, Object Diagram, Composite Structure Diagram, Component Diagram, Package Diagram,* dan *Deployment Diagram*.
3) *Interaction Diagram* terdiri dari semua *Behavior Diagram* yang lebih umum, *Sequence Diagram, Timing Diagram, Communication Diagram,* dan *Interaction Overview Diagram*.

Pemodelan perangkat lunak yang akan dibahas dalam buku ini yaitu *Use Case Diagram, Activity Diagram,* dan *State Machine Diagram*. Materi akan disampaikan dengan tahapan sebagai berikut:

1) Pemahaman teori dasar dari tiga diagram tersebut. Tujuannya agar pembaca dapat memahami konsep dasar dan notasi yang digunakan
2) Penggunaan aplikasi pendukung dalam memodelkan diagram
3) Mempelajari studi kasus satu dan dua yang telah disediakan
4) Menjawab latihan yang diberikan untuk penguatan pemahaman diagram
5) Mengerjakan tugas mandiri sesuai kasus yang pembaca tentukan sendiri. Tahapan ini diharapkan pembaca dapat menggambarkan diagram sesuai kasus dengan tepat.

3. Aplikasi Pendukung

Banyak sekali aplikasi pendukung berbasis *dekstop* atau *online* yang dapat membantu dalam menggambar pemodelan UML. Beberapa aplikasi yang dapat digunakan antara lain, StarUML, Visual Paradigm, Microsoft Visio, UML Designer Tool, Astah dan lainnya. Sedangkan untuk media *online,* kita dapat menggunakan EdrawMax, Moqups, Adobe Spark dan masih banyak lainnya. Mahasiswa dapat menggunakan aplikasi pendukung yang berbeda dengan aplikasi pendukung dalam kegiatan praktik. Karena pada dasarnya notasi yang digunakan memiliki bentuk yang sama.

Dalam pembahasan di buku ini, aplikasi yang digunakan adalah *staruml.* Aplikasi ini dapat diunduh secara gratis di halaman *www.staruml.io*. Platform yang didukung antara lain Windows, macOs, dan Linux (Ubuntu/Fedora).

Gambar 1. Logo StarUML

Beberapa fitur yang ada di StarUML di versi 5, antara lain:

- Kompatibel dengan metamodel dan diagram standar 2.x.
- Adanya dukungan pemodelan dengan menggunakan diagram SysML: Persyaratan,

Definisi Blok, Blok Internal, dan Diagram Parametrik.

- Dapat membuat diagram ERD dan DFD.
- Retina Display Support. Dapat memperbesar tampilan layar berbentuk lingkaran (RETINA).
- Extension Manager. Kemudahan instalasi ekstensi pihak ketiga.
- Pengembangan Model-Drive. Data pemodelan disimpan dalam format JSON yang sangat sederhana, sehingga dapat digunakan dengan mudah untuk menghasilkan kode khusus dengan CLI (Command-Line Interface).
- Validasi Model *Asynchronous*. Banyak aturan validasi model didefinisikan dan diperiksa secara asinkron setiap kali Anda menyimpan atau membuka *file* model Anda.

Bab 2

Use Case Diagram

Capaian Pembelajaran - Sub CPMK

- ✓ Menjelaskan syarat *Use Case* dan *dependency*
- ✓ Menjabarkan fungsi *Actor* dan generalisasi
- ✓ Membuat rancangan perangkat lunak menggunakan *Use Case Diagram*

1. Dasar *Use Case Diagram*

Pokok bahasan materi kuliah adalah memahami notasi dan bagaimana membuat sebuah pemodelan perangkat lunak menggunakan *use case diagram*. Tentunya anda harus melakukan analisis terlebih dahulu terhadap proses bisnis yang terjadi di sebuah organisasi.

Secara lebih rinci, anda dapat mengidentifikasi syarat *use case*, dependensi, *actor*, dan menbuat asosiasi antara *actor* dengan *case* yang dimilikinya. Selain itu anda diharapkan dapat menggunakan aplikasi pendukung dalam perancangan sesuai dengan petunjuk yang benar.

Diagram *use case* digunakan untuk memodelkan perilaku dinamis dari sistem pada saat beroperasi (*running*). Diagram ini terdiri dari interaksi faktor internal dan eksternal. Agen internal dan eksternal disebut *actor*.

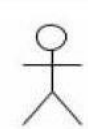
Dalam pemodelannya, diagram *use case* terdiri dari *actor*, *use case* dan relasi. Diagram *use case* dapat memodelkan sistem/sub sistem. Setiap satu diagram

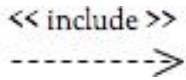
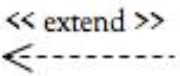
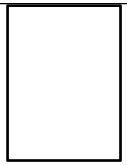
menjelaskan fungsional tertentu dari sistem. Sehingga keseluruhan sistem bisa terdiri dari sejumlah diagram *use case*.

Tujuan diagram *use case* antara lain;

- Memodelkan bentuk persyaratan-persyaratan yang ada agar menjadi sebuah sistem.
- Mendapatkan tampilan proses bisnis dari sistem.
- Mengidentifikasi faktor internal dan eksternal yang terlibat.
- Menampilkan interaksi persyaratan.

Tabel 1. Notasi *Use Case Diagram*

Simbol	Keterangan
	*Actor* Mewakili peran sistem lain, orang, atau alat ketika berkomunikasi dengan *Use Case*
	*Use Case* Abstraksi dan interaksi antara *Actor* dengan system
	Association Hubungan antara obyek satu dengan obyek lainnya
	*Dependency* Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri

	akan mempengaruhi elemen uang bergantung padanya elemen yang tidak mandiri
	*Include* Menspesifikasikan bahwa *Use Case* sumber scara eksplisit
	*Generalization* Interaksi aturan-aturan dengan elemen lain
	*Extend* Menspesifikasikan bahwa *Use Case* target memperluas perilaku *Use Case* sumber pada suatu titik yang diberikan
	*System* Menspesifikasikan paket yang menampilkan sistem secara terbatas

Actor merupakan representasi dari pengguna, waktu atau sistem eksternal yang berinteraksi. Contohnya seperti manajer proyek, sumber daya manusia dan admin. Waktu dapat menjadi *actor* ketika waktu dapat memicu sebuah kejadian (*event*) saat penggunaan sistem. Hal yang perlu diidentifikasi antara lain selain *actor* yaitu fungsionalitas (*use case*), relasi, dependensi.

Ada beberapa pertanyaan yang dapat membantu kita dalam mencari *actor*. Pertanyaan pertama adalah siapa yang akan menggunakan sistem tersebut. Pertanyaan

kedua yaitu apakah sistem itu dapat memberikan sebuah nilai (value) kepada *actor*. Sebagai pengingat, penamaan *actor* diberi nama sesuai tugas atau perannya di dalam sistem.

Pembuatan diagram *use case* harus efektif seperti penamaan *use case* yang sederhana dan mengidentifikasi bagaimana fungsi dijalankan. Begitu juga dengan penamaan *actor* harus tepat, penunjukkan dependensi dan relasi yang jelas. Anda tidak perlu memasukkan semua tipe relasi yang ada, sebab tujuan utamanya adalah mengidentifikasi persyaratan.

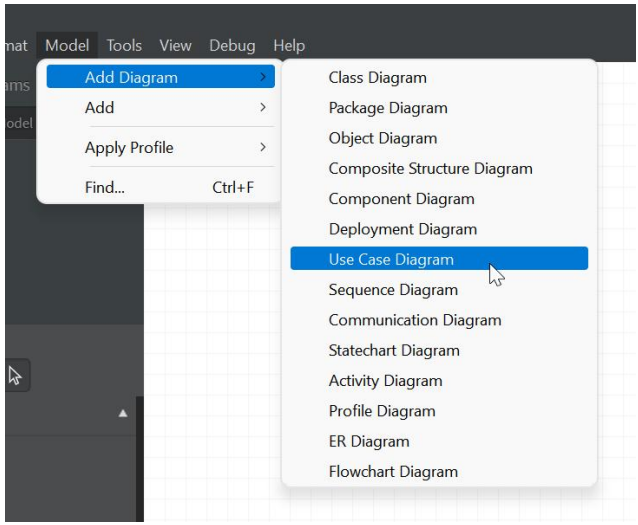
Dependensi *include* seperti aktifitas manajer proyek, manajer sumberdaya, admin sistem yang berinteraksi pada sistem manajemen proyek. *Use case* dengan perilaku umum. Dependency *include* menyatakan *use case* basis akan memasukkan atau memanggil *use case* lain.

Dependency *extend* bisa diterapkan jika dari satu *use case* (disebut *use case extend*) ke *use case* lain (disebut *use case* basis) mengindikasikan *use case* extend akan menyisipkan dan menambah *use case* basis.

2. Pemodelan

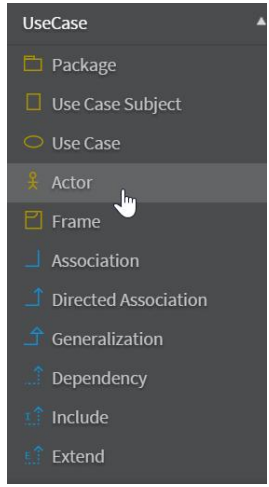
a) Aplikasi Pemodelan

- Buka aplikasi StarUML
- Pilih *use case diagram*



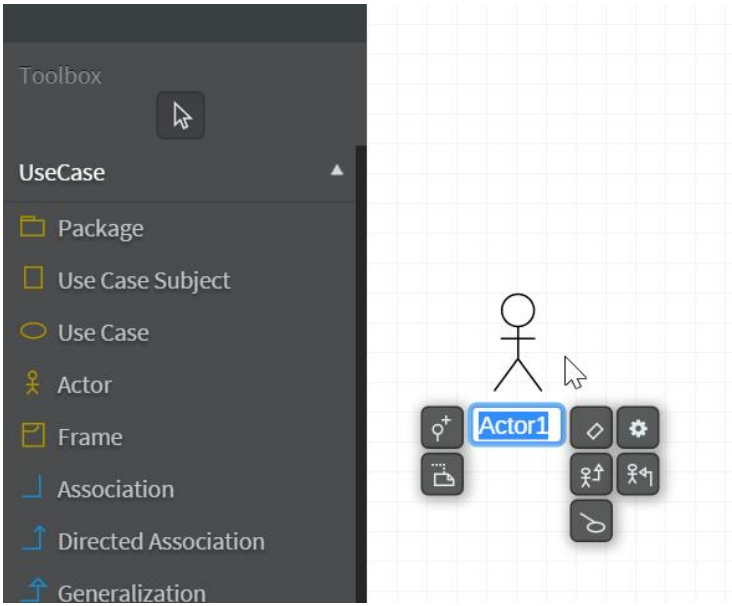
Gambar 2. Menambahkan Diagram

- Maka *toolbox use case* akan muncul seperti gambar di bawah ini:



Gambar 3. *Actor*

- Klik *Actor*, kemudian klik lagi di area kerja. Beri nama *Actor*1 dengan Mahasiswa

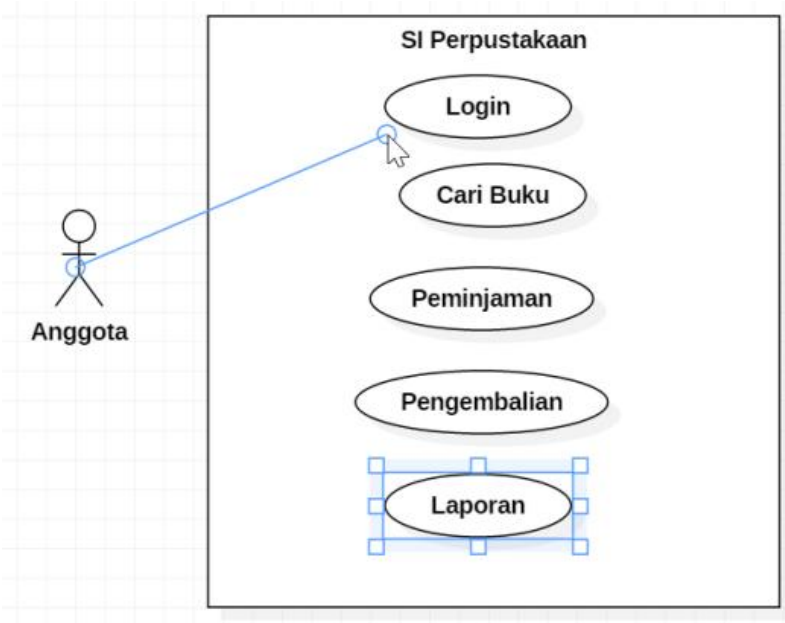


Gambar 4. Nama *Actor*

- Klik *use case* dan letakkan

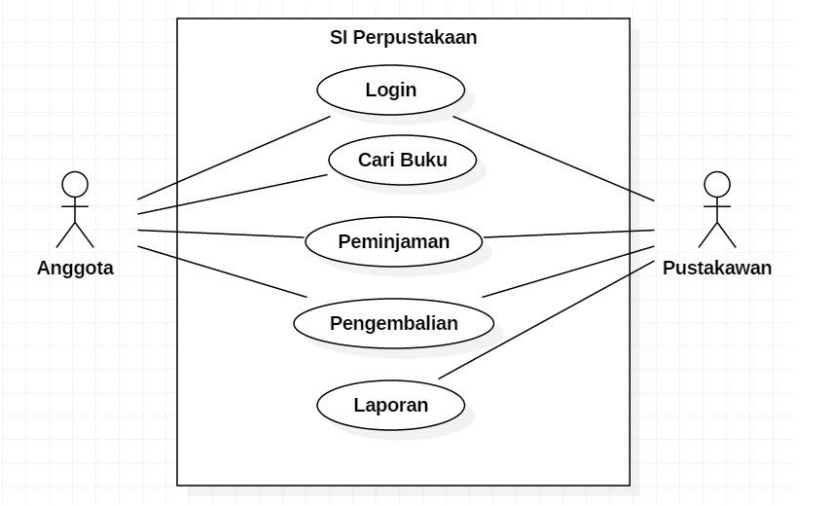
Gambar 5. Nama *Use Case*

- Klik *association*. Kemudian klik dari *Actor* dan drag menuju UseCase. Lalu lepas kursor.



Gambar 6. Asosiasi

● Buatlah *Use Case* seperti ini



Gambar 7. *Use Case* Perpustakaan

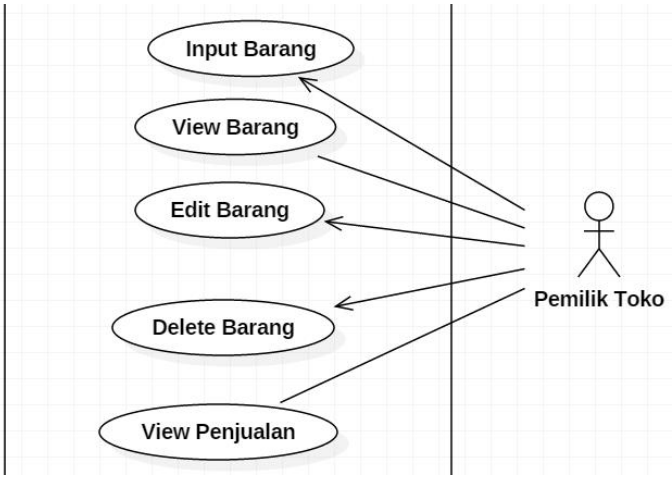
b) Studi Kasus 1

Sistem Informasi Penjualan

"*Pak Andi memiliki sebuah toserba di depan rumahnya. Setiap hari toserba itu melayani penjualan hingga 100-150 pelanggan. Tentunya pegawai toserba kewalahan dalam melayani karena masih menggunakan buku dalam pencatatannya. Sehingga pelanggan harus menunggu untuk dilayani kebutuhannya.*"

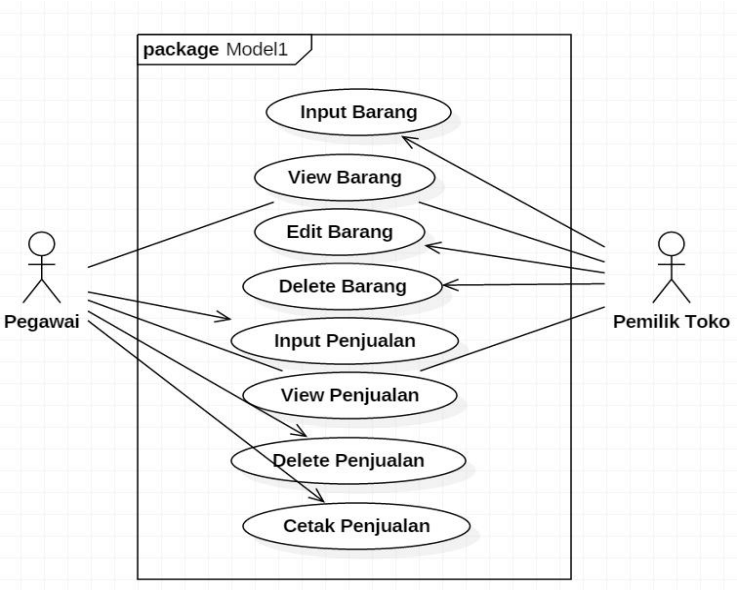
Dari kasus seperti di atas, kita analisis terlebih dahulu permasalahannya antara lain adalah proses pencatatan data penjualan masih menggunakan buku (manual).

Selanjutnya, kita tentukan kegiatan yang ada atau proses bisnis yang terjadi di toserba. Contohnya seperti penjualan barang dan cetak laporan. Kedua proses bisnis ini menjadi hal paling penting dalam kasus tersebut.



Gambar 8. *Use Case* Pemilik Toko

Sebagai pemilik toko, proses *input* barang, edit barang dan delete barang memiliki asosiasi terarah (*directed association*). Karena pemilik toko yang memiliki hak akses penuh terhadap proses tersebut. Pemilik toko dapat melihat data barang dan data penjualan.



Gambar 9. *Use Case* Sistem Informasi Penjualan

Pegawai juga memiliki tiga asosiasi terarah yakni *input* penjualan, delete penjualan dan cetak penjualan. Pegawai dapat melihat data barang dan data transaksi penjualan yang ada di sistem.

Tentunya pemodelan di atas berdasarkan proses bisnis yang berjalan di sebuah organisasi. Hasil pemodelan bisa berbeda dengan organisasi lain meskipun bidangnya sama.

c) Studi Kasus 2

Sistem Informasi Penerimaan Siswa Baru

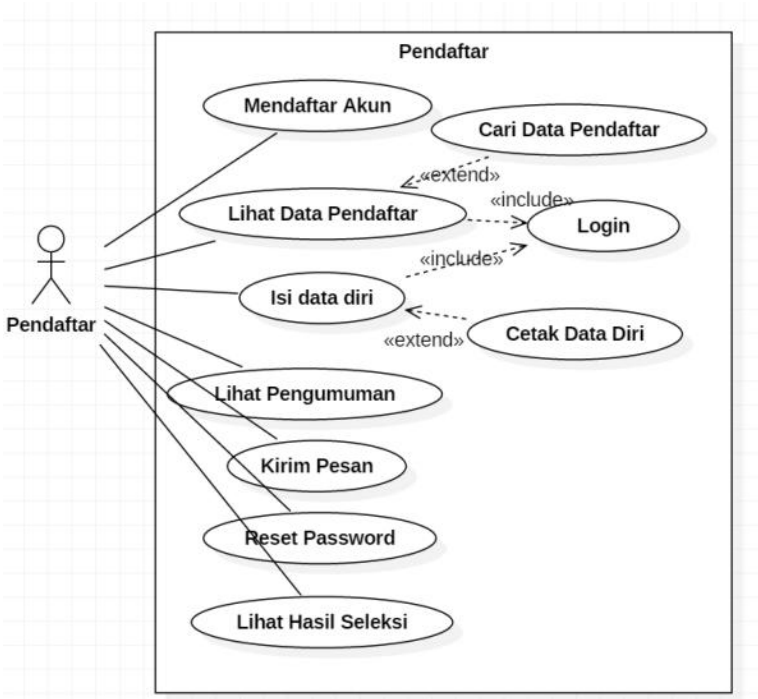
"SMP xyz ingin memanfaatkan teknologi informasi dalam proses penerimaan siswa baru pada tahun ajaran ini. Supaya memudahkan

pendaftar dalam mendaftarkan dirinya secara online dari rumah."

Untuk itu kita perlu menjabarkan apa saja yang bisa dilakukan oleh pendaftar dalam sistem. Tentunya harus memudahkan pendaftar yang kita anggap masih awam dalam menggunakan sistem informasi online.

Kebutuhan pendaftar dalam sistem dapat kita jabarkan seperti di bawah ini, yaitu:

- Pendaftar dapat mendaftarkan akunnya
- Dapat melihat informasi pendaftar yang ada
- Bisa mengisi data diri dan mencetaknya
- Melihat pengumuman online
- Mengirim pesan kepada panitia
- Mengatur ulang passwordnya
- Melihat hasil seleksi



Gambar 10. *Use Case* Pendaftar

Terdapat relasi extend antara cari data pendaftar dengan lihat data pendaftar karena *use case* lihat data pendaftar dapat berdiri sendiri tanpa harus menjalankan *use case* cari data pendaftar. Hal yang sama terjadi pada *use case* isi data diri dengan cetak data diri.

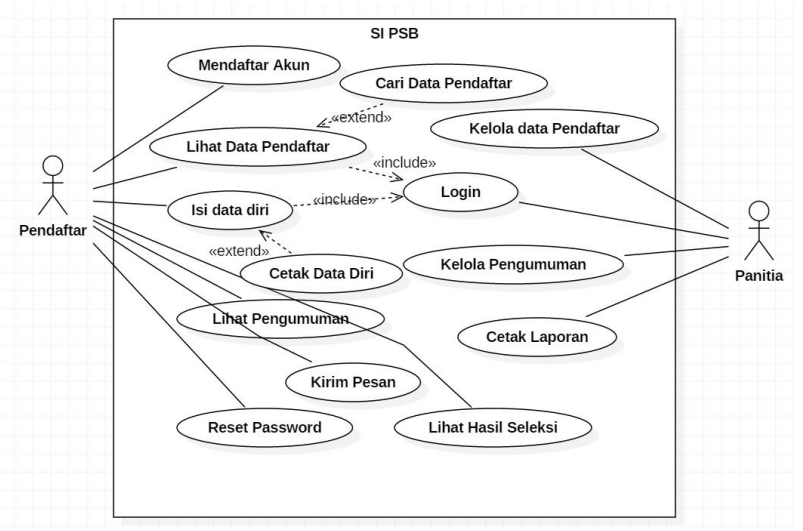
Use case lihat data pendaftar sangat bergantung pada *use case* login. Jika tanpa login, maka pendaftar tidak akan bisa melakukan *use case* lihat data pendaftar.

Actor dalam sistem penerimaan siswa baru selain pendaftar tentunya adalah panitia PSB dan

kepala sekolah. Selanjutnya kita tentukan kebutuhan dari panitia PSB dalam sistem tersebut. Kebutuhan panitia PSB antara lain sebagai berikut:

- Kelola data pendaftar.
- Kelola pengumuman.
- Cetak laporan. Laporan data keseluruhan pendaftar yang ada, laporan data pendaftar yang lolos seleksi.

Berdasarkan kebutuhan di atas, maka kita dapat memodelkannya seperti gambar di bawah.



Gambar 11. *Use Case* Penerimaan Siswa Baru

Actor Panitia dapat mengelola data pendaftar yang di dalamnya ada proses *input, read, update, delete*. Begitu juga di *Use Case* kelola

pengumuman. Panitia dapat mencetak semua laporan yang ada.

d) Latihan

- Jelaskan tentang include, extend, dependency, dan generalization!
- Berikan contoh penggunaannya!

e) Tugas Mandiri

- Buatlah diagram *Use Case* menggunakan kasus yang memiliki minimal 3 *Actor*.
- Berikan penjelasan analisisnya!

Bab 3

Activity Diagram

Capaian Pembelajaran - Sub CPMK

- ✓ Mampu menjabarkan elemen diagram aktifitas dengan benar sesuai sumber yang ada
- ✓ Mampu merincikan aksi/aktifitas dengan benar

1. Dasar *Activity Diagram*

Pembahasan pada modul 3 adalah urutan aktifitas, proses bisnis, *fork* dan *join* di dalam *activity diagram*. Tujuan praktik adalah mahasiswa dapat membuat model *activity diagram* dari kasus sesuai dengan proses bisnis yang berlaku.

Activity diagram atau diagram aktivitas merupakan salah satu pemodelan yang sangat bermanfaat dalam perancangan perangkat lunak. Karena dapat membantu kita dalam memahami proses bisnis sistem secara keseluruhan. Diagram ini bisa menunjukkan aktifitas-aktifitas proses bisnis yang bergantung satu sama lain.

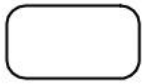
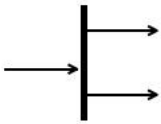
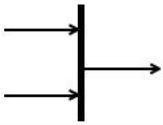
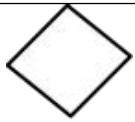
Fokus dari diagram aktivitas ini adalah menggambarkan aktivitas sistem yang ada dan bukan apa saja yang dilakukan *actor*. Diagram aktivitas lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Diagram ini dibuat berdasarkan sebuah atau beberapa *use case* yang telah dibuat sebelumnya dan bukan sebaliknya. Sebuah aktifitas dapat digambarkan dengan satu atau lebih *use case*. Perbedaan antara diagram

aktifitas dan *use case* adalah aktifitas menggambarkan atau memodelkan proses yang berjalan pada sistem, sedangkan *use case* menggambarkan bagaimana *actor* menggunakan sistem tersebut dalam melakukan aktifitas.

Pada tabel di bawah ini adalah notasi diagram aktivitas.

Tabel 2. Notasi *Activity Diagram*

Simbol	Keterangan
	*Initial* Awal dari kumpulan aktifitas atau aksi
	Final Menghentikan seluruh aliran kontrol atau aliran obyek pada sebuah aksi atau aktifitas
	Activities Kumpulan aktifitas atau aksi
	*Fork* (Percabangan) Memecah sebuah perilaku menjadi aksi atau aktifitas paralel
	*Join* (Penggabungan) Menggabungkan kembali menjadi satu aliran perilaku dari aksi atau aktifitas paralel
	*Decision* Mewakili suatu pengujian yang bertujuan untuk memastikan aliran

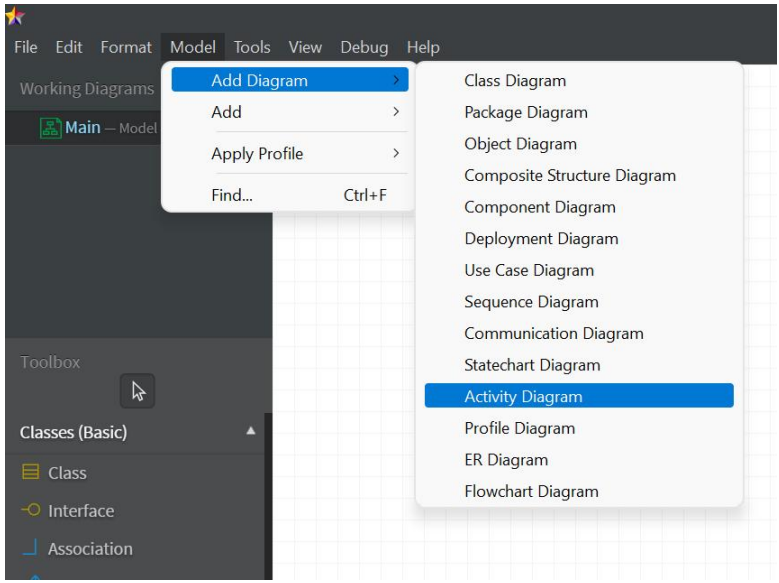
	kontrol dan aliran obyek menuju ke satu arah
Swimlane	Kolom yang berguna untuk mengelompokkan aktifitas tertentu berdasarkan *Actor*

2. Pemodelan *Activity Diagram*

a) Aplikasi Pemodelan

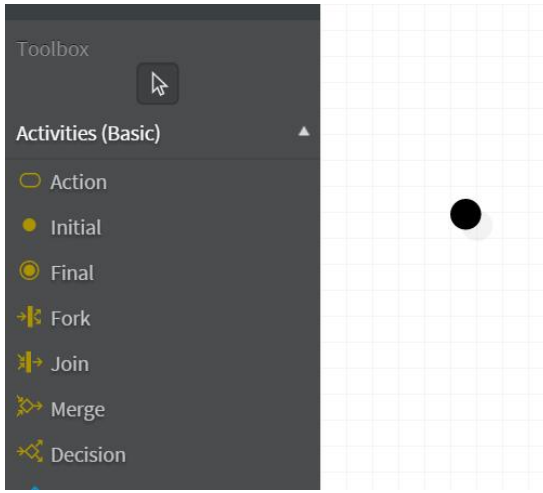
Buka aplikasi StarUML.

Klik *Model -> Add Diagram -> Activity Diagram.*



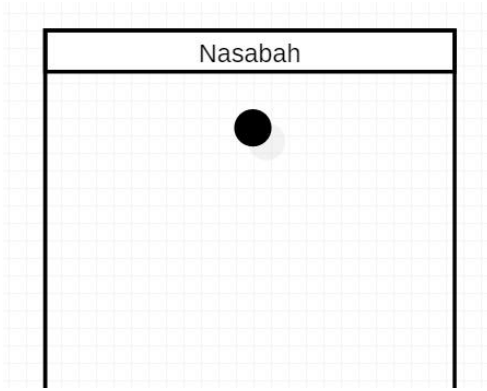
Gambar 12. *Activity Diagram*

- Mulailah dengan *node* awal untuk titik awal dengan klik *initial*



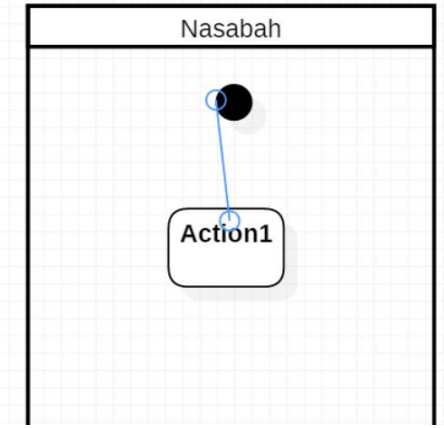
Gambar 13. *Initial*

- Tambahkan partisi jika relevan untuk analisis yang dibuat



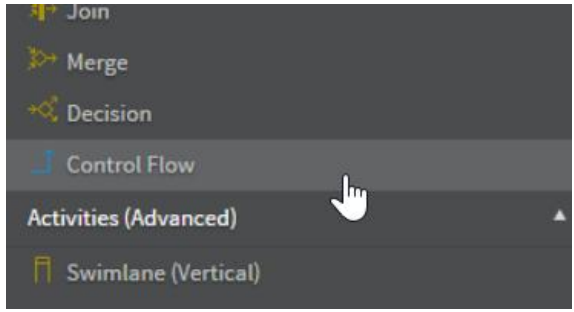
Gambar 14. *Swimlane*

- Tambahkan aksi untuk setiap langkah utama dari aktifitas

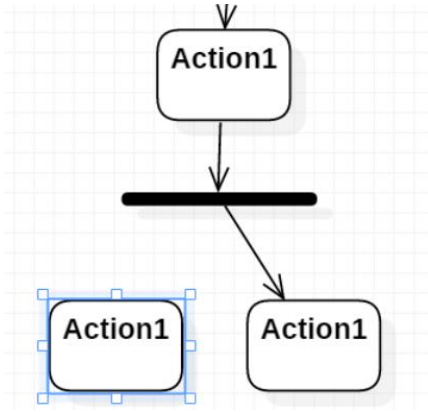


Gambar 15. Contoh Asosiasi

- Ketika menambahkan *fork*, drag *control flow* dari aksi sebelumnya menuju *fork*. Begitu juga dari *fork* menuju aksi berikutnya.

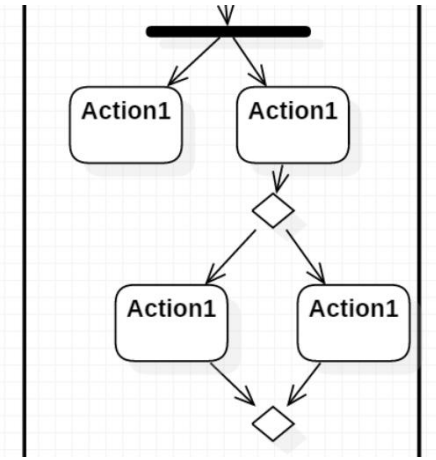


Gambar 16. Menu *Control Flow* (Asosiasi)



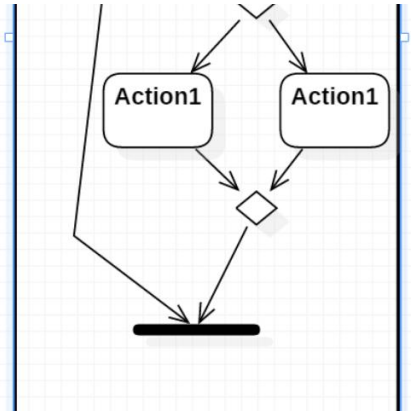
Gambar 17. Penggunaan *Fork*

- Tambahkan *decision* jika alur dipecah menjadi beberapa pilihan. Jangan lupa untuk menggabungkan kembali dengan *Merge*



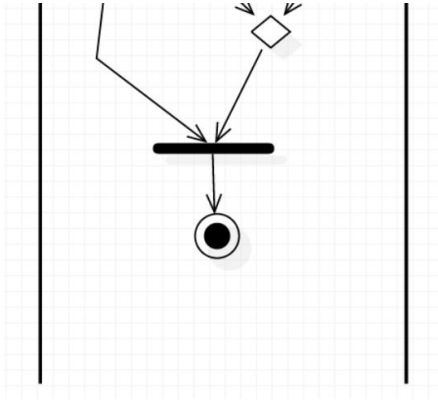
Gambar 18. Penggunaan *Merge*

- Tambahkan *Join* untuk menggabungkan aliran kontrol



Gambar 19. Penggunaan *Join*

- Akhiri proses dengan notasi untuk akhir aktivitas

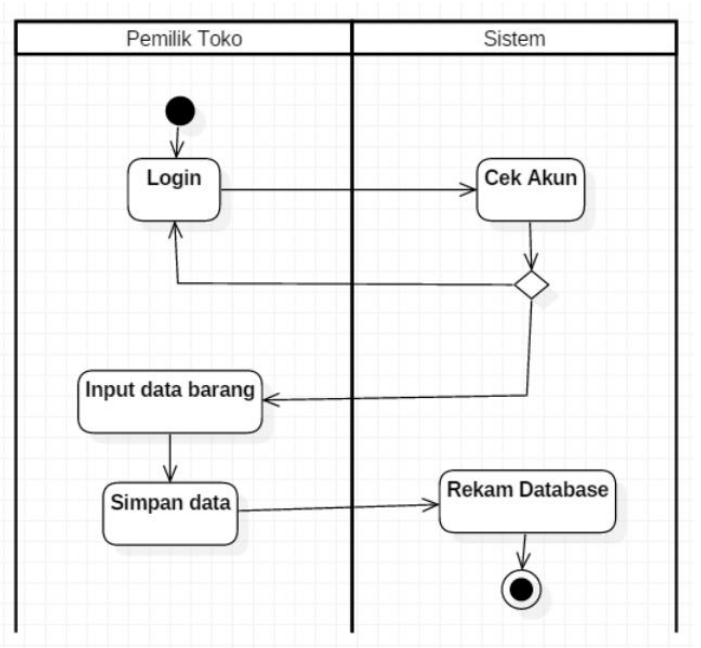


Gambar 20. Penggunaan *Final Node*

b) Studi Kasus 1

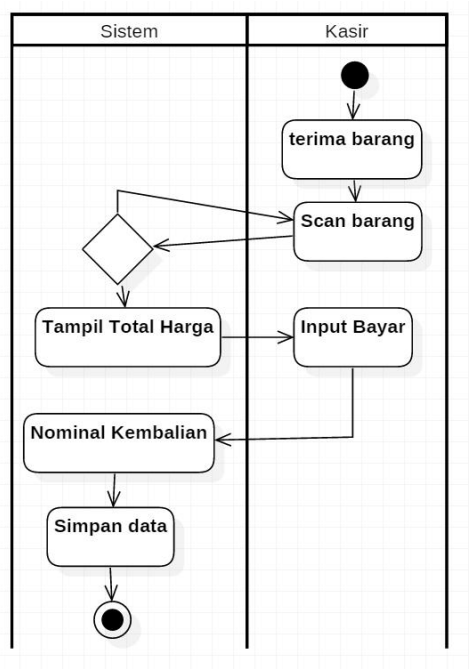
Pada sistem informasi penjualan barang kita akan membahas *activity diagram* untuk memasukkan data barang. Ada beberapa rangkaian aktifitas dalam memasukkan data barang, yaitu:

1) *Login*. Aktifitas ini bertujuan untuk masuk ke dalam sistem sebagai pemilik toko. Berdasarkan pemodelan *use case diagram* sebelumnya, pemilik toko memiliki hak akses sebagai admin di dalam sistem.
2) Cek akun. Aktifitas ini dilakukan sistem untuk mengecek username dan password yang digunakan sudah benar atau belum
3) *Decision*. Penentuan keputusan terhadap akun yang dimasukkan. Jika masukan tidak terdeteksi, maka kembali ke aktifitas login. Jika akun pemilik toko teridentifikasi, maka lanjut ke aktifitas berikutnya.
4) *Input* data barang. Pemilik toko memasukkan data barang sesuai permintaan sistem.
5) Simpan data. Kemudian pemilik toko akan menekan tombol simpan untuk menyimpan data barang ke dalam basis data.
6) Rekam basis data. Sistem akan menyimpan atau merekam data barang ke dalam basis data.



Gambar 21. Diagram *Input* Data Barang

Tentunya aktifitas Pemilik Toko tidak hanya sebatas memasukkan data barang. Ada aktifitas lainnya seperti menghapus, update, mencetak atau hanya melihat data barang saja.



Gambar 22 Pembelian Barang

Pada gambar 22 merupakan model dari pembelian barang. Alurnya dapat kita lihat di bawah ini:

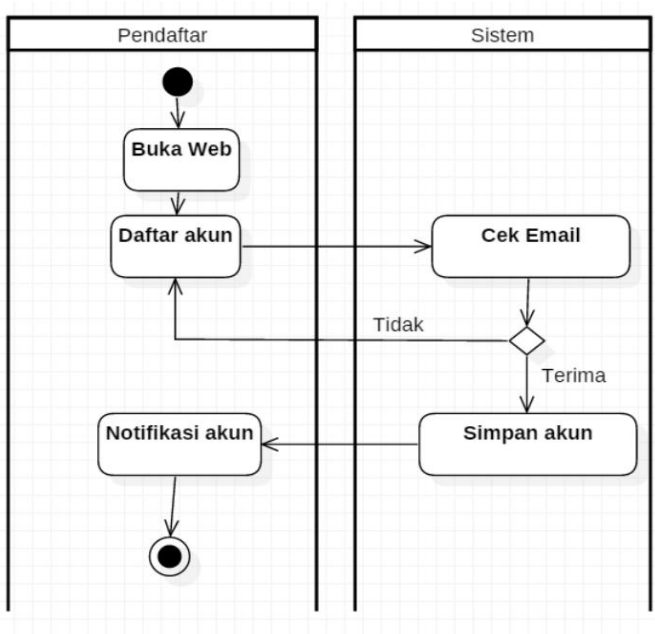
- Kasir menerima barang yang akan dibeli oleh pelanggan.
- Kasir melakukan scan barang tersebut.
- Jika sistem mengenali identitas dari barang, sistem akan menampilkan total harga yang harus dibayar pelanggan. Jika tidak dikenali, maka proses scan barang diulang.

- Kasir memasukkan nominal uang yang diberikan pelanggan.
- Sistem akan menentukan total kembalian yang harus diberikan kepada pelanggan.
- Sistem akan menyimpan data pembelian ke dalam basis data.

c) Studi Kasus 2

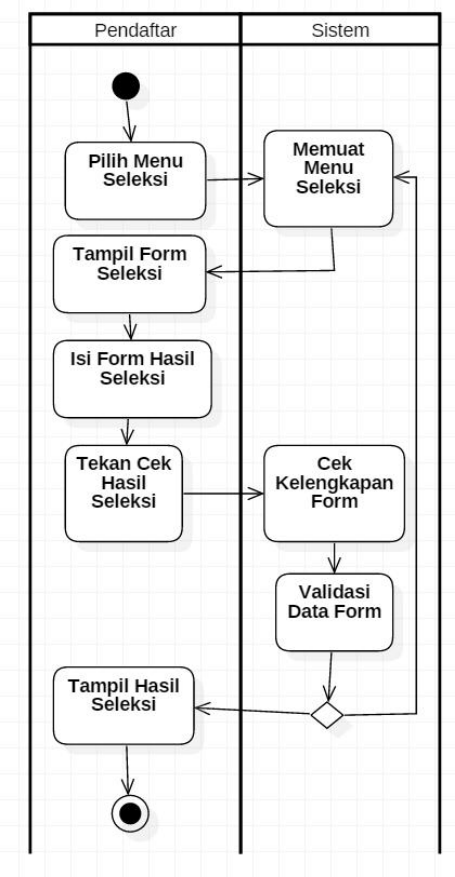
Studi kasus berikutnya kita akan membahas tentang pemodelan *Activity Diagram* pada sistem informasi penerimaan siswa baru.

Untuk mengakses sistem informasi tersebut, Pendaftar harus membuka website terlebih dahulu. Kemudian mengisi data-data yang diperlukan dalam pembuatan akun. Selanjutnya sistem akan memeriksa email yang digunakan. Apakah sudah terdaftar atau belum di basis data. Jika sudah terdaftar, maka Pendaftar harus mengulangi proses daftar akun. Namun jika belum terdaftar, sistem akan menyimpan akun dan mengirimkan notifikasi akun ke Pendaftar.



Gambar 23. Diagram Daftar Akun

Jadi bisa kita lihat berdasarkan diagram di atas. Aktifitas yang dibuat berkaitan langsung dengan sistem. Apa saja yang menjadi masukan ke dalam sistem sampai keluaran dari sistem untuk pengguna. Harap menjadi perhatian bahwa *Activity Diagram* menggambarkan atau memodelkan proses yang terjadi di dalam sistem.



Gambar 24 Menampilkan Hasil Seleksi

Diagram aktivitas di atas menggambarkan alur menampilkan hasil seleksi penerimaan siswa baru, seperti berikut:

- Pendaftar memilih menu seleksi
- Sistem memuat halaman seleksi.

- Pendaftar melihat form seleksi yang disediakan sistem.
- Pendaftar mengisi form seleksi dengan akun pendaftar.
- Sistem memeriksa akun yang telah dimasukkan. Apakah akun tersebut terdaftar atau tidak.
- Jika terdaftar, sistem akan menampilkan hasil seleksi kepada Pendaftar. Jika masukan salah, sistem akan memuat ulang halaman seleksi

d) Latihan

- Jelaskan perbedaan decision *node* dengan *fork*! Berikan contoh narasi
- Jelaskan perbedaan *merge node* dengan *join*! Berikan contoh narasi
- Jabarkan fungsi-fungsi *activity diagram* dalam pengembangan perangkat lunak!

e) Tugas Mandiri

- Buat *activity diagram* dari alur transaksi finansial perbankan yang terdapat di atm, yang meliputi:
- Informasi saldo
- Transfer rekening yang 1 bank
- Berikan penjelasan pada diagram tersebut

Bab 3

State Machine Diagram

Capaian Pembelajaran - Sub CPMK

- ✓ Mampu menjelaskan tentang *State*, *event*, *Composite State*, dan Transisi berdasarkan sumber yang ada
- ✓ Membuat pemodelan menggunakan *State Machine Diagram*

1. Dasar *State Machine Diagram*

Pokok bahasan pada bab ini adalah tentang *state*, *event*, *composite state* dan transisi. Diagram ini bertujuan untuk menggambarkan siklus elemen yang ada di dalam sistem. Lebih tepatnya pemodelan perilaku dari sebuah obyek tunggal yang menunjukkan urutan kejadian terhadap suatu obyek dalam merespon *event*.

Elemen dasar dari diagram ini adalah *state* dan transisinya. Elemen dapat berkomunikasi dengan elemen lain dalam satu obyek. Siklus dalam elemen dimulai dari:

- Membuat kondisi,
- Mengetahui sesuatu,
- Mengerjakan sesuatu
- Berkomunikasi dengan elemen lainnya untuk meminta proses
- Memiliki elemen lain yang dapat berkomunikasi untuk meminta proses dari elemen lain

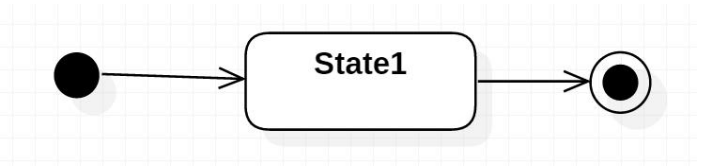
State merupakan sebuah kondisi atau sebuah situasi khusus dari suatu elemen selama siklus hidup. *State* aktif adalah *state* yang sedang berjalan dari elemen. Terdapat tiga *state* yaitu; simple, *initial*, final.

Simple *state* menyatakan kondisi atau situasi dari elemen. Notasi yang berbentuk kotak dengan sudut tumpul yang bertuliskan nama *state* di dalamnya.

Initial adalah notasi berbentuk lingkaran hitam sebagai *state* awal dari elemen. Sedangkan final *state* adalah notasi berbentuk lingkaran dengan titik hitam di tengahnya. Fungsinya sebagai *state* akhir dari elemen.

Transisi adalah notasi berbentuk panah yang menyatakan perpindahan dari satu sate ke *state* berikutnya. Transisi memiliki tiga bagian untuk labelnya antara lain *trigger [guard]*, dan *activity*.

- *Trigger* adalah sinyal kejadian yang memicu perubahan terhadap *state*
- *Guard* merupakan label dengan kondisi. Jika guard ada, maka kondisi boolean harus bernilai true sehingga trigger memicu transisi
- *Activity* adalah label yang terdapat beberapa perilaku dijalankan selama transisi

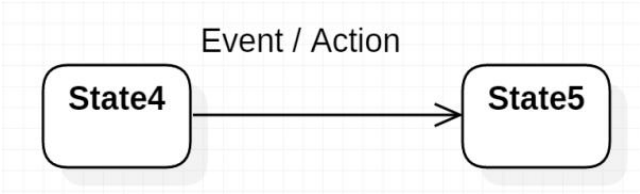


Gambar 25. Rangkaian *State*

Ada beberapa syarat dalam penggunaan transisi, yaitu:

- Elemen sebagai sumber *State*
- Terjadinya *event*
- Ada aksi dilakukan
- Elemen memasuki target *State*

Jika sebuah *event* menyebabkan transisi ditulis pada panah dan dipisahkan dengan tanda slash '/'.



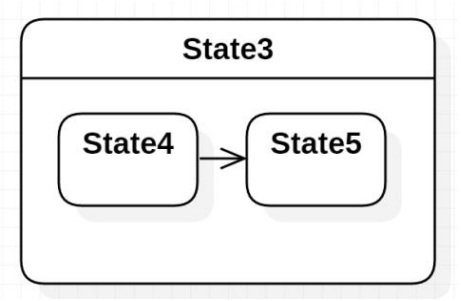
Gambar 26. Transisi

Self-Transition merupakan sebuah transisi dengan *State* sumber dan target yang sama.

Gambar 27. *Self Transition*

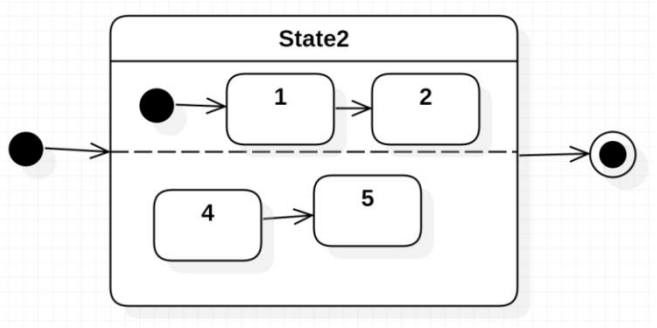
Di dalam *state machine diagram* (*state*chart versi 1.x) ada *composite state*. *Composite state* merupakan sebuah region yang memiliki satu atau lebih *statechart diagram* didalamnya. *Composite state* dapat dibuat dari:

- Terdapat beberapa *state*chart dalam sebuah *state*
- *State* yang berbeda terbagi dalam region dan dapat aktif dalam waktu bersamaan



Gambar 28. *Composite State*

Sedangkan *Orthogonal State* adalah salah satu jenis *Composite State* yang memiliki lebih dari 1 region. Region merupakan kelompok *State* yang digambarkan garis putus-putus untuk memisahkan *State* satu dengan *State* lainnya.



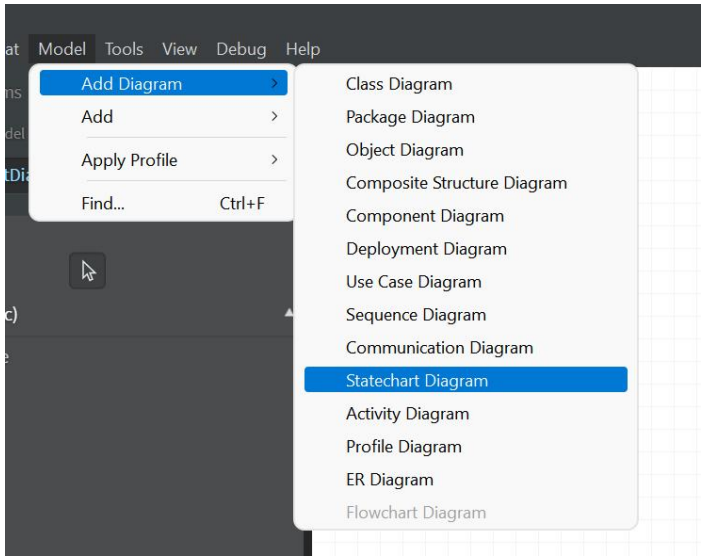
Gambar 29. *Orthogonal State*

Substate merupakan *state* generalisasi berasal dari super-*state* (*composite state*) yang terdiri dua atau lebih *state* level rendah. Istilah yang tepat adalah dekomposisi. Dekomposisi *state* dapat dilakukan hingga beberapa level sesuai kebutuhan pembagiannya. Karena masih dalam bagian *state*chart, maka notasinya sama dengan *state*chart.

2. Pemodelan *Activity Diagram*

a) Aplikasi Pemodelan

- Buka aplikasi StarUML.
- Klik Model->Add Diagram-> *StateChart Diagram*.

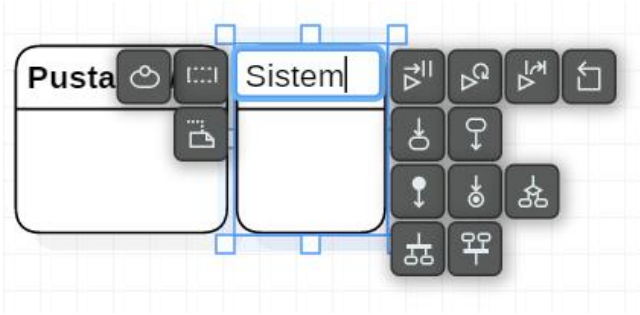


Gambar 30. Tambah *StateChart Diagram*

- Bukalah tab *state*chart (advanced) dan klik *composite state*. Kita akan membuat *composite state* pustakawan dan sistem

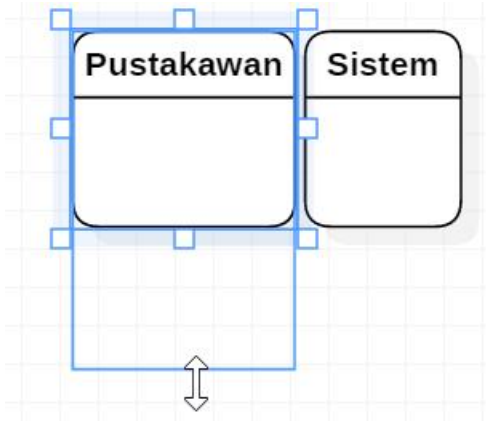
Gambar 31. Menu *Composite State*

- Klik kiri pada area kerja



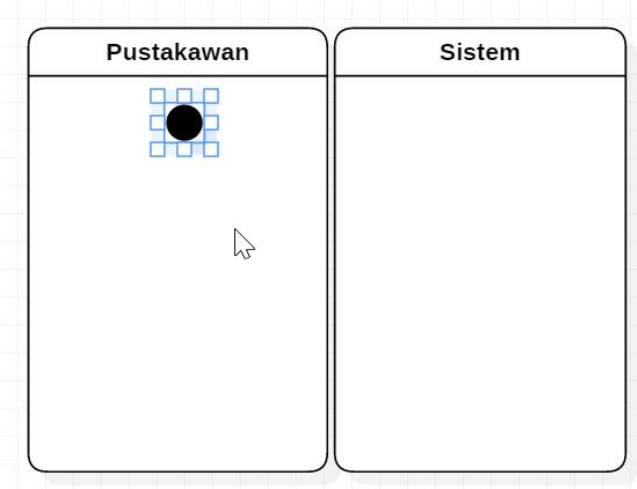
Gambar 32. Nama *Composite State*

- Anda dapat memperbesar ukuran kotaknya dengan mengaktifkan *Composite State*nya terlebih dahulu. Kemudian klik dan drag kotak putih kecil.



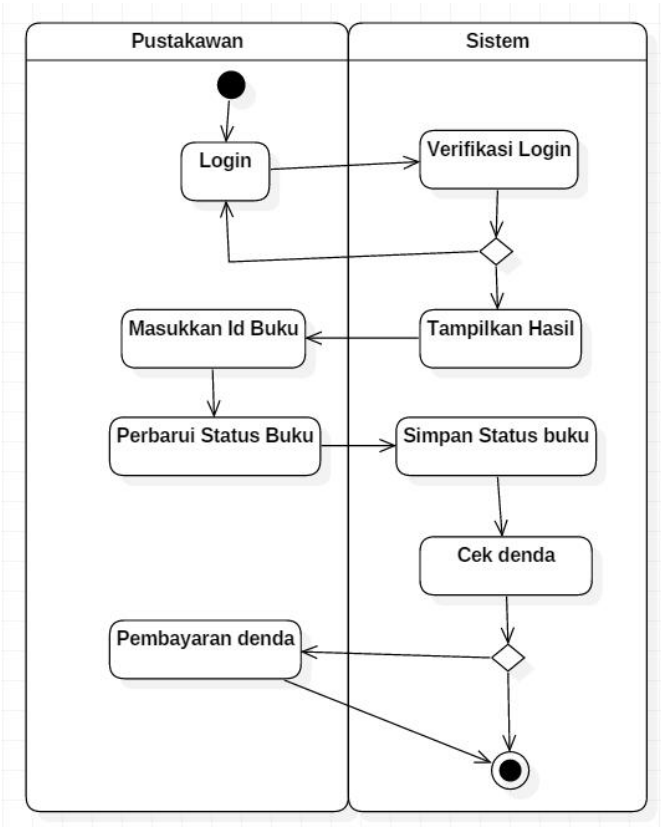
Gambar 33. Ukuran *Composite State*

- Mulailah dengan membuat *Initial Node* terlebih dahulu



Gambar 34. *Initial State*

- Buatlah *State*chart seperti gambar di bawah ini. Aturlah posisi setiap *State* agar tetap rapi.



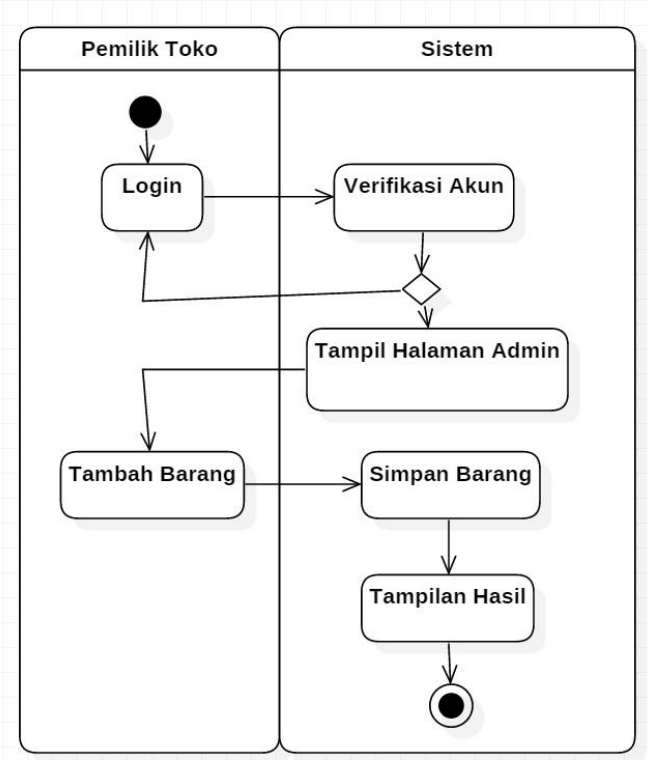
Gambar 35. Pengembalian Buku

b) Studi Kasus 1

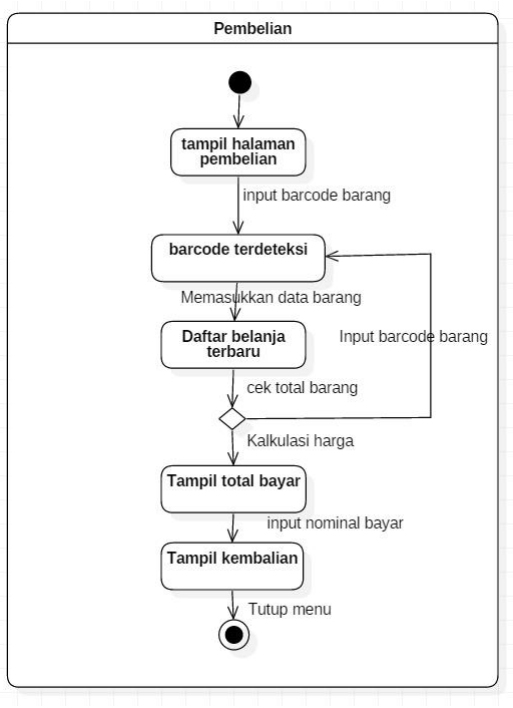
Sistem informasi penjualan

Pemilik toko sebagai admin dari sistem informasi. Sehingga pemilik toko memiliki akses untuk menambahkan data barang. Pemilik toko

harus login terlebih dahulu untuk menggunakan sistem. Kemudian masuk ke halaman admin dan memilih menu tambah barang. Setelah memasukkan data barang kemudian data tersebut disimpan oleh sistem. *State* terakhir adalah menampilkan pesan bahwa penyimpanan berhasil.



Gambar 36. *State* Tambah Barang



Gambar 37 Pembelian Barang

Model di atas menggambarkan proses bagaimana sistem melakukan pencatatan aktivitas pembelian. Penjelasannya adalah sebagai berikut:

- State pertama adalah sistem menampilkan halaman pembelian.
- Kemudian kasir memasukkan data barcode. Sistem akan membaca barcode barang
- Barang akan dimasukkan ke dalam daftar belanja

- Jika masih ada barang yang belum dimasukkan, maka kembali input data barang. Maka proses mengulang kembali ke state barcode terdeteksi
- Jika sudah tidak ada, maka sistem akan melakukan kalkulasi total bayar terhadap semua barang yang ada
- Kemudian, nominal bayar dimasukkan sesuai uang yang diberikan oleh pembeli.
- Lalu sistem akan menghitung uang kembalian dan menampilkannya ke layar
- Setelah pembayaran selesai, maka menu pembelian tertutup

c) Studi Kasus 2

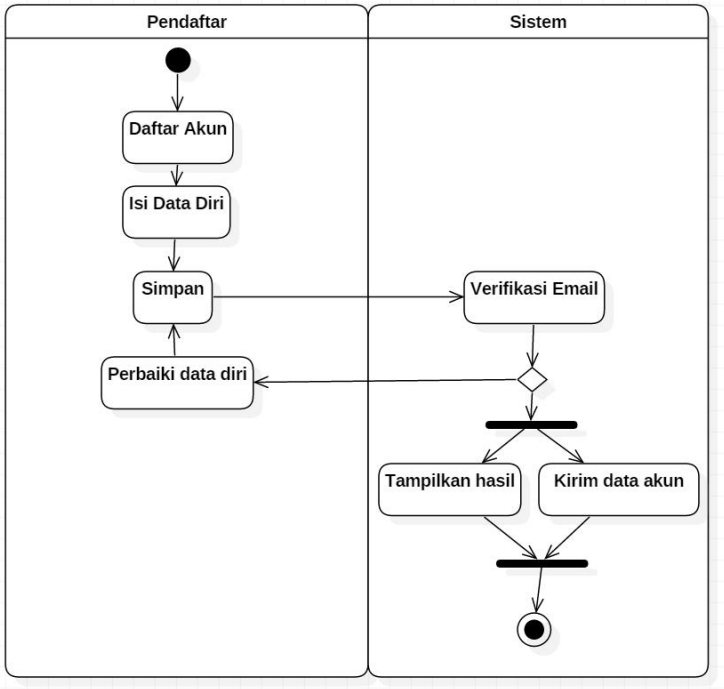
Sistem informasi penerimaan mahasiswa baru

Pada sistem informasi penerimaan mahasiswa baru, Pendaftar dapat melakukan pendaftaran akun baru. Langkah-langkahnya adalah sebagai berikut;

- Pendaftar memilih menu Daftar Akun
- Pendaftar dapat mengisi data diri dengan benar dan menekan tombol Simpan
- Sistem akan mengecek email yang didaftarkan. Apakah email tersebut telah didaftarkan atau belum.

- Jika email telah terdaftar sebelumnya, Pendaftar harus memperbaiki data dirinya.
- Jika belum terdaftar, maka terdapat *Fork Node*. *State* tampilkan hasil penyimpan melalu tampilan antarmuka dan mengirim email pemberitahuan ke email Pendaftar

Untuk desain model pendaftaran akun baru dapat dilihat pada gambar 35.



Gambar 38. *State* Daftar Akun

d) Latihan

- Jelaskan apa yang dimaksud dengan *event*!
- Apa yang dimaksud dengan Entry Action dan Exit Action pada Transisi? Berikan contoh modelnya
- Jelaskan tentang Send *event*! Berikan contoh notasinya!

e) Tugas Mandiri

Buatlah diagram *State* dari pendaftaran proposal Proyek Akhir sesuai dengan proses bisnis di kampus anda.

Daftar Pustaka

Andriyanto, S., & Mulyani, L. N. (2020). Analisa Dan Perancangan Perangkat Lunak: Digital Library. Politeknik Manufaktur Negeri Bangka Belitung.

Kusnadi, I. T., Supiandi, A. S., Syabaniah, R. N., & Oktapiani, R. (2019). *Pemodelan Sistem Berbasis Objek with UML* (1st ed.). Graha Ilmu.

Nugroho, Eko, (2008), Sistem Informasi Manajemen: Konsep, Aplikasi, dan Perkembangannya. Andi.

Glosarry

Diagram	Gambaran (buram, sketsa) untuk memperlihatkan atau menerangkan sesuatu
Event	Kejadian
State	Sebuah kondisi atau sebuah situasi khusus dari suatu elemen selama siklus hidup
Tool	Perangkat lunak pendukung

Biografi Penulis

Penulis merupakan salah seorang dosen tetap di Jurusan Teknik Elektronika dan Informatika, Politeknik Manufaktur Negeri Bangka Belitung.

Penulis lulusan Sarjana Teknik Informatika di Universitas Ahmad Dahlan Yogyakarta dan Magister Teknik Informatika Universitas AMIKOM Yogyakarta.

Bidang minat penulis adalah rekayasa perangkat lunak, desain grafis, game design, dan multimedia.

Buku ini mengulas tentang diagram Unified Modeling Language dengan fokus pada behavior diagram. Dalam kategori ini terdapat tiga diagram yakni Use Case Diagram, Activity, dan State Machine Diagram. Buku ini berisi tentang teori dasar, langkah penggunaan aplikasi pendukung, penyelesaian studi kasus, latihan teori dan tugas mandiri.

Buku ini dapat dibaca oleh siswa, mahasiswa atau umum yang bertujuan untuk memahami bagaimana membuat model dari perangkat lunak. Model tersebut menggunakan UML dengan aplikasi pendukung StarUML. Setiap tahapan perlu dibaca dengan seksama, sehingga pembaca dapat menerapkan pada kasus yang dimilikinya.

POLMANBABEL PRESS

Penerbit Politeknik Manufaktur Negeri Bangka Belitung (POLMANBABEL PRESS)
Kawasan Industri Air Kantung, Sungailiat,
Bangka 33211
Telp: 0717 93586
E-Mail: polman@polman-babel.ac.id

ISBN 978-623-97870-8-0
9 786239 787080