RANCANG BANGUN APLIKASI UNTUK KONVERSI *BUSINESS PROCESS MODEL NOTATION* KE MODEL *UNIFIED MODELING LANGUAGE* BERBASIS *WEBSITE*

TUGAS AKHIR

Program Studi

S1 Sistem Informasi

Oleh:

Anak Agung Angga Wijaya

16410100123

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2020

RANCANG BANGUN APLIKASI UNTUK KONVERSI *BUSINESS PROCESS MODEL NOTATION* KE MODEL *UNIFIED MODELING LANGUAGE* BERBASIS *WEBSITE*

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk menyelesaikan
Program Sarjana Komputer

Oleh :

Nama : Anak Agung Angga Wijaya

NIM : 16410100123

Program Studi : S1 Sistem Informasi

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2020

Tugas Akhir

RANCANG BANGUN APLIKASI UNTUK KONVERSI *BUSINESS PROCESS MODEL NOTATION* KE MODEL *UNIFIED MODELING LANGUAGE* BERBASIS *WEBSITE*

Dipersiapkan dan disusun oleh

Anak Agung Angga Wijaya

NIM: 16410100123

Telah diperiksa, diuji dan disetujui oleh Dewan Pembahas

Pada: Kamis, 06 Februari 2020

Susunan Dewan Pembahas

Pembimbing:

I. Dr. Anjik Sukmaaji, S.Kom., M.Eng.
NIDN: 0731057301

II. Teguh Sutanto,M.kom.
NIDN. 0713027801

Pembahas:

Tutut Wurijanto, M.kom.
NIDN: 0703056702

Tugas Akhir ini telah diterima sebagai salah satu persyaratan
Untuk memperoleh gelar Sarjana

Fakultas Teknologi dan Informatika
UNIVERSITAS Dinamika

Dr. Jusak
NIDN: 0708017101

Dekan Fakultas Teknologi dan Informatika
UNIVERSITAS DINAMIKA

PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa Universitas Dinamika, saya :

Nama	: Anak Agung Angga Wijaya
Nim	: 16410100123
Program Studi	: S1 Sistem Informasi
Fakultas	: Fakultas Teknologi dan Informatika
Jenis Karya	: Tugas Akhir
Judul Karya	:**RANCANG BANGUN APLIKASI UNTUK KONVERSI *BUSINESS PROCESS MODEL NOTATION* KE MODEL *UNIFIED MODELING LANGUAGE* BERBASIS *WEBSITE***

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, saya menyetujui memberikan kepada Universitas Dinamika Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalti Free Right*) atas seluruh isi/ sebagian karya ilmiah saya tersebut di atas untuk disimpan, dialihmediakan dan dikelola dalam bentuk pangkalan data (*database*) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta.
2. Karya tersebut di atas adalah karya asli saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya atau pendapat orang lain yang ada dalam karya ilmiah ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiat pada karya ilmiah ini, maka saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, 15 Februari 2020

menyatakan

Anak Agung Angga Wijaya

NIM: 16410100123

“ a luta continua a vitória é certa”

Ku persembahkan kepada

Keluargaku yang ku sayangi,

Beserta semua teman dan sahabat yang selalu

Mendukungku.

ABSTRAK

Dalam konsep *Model Driven Architecture* (MDA), terdapat *Computation Independent Model* (CIM) atau dikenal *Business Process Model Notation* (BPMN) dan *Platform Independent Model* (PIM) atau dikenal dengan Model UML (*Unified Modeling Language*). Saat ini tidak ada sistem yang dapat mengotomatasisasi BPMN kedalam simbol pemrograman, yakni UML. Metode konversi tersebut menurut Rhazali dapat dilakukan dengan mengikuti 8 aturan.Proses Konversi dari BPMN ke UML menggunakan data dari *Business Process Modeling* (BPM) dengan format *Bizagi Modeler* sebagai masukan dan akan menghasilkan file DrawIO. DrawIO ini adalah suatu aplikasi *standalone* yang dapat menampilkan notasi dari XML yang sudah dibuat. Sistem yang dibuat dapat menghasilkan UML dari data BPM tetapi file BPM yang dibuat harus sudah mengikuti aturan yang telah dibuat oleh Rhazali. Keluaran sistem ini adalah *file* DrawIO yang memiliki dua diagram yaitu *use case* dan *class diagram*. DrawIO hasil konversi BPMN ke UML tersebut berdasarkan 8 aturan Rhazali dengan menambahkan : (1) Penamaan pada *component* non-task, (2) pengerjaan BPMN haru berurutan, (3) penamaan sub diagram . Sistem ini dapat dikembangkan lebih lanjut dengan menambahkan modul aplikasi yang dapat dibaca menggunakan StarUML, agar kode UML dapat langsung di*generate* menjadi kode program bahasa pemrograman.

Kata Kunci: *Website*, BPMN, UML,DrawIO,Konversi,MDA

KATA PENGANTAR

Puji syukur ke pada Tuhan Yang Maha Esa atas segala nikmat yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul "Rancang Bangun Aplikasi Untuk Konversi *Business Process Model Notation* ke Model *Unified Modeling Language* Berbasis Website".

Penyelesaian tugas akhir ini tidak terlepas dari bantuan berbagai pihak yang telah memberikan banyak masukan, nasihat, saran, kritik dan dukungan moral kepada penulis. Oleh karena itu penulis menyampaikan rasa terima kasih kepada:

1. Ayah dan Ibuku tercinta serta keluarga besarku yang selalu mendoakan, mendukung, dan memberikan semangat di setiap langkah dan aktifitas penulis.
2. Bapak Prof. Dr. Budi Jatmiko, M.Pd. selaku Rektor Universitas Dinamika yang telah mengesahkan dan memberikan kesempatan secara resmi dalam melakukan tugas akhir.
3. Bapak Dr. Anjik Sukmaaji, S.Kom., M.Eng selaku Ketua Program Studi Sistem Informasi Universitas Dinamika dan Dosen pembimbing yang telah memberikan motivasi dan arahan untuk menyelesaikan tugas akhir ini .
4. Ibu Norma Ningsih, S.ST., M.T. selaku Dosen Wali yang telah mendukung, membimbing selama penulis menempuh kuliah.
5. Bapak Teguh Sutanto, M.Kom selaku Dosen pembimbing yang telah memberikan motivasi dan arahan untuk menyelesaikan tugas akhir ini .
6. Bapak Sholiq ST., M.Kom selaku dosen yang telah memberikan bimbingan dan arah terkait detail teknikal dalam pembuatan tugas akhir ini .
7. Bapak Tutut Wurijanto, M.Kom. selaku dosen pembahas yang telah memberikan saran dan juga dukungan dalam pelaksanaan tugas akhir.

8. Semua teman teman saya tergabung dalam grup “Cangkruk Lapangan Tembak” dan “Sobat Ambyar” yang telah memberikan dukungan dan motivasi.
9. Pihak-pihak lain yang tidak dapat disebutkan satu-persatu yang telah memberikan bantuan dan dukungan kepada penulis.

Semoga Tuhan Yang Maha Esa memberikan balasan yang setimpal kepada semua pihak yang telah memberikan bantuan, bimbingan, dan nasehat dalam proses tugas akhir ini.

Penulis menyadari bahwa Tugas Akhir yang dikerjakan masih banyak terdapat kekurangan, sehingga kritik yang bersifat membangun dan saran dari semua pihak sangatlah diharapkan agar aplikasi ini dapat diperbaiki menjadi lebih baik lagi dikemudian hari. Semoga Tugas Akhir ini dapat diterima dan bermanfaat bagi penulis dan semua pihak.

Surabaya, 15 Februari 2020

Penulis

DAFTAR ISI

ABSTRAK vi

KATA PENGANTAR vii

DAFTAR ISI ix

DAFTAR GAMBAR xi

DAFTAR TABEL xii

DAFTAR LAMPIRAN xiii

BAB I PENDAHULUAN 1

1.1. Latar Belakang Masalah 1

1.2. Perumusan Masalah 4

1.3. Pembatasan Masalah 4

1.4. Tujuan Penelitian 4

1.5. Manfaat Penelitian 5

BAB II LANDASAN TEORI 6

2.1. Extensible Markup Language(XML) 6

2.2. Unified Modeling Language 7

2.3. *Model Driven Architecture* 8

2.4. *Business Process Modeling Notation* (BPMN) 9

2.5. Transformation Method CIM to PIM:From Business Process Models Defined in BPMN to Use Case and Class Models Defined in UML 10

2.6. Agile Software Development Phase 12

BAB III METODOLOGI PENELITIAN 16

3.1. Identifikasi Masalah 16

3.2. Studi Literatur 16

3.3. Model Konversi BPMN ke UML 17

BAB IV HASIL DAN PEMBAHASAN 26

4.1. Hasil 26

4.1.1. Contoh *Sample Recruitment* 26

4.1.2. Contoh *Sample E-Commerce* 32

4.1.3. Software Testing 38
4.1.4. Tampilan Aplikasi Konversi BPMN ke UML 38
4.2. Pembahasan 41
BAB V PENUTUP 42
5.1. Kesimpulan 42
5.2. Saran 42
DAFTAR PUSTAKA 43
LAMPIRAN 48
DAFTAR RIWAYAT HIDUP 176

DAFTAR GAMBAR

Gambar 2.1 Struktur XML 6
Gambar 2.2 Urutan pembuatan UML 8
Gambar 2.3 Simbol *Event* 9
Gambar 2.4 Simbol *Gateway* BPMN 10
Gambar 2.5 Simbol *Swimlanes*/*Pool* 10
Gambar 2.6 Model Pengembangan Agile 12
Gambar 3.1 Konsep Konversi 17
Gambar 3.2 Use case Aplikasi Konversi 18
Gambar 3.3 Class Diagram Aplikasi Konversi 22
Gambar 3.4 Penambahan nama pada *component* non *task* 23
Gambar 3.5 Penamaan Diagram *Sub process* 24
Gambar 4.1 *Collaboration Level Sample* 26
Gambar 4.2 *Process level* diagram "*job vacancy advertisement*" 27
Gambar 4.3 Hasil *Usecase* diagram *Recruitment* 28
Gambar 4.4 Hasil *Class* Diagram *Recruitment* 28
Gambar 4.5 BPM Fokus Relasi *Gateway* 30
Gambar 4.6 BPM Fokus Relasi *Sequence* 30
Gambar 4.7 *Collaboration Level Sample E-Commerce* 32
Gambar 4.8 *Process level* diagram "*select product of order*" 33
Gambar 4.9 Hasil *Usecase* diagram *E-Commerce* 34
Gambar 4.10 Hasil *Usecase* diagram *E-Commerce* 34
Gambar 4.11 BPM Fokus Relasi *Gateway Sample E-Commerece* 36
Gambar 4.12 BPM Fokus Relasi *Sequence Sample E-commerce* 36
Gambar 4.13 Halaman UI *Login* 38
Gambar 4.14 Halaman UI *Register* 39
Gambar 4.15 Halaman UI *Upload File* 39
Gambar 4.16 Halaman UI *History* 40
Gambar 4.17 Tampilan "*View*" pada Halaman *History* 41

DAFTAR TABEL

Tabel 4.1 *Output* Unit *Testing* Iterasi 1 38

DAFTAR LAMPIRAN

Lampiran 1 : *Agile Development* 48
Lampiran 2 : *Sample* Konversi 98
Lampiran 3 : *Array Collecting Usecase Sample* 1 135
Lampiran 4 : *Array Collecting Task Sample* 1 136
Lampiran 5 : *Collecting Dataobject Sample* 1 137
Lampiran 6 : *Collecting TaskRelation Sample* 1 138
Lampiran 7 : *Validation Usecase Sample* 1 143
Lampiran 8 : *Validation Class Sample* 1 144
Lampiran 9 : *Generate Header Tag DrawIO Sample* 1 145
Lampiran 10 : *Generate Class Diagram belum di encode Sample* 1 145
Lampiran 11 : *Generate Class Diagram sudah diencode Sample* 1 146
Lampiran 12 : *Generate Usecase diagram belum diencode Sample* 1 146
Lampiran 13 : *Generate Usecase diagram sudah diencode sample* 1 147
Lampiran 14 : *Final Generate DrawIO* 148
Lampiran 15 : *Array Collecting Usecase Sample 2* 149
Lampiran 16 : *Array Collecting Task Sample 2* 149
Lampiran 17 : *Array Collecting Dataobject Sample* 2 150
Lampiran 18 : *Array Collecting TaskRelation Sample* 2 151
Lampiran 19 : *Validation Usecase Sample* 2 153
Lampiran 20 : *Validation Class Sample* 2 154
Lampiran 21 : *Generate Header Tag DrawIO Sample* 2 155
Lampiran 22 : *Generate Class Diagram belum di encode Sample* 2 155
Lampiran 23 : *Generate Class Diagram sudah diencode Sample* 2 156
Lampiran 24 : *Generate Usecase diagram belum diencode Sample* 2 156
Lampiran 25 *: Generate Usecase diagram* sudah *diencode Sample* 2 159
Lampiran 26 : *Final Generate DrawIO* 159
Lampiran 27 : *Raw Data Activity Diagram "Select product of order"* 160
Lampiran 28 : *Raw Data "Activity" Diagram "Main Diagram"* 161
Lampiran 29 : *Raw "TaskRelation" Data "select product of order"* 162
Lampiran 30 : *Raw DataObject select product of order* 171
Lampiran 31 : Tahapan BPMN ke XML 172

Lampiran 32 : Contoh *Sample Paper Rhazali*..... 174

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Business Process Modeling Notation (BPMN) merupakan suatu cara menggambarkan suatu bisnis proses diagram yang mana didasarkan kepada teknik diagram alur, dirangkai untuk membuat model-model grafis dari operasi-operasi bisnis dimana terdapat aktivitas-aktivitas dan kontrol-kontrol alur yang mendefinisikan urutan kerja. (Ramdhani, 2015).

BPMN ini diperlukan dalam pembuatan suatu aplikasi. Karena program tersebut diharapkan bisa mengefisiensikan suatu proses bisnis. Pemodelan proses bisnis dapat digunakan untuk mengidentifikasi kebutuhan sistem sebagai dasar pengembangan sistem informasi (Sari & Tj, 2015). Simulasi proses bisnis dilakukan dalam beberapa tahapan sehingga dapat mengetahui waktu pengerjaan dan permasalahan apa saja yang dihadapi (Yaqin, Adawiyah, Ayu Ningrum, & Janan, 2019). Oleh karena itu proses bisnis tidak hanya dijadikan suatu standar operasional tetapi juga bisa mengetahui permasalahan yang akan terjadi.

BPMN sudah digunakan banyak orang untuk menggambarkan proses bisnis yang terjadi. BPMN menjadi standar dan teknologi yang dipilih di antara pendekatan lain bagi sebagian besar pakar bisnis untuk memodelkan proses, karena ia menawarkan keunggulan bahasa grafis, serta kesederhanaan, standardisasi (ISO / IEC 1951: 2013) dan dukungan untuk proses eksekusi (Arevalo, Escalona, Ramos, & Domínguez-Muñoz , 2016). Karena sudah menjadi

standar maka pengunaan BPMN ini sudah dipastikan menyentuh berbagai perusahaan yang ada di dunia.

Tetapi jika hanya menggunakan BPMN sebagai dasar dalam pengembangan aplikasi tidak dapat digunakan untuk implementasi ke dalam pengembangan aplikasi. Berdasarkan *Model Driven Architecture* (MDA), *Computation Independent Model* (CIM) adalah model yang digunakan oleh manajer bisnis dan analis bisnis untuk menggambarkan proses bisnis (Rhazali, Mouloudi, & Hadi, 2014). CIM tidak menunjukan detail struktur sistem melainkan menjembatani kedua orang ahli (Betari, Filali, Azzaoui, & Amine Boubnad, 2018). Untuk pendekatan dalam pengembangan aplikasi dan memastikan proses bisnis tetap terjaga dan masih utuh maka perlu dilakukan transformasi ke *Platform Indepedent Model* (PIM). Dalam sehari-hari *Platform Indepedent Model* (PIM) ini bisa juga di sebut Model (*Unified Modeling Language*) UML. (Rhazali, Mouloudi, & Hadi, 2014)

UML sudah digunakan diberbagai perusahaan khususnya perusahaan yang bergerak di bidang *software development*. Hampir semua perusahaan menggunakan UML ini sebagai rancangan dasar dalam pembuatan aplikasi. UML telah menjadi standar untuk pengembangan sistem berorientasi objek dan akan segera menjadi standar internasional. UML menjanjikan untuk meringankan beberapa masalah yang terkait dengan pengembangan perangkat lunak berorientasi objek (Grossman, Aronson, & McCarthy, 2005). Karena sudah menjadi standar maka pengunaan UML ini sudah dipastikan menyentuh berbagai perusahaan yang ada di dunia.

Model UML memiliki banyak jenis diagram. UML mempunyai berbagai diagram antara lain : *Diagram use case,* diagram aktivitas, diagram sekuensial,

diagram kelas, dan lain sebagainya.Berbagai diagram tersebut memberikan pengertian yang berbeda terhadap sistem (Sholiq & Robandi, Analisis dan Perancangan Berorientasi Obyek, 2010). UML bisa dibilang sebagai *blueprint* karena dengan UML bisa diketahui informasi secara detail tentang koding program atau membaca cara kerja program (Wati & Kusumo, 2016).

Namun yang sering terjadi dalam proses transformasi dari CIM ke PIM masih dilakukan manual dan dikhawatirkan tidak mematuhi kaidah dalam MDA(*Model Driven Architecture*). Ataupun dalam prosesnya tidak mengambarkan secara utuh apa yang terjadi di CIM. Proses konversi dari CIM ke PIM sangat bergantung terhadap perancang sehingga kualitas dari PIM itu sendiri tidak terkontrol (Wei, Mei, Zhao, & Jie Yang, 2005). Oleh karena itu penulis ingin tetap memastikan CIM tetap produktif, memastikan dasar dari pembuatan PIM adalah CIM dan diharapkan bisa memangkas waktu dalam konversi dari proses bisnis menjadi dasar pengembangan aplikasi.

Pada *paper* "*Transformation Method CIM to PIM : From Business Processes Models Defined in BPMN to Use Case and Class Models Defined in UML*" telah membuat prosedur-prosedur terkait otomatisasi konversi dari BPMN ke UML. Didalam *paper* tersebut dijelaskan aturan-aturan yang mendukung konversi dari BPMN ke UML secara otomatis. Aturan-aturan yang dibuat adalah Aturan membuat CIM pada *level collaboration*, Aturan membuat CIM pada *level process*, Aturan mengubah dari CIM *level collaboration* menjadi *use case*, dan Aturan mengubah dari CIM *level process* menjadi *Class Diagram*.

Hasil dari aplikasi ini adalah dapat mengeluarkan UML dengan memanfaatkan DrawIO. DrawIO ini adalah platform independen yang berfungsi untuk menvisualisasikan XML.UML yang dibangun didasarkan dengan Aturan-aturan (Rhazali, Mouloudi, & Hadi, 2014) dan tambahan aturan dari penulis. Sehingga keluaran dari aplikasi ini akan terstandar.

1.2. Perumusan Masalah

Berdasarkan latar belakang diatas, maka dapat dirumuskan permasalahan yang ada yaitu :

Bagaimana merancang bangun sistem konversi BPMN ke UML menggunakan *Model Driven Architecture* berbasis website ?

1.3. Pembatasan Masalah

Berdasarkan rumusan masalah maka dibuatlah batasan masalah sebagai fokus dari penelitian. Batasan masalah sebagai berikut:

1. Aplikasi ini berbasis website dan menggunakan Laravel.
2. Aplikasi ini menggunakan aturan-aturan yang telah dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014).
3. Aplikasi ini hanya menerima file dari bizagi modeler.
4. Aplikasi ini akan mengeluarkan keluaran yang mengikuti format dari Draw.IO(XML).
5. Aplikasi ini hanya mengeluarkan use case diagram dan class diagram(tanpa method, property dan relasi).

1.4. Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah, maka tujuan dari tugas ini adalah menghasilkan aplikasi rancang bangun aplikasi untuk konversi business

process model notation ke model *Unified Modeling Notation* berbasis website yang telah lolos uji dengan aturan yang dibuat (Rhazali, Mouloudi, & Hadi, 2014).

1.5. Manfaat Penelitian

Manfaat Pengguna diantaranya:

1. Dapat mengefisiensikan waktu yang dibutuhkan untuk menkonversi dari BPMN ke UML
2. Dapat mengefisiensikan *cost* yang dikeluarkan akibat lamanya proyek

Manfaat peneliti diantaranya :

1. Dapat memahami XML yang di suatu diagram.
2. Dapat dikembangkan sehingga dapat mengeluarkan tipe diagram UML yang lain.

BAB II

LANDASAN TEORI

2.1. Extensible Markup Language(XML)

XML adalah bahasa *markup* yang digunakan untuk memberi label, mengkategorikan, dan mengatur informasi dengan cara tertentu. Markup dapat menggambarkan dokumen atau struktur data dan organisasi. Konten, seperti teks, gambar, dan data , adalah bagian dari kode yang berisi tag markup.XML tidak terbatas pada satu set markup tertentu. Seseorang dapat membuat markup sendiri sesuai dengan data dan kebutuhan dokumen anda. Fleksibilitas XML telah menyebabkan pengunaannya yang luas dalam pertukaran data yang dikemas dalam banyak bentuk. (Dykes & Tittel, 2005).

XML digunakan untuk mendeskripsikan susunan informasi dan berfokus pada informasi itu sendiri. XML terutama dibutuhkan untuk menyusun dan menyajikan informasi dengan format yang tidak mengandung format standar layaknya *heading*, *paragraph*, *table* dan lain sebagainya. (Junaedi, 2003).

Gambar 2.1 Struktur XML
(Sumber : TutorialsPoint)

Document prolog adalah tempat dimana kita memberikan informasi yang diperlukan untuk memberitahu XML *processor* untuk mem*parsing* XML. Sedangkan *Document Elements* berisi data yang akan disimpan.XML dengan model blok bangunan dan membagi sesuai dengan hierarki bagian.

2.2. Unified Modeling Language

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. (Dharwiyanti & Wahono, 2003).

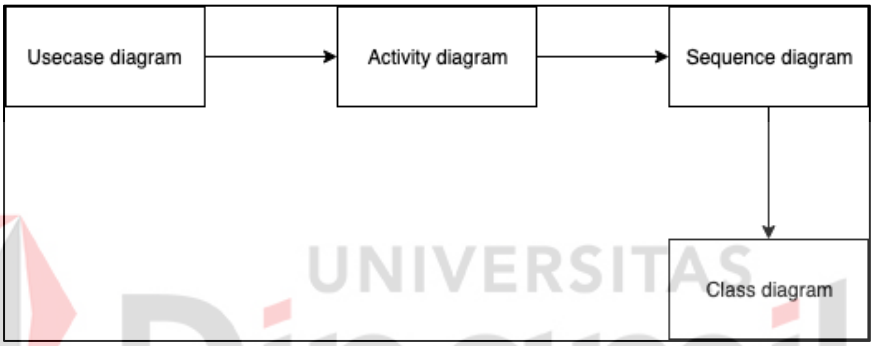
UML dianggap sebagai bahasa pemodelan standar industri dengan notasi grafis yang kaya, dan set diagram dan elemen yang komprehensif. Ini digunakan untuk menentukan, memvisualisasikan, memodifikasi, membangun dan mendokumentasikan artefak dari sistem intensif perangkat lunak berorientasi objek di bawah pengembangan (Lee, 2012).

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam VB atau C . (Sugiarti, 2013).

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk

menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). (Dharwiyanti & Wahono, 2003).

Didalam Unified Modeling Language terdapat beberapa diagram yang saling membutuhkan seperti gambar 2.2



Gambar 2.2 Urutan pembuatan UML

2.3. *Model Driven Architecture*

Strategi pengembangan perangkat lunak berbasis MDA berasal persyaratan dari organisasi bisnis. Dalam strategi ini, gunakan dan pertukaran informasi didorong oleh kebutuhan bisnis yang lebih banyak daripada solusi perangkat lunak. MDA jelas menyajikan bisnis pendekatan termotivasi untuk pengembangan sistem perangkat lunak. Ini pendekatan pengembangan perangkat lunak dimulai dengan model pertama MDA yaitu *Computational Independent Model* (CIM) yang menggambarkan suasana bisnis dan persyaratan bisnis. CIM adalah kemudian ditransformasikan ke model selanjutnya bernama *Platform Independent Model* (PIM) yang secara eksplisit menjelaskan layanan dan antarmuka yang disediakan

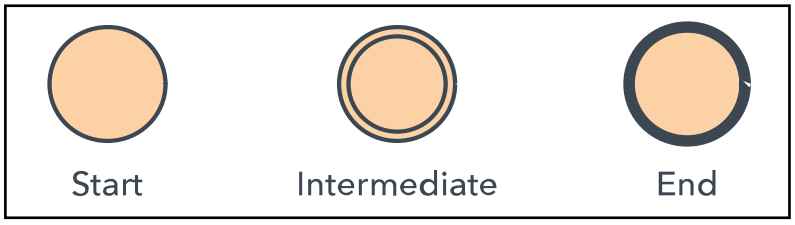
oleh sistem perangkat lunak tanpa mempertimbangkan *platform* teknologi. PIM selanjutnya diubah menjadi *Platform Specific Model* (PSM) untuk realisasi sistem perangkat lunak untuk teknologi spesifik. (Singh & Sood, 2010).

2.4. *Business Process Modeling Notation* (BPMN)

Business Process Modeling Notation (BPMN) menggambarkan suatu bisnis proses diagram yang mana didasarkan kepada teknik diagram alur, dirangkai untuk membuat model-model grafis dari operasi-operasi bisnis dimana terdapat aktivitas-aktivitas dan kontrol-kontrol alur yang mendefinisikan urutan kerja. (Ramdhani, 2015).

Tujuan utama BPMN adalah untuk memberi semua pengguna bisnis notasi yang mudah dimengerti, dimulai dengan analis bisnis yang menciptakan konsep awal proses, pengembang teknologi yang bertanggung jawab untuk menerapkan proses yang ada, dan proses pengelolaan dan pemantauannya. Oleh karena itu BPMN berfungsi sebagai jembatan antara perancangan proses bisnis dan implementasi proses bisnis. (Helmi, Aknuranda, & Saputra, 2018).

BPMN memiliki berbagai macam notasi yang menggambarkan sesuatu hal. Notasi yang ada didalam BPMN adalah *Event, Activity, Gateway, Swimlanes/Pool, Connecting Object.* Notasi diatas dibuat agar pengguna dapat memahami dapat lebih mudah memahami proses bisnis yang terjadi.

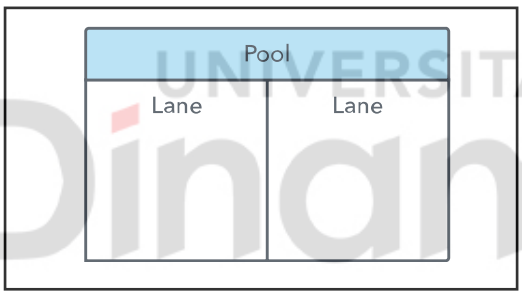


Gambar 2.3 Simbol *Event*
(Sumber : LucidChart)

Event biasanya berbentuk bulat dan mereka mewakili sesuatu. *Activity* dibpmn digambarkan dengan simbol persegi Panjang simbol bulat dan memiliki arti sesuatu yang dikerjakan.

Gateway berbentuk berlian dan diujungnya menghubungkan lebih dari satu alur, *gateway* sendiri berfungsi untuk memisahkan satu alur menjadi dua alur yang berbeda.

Gambar 2.4 Simbol *Gateway* BPMN
(Sumber : LucidChart)



Gambar 2.5 Simbol *Swimlanes*/*Pool*
(Sumber : LucidChart)

Swinlanes/Pool digambarkan dengan kotak yang lebar dan berfungsi untuk mengelompokan aspek yang berbeda.

Connecting Object digambarkan dengan suatu garis dan memiliki fungsi untuk menghubungkan semua object yang ada. (Lucid Software Inc., 2019).

2.5. Transformation Method CIM to PIM:From Business Process Models Defined in BPMN to Use Case and Class Models Defined in UML

Dalam penelitian yang dilakukan oleh penulis mengacu pada *paper* ini, untuk dijadikan dasar dalam melakukan konversi dari BPMN ke UML.*Paper* ini dibuat

oleh (Rhazali, Mouloudi, & Hadi, 2014). Didalam *paper* ini memuat beberapa aturan yang akan diimplementasi kan ke dalam aplikasi diantaranya adalah :

1. Aturan dari CIM ke Use Case

a. Setiap tugas dalam proses level bpm akan dijadikan menjadi *use case*

b. Pada kolaborasi level bpm , tugas yang berada didalam suatu *pool* maka *pool* tersebut akan menjadi actor *use case*

c. Jika terdapat gerbang ekskulif diantara dua tugas maka akan menjadi relasi "extend" pada *use case*

d. Jika terdapat *task* yang berurutan maka akan menjadi relasi "include" pada *use case*

e. Dalam menerjemahkan harus berurutan dan tidak boleh berjalan mundur

f. Setiap task yang ada di kolaborasi level bpm akan diterjemahkan *package*

2. Aturan dari Process Level ke Class

a. Dalam proses level, *dataobject* akan diubah menjadi *classes*

b. Jika ada *dataobject* yang memiliki nama yang sama maka hanya ditransformasi kan menjadi satu *class* saja.

Tetapi untuk memastikan aturan-aturan konversi bisa berjalan dengan baik perlu juga aturan-aturan yang mendasari pembuatan BPM. Didalam *paper* ini juga memuat langkah-langkah yang memuat cara untuk membuat bpmn agar sesuai dengan metode ini.Berikut aturan-aturan yang dalam membuat BPMN:

1. Aturan untuk membangun Model Diagram Kolaborasi BPMN

a. Menjelaskan secara sederhana dan tidak kompleks.Untuk pembuatan sub proses berkisar 4 sampai 12 *task.*

b. Jika suatu sub proses kurang dari 4 *task* atau merupakan operasi pelengkap, maka akan digabungkan menjadi satu sub proses

c. Hindari semaksimal mungkin , penyajian tugas manual

d. Model hanya menyajikan deskripsi urutan kegiatan proses bisnis yang umum

e. Fokus pada sub-proses dan urutannya

f. Identifikasi actor yang terlibat

g. Hindari pengunaan gateway pada *collaboration diagram.*

2. Aturan untuk membangun Model Diagram Proses BPMN

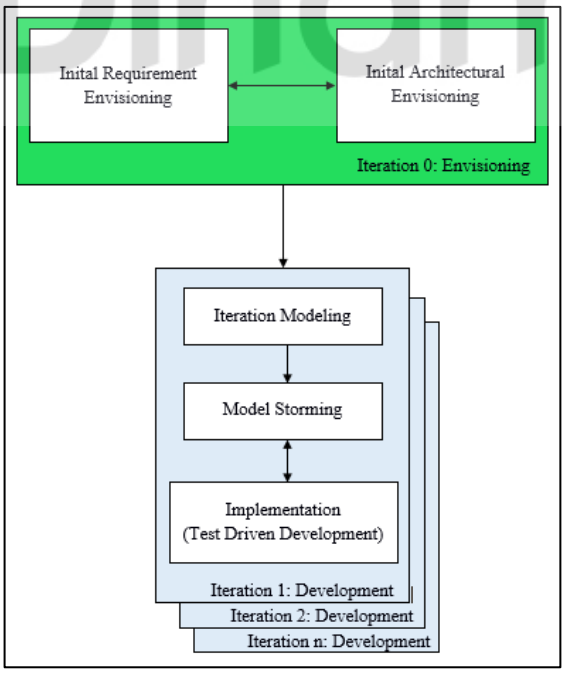
a. Detailkan sub proses menjadi beberapa *task*

b. Jangan mewakili tugas manual dari *collaboration* diagram

c. Memiliki Gateway

d. Menambahkan *dataobject* pada setiap *task*

2.6. Agile Software Development Phase



Gambar 2.6 Model Pengembangan Agile
(Sumber : Ambler)

Menurut Ambler (2012), *Agile Model-Driven Development* (AMDD) merupakan *Model-Driven Development* (MDD) versi Agile. MDD sendiri adalah sebuah pendekatan dalam pengembangan perangkat lunak dengan cara memodelkan secara menyeluruh sebelum memulai menulis kode. Akan tetapi AMDD tidak memodelkan secara menyeluruh dan membuat model agar segera dieksekusi. Ambler (2012) membuat ilustrasi AMDD pada diagram yang dapat dilihat pada Gambar 2.6.

2.6.1. Envisioning

Fase *envisioning* adalah iterasi ke 0 dari agile. Fase *envisioning* mengidentifikasi ruang lingkup sistem yang akan dibuat dan idendifikasi arsitektur yang dibutuhkan. Tujuan dilakukannya fase ini adalah untuk mengeksplorasi persyaratan dan strategi menyeluruh terhadap sistem yang akan dibuat. Terdapat 2 tahapan dalam fase *envisioning*, diantaranya adalah sebagai berikut:

a) Initial Requirement

Menurut Ambler (2012) pada tahapan ini digunakan untuk memodelkan *requirements* awal mengenai sistem yang akan dibuat. Hal ini agar setiap tim dapat memahami tujuan dari dibuatnya sistem (bukan untuk membuat dokumen secara rinci). Model yang akan dibuat pada tahapan ini adalah *usage model*, *domain model* dan *user interface model*.

1. *Usage model* menggambarkan interaksi *user* dan sistem. *Usage model* digambaran dengan *user stories.*
2. *Domain model* mengambarkan interaksi antara entitas bisnis dan relasinya. *Domain model* dapat digambarkan dengan use case.
3. *User interface model* merupakan draf rancangan *user interface* dari sistem.

b) Initial Architectural

Menurut Ambler (2012) tahapan ini bertujuan untuk mengidentifikasi arsitektur. Pemodelan arsitektur bersifat dinamis. Menurut Astamal (2013) Pemodelan Arsitektur dapat digambarkan dengan deployment diagram.

2.6.2. Development

Tahap *development* merupakan gabungan dari fase *Iteration Modeling*, *Model Storming* dan *Executable Specification*. Ketiga fase tersebut digabung dikarenakan akan ada proses perulang ke tiga fase tersebut pada iterasi. Iterasi akan diulang berdasarkan *user story* atau fitur sistem yang akan dibuat.

A. Iteration Modeling

Pada tahapan ini dilakukan estimasi waktu yang dibutuhkan untuk mengerjakan suatu sistem. Perhitungan waktu tersebut berdasarkan dari fitur apa saja di setiap iterasi, kemudian dilakukan membagi berdasarkan prioritas .Selain berdasarkan fitur perhitungan estimasi dapat dilakukan dengan melihat *use case* dan desain *user interface* yang telah dibuat sebelumnya. Selain itu terjadi pendetilan aktifitas pada *use case* tersebut, pendetilan dapat digambarkan dengan menggunakan activity diagram.

B. Model Storming

Tahapan ini mengidentifikasi masalah yang perlu diselesaikan. Identifikasi ini dilakukan oleh tim yang akan mengerjakan sistem. Menurut (Ambler S. , 2012) Terdapat dua tahap pada yang harus dilakukan pada Model *Stroming* diantaranya adalah *Analysis Model Storming* dan *Design Model Storming*.

1. *Analysis Model Storming* : melakukan analisis kebutuhan menggunakan *user interface* atau *wireframe*

2. *Design Model Storming* : melakukan penggambaran alur menggunakan *sequence diagram* atau *flow chart*

C. Implementation

Pada fase ini setelah analisis dan desain sistem telah selesai maka akan dilakukan pembuatan desain dan skenario *testing* kemudian baru dilakukan penulisan kode. Testing yang digunakan adalah Test Driven Development (TDD). Menurut (Wibowo, 2017) TDD adalah metode testing yang menjadikan testing sebagai acuan pengembangan suatu aplikasi atau sistem

BAB III

METODOLOGI PENELITIAN

3.1. Identifikasi Masalah

Permasalahan yang sering terjadi dalam proses transformasi dari *Computation Independent Model* (CIM) ke *Platform Independent Model* (PIM) adalah masih dilakukan manual dan dikhawatirkan tidak mematuhi kaidah dalam *Model Driven Architecture* (MDA). Ataupun dalam prosesnya tidak mengambarkan secara utuh apa yang terjadi di CIM. Proses konversi dari CIM ke PIM sangat bergantung terhadap perancang sehingga kualitas dari PIM itu sendiri tidak terkontrol (Wei, Mei, Zhao, & Jie Yang, 2005).

Oleh karena itu dengan aplikasi ini diharapkan adanya standarisasi dalam proses konversi dari *Business Process Model Notation* (BPMN) ke *Unified Modeling Language* (UML).Dan juga bisa mengurangi waktu yang digunakan untuk mengkonversi BPMN ke UML.

3.2. Studi Literatur

Pada tahap ini penulis mengkaji dan mencari beberapa referensi teori yang sesuai dengan topik yang diangkat oleh penulis.Referensi yang dimaksud adalah sebagai berikut:

1. *Agile Model Development Driven* (AMDD)
2. Transformation Method CIM to PIM:From Business Process Models Defined in BPMN to Use Case and Class Models Defined in UML
3. XML *Concept*

Referensi diatas didapatkan dari buku, jurnal, artikel, laporan penelitian dan situs internet. Pengambilan data, pemrosesan data, dan *output* didasarkan pada studi literatur diatas.Studi literatur diatas juga digunakan untuk memperkuat dasar teori dalam melakukan penelitian ini.Data-data yang didapat akan dipergunakan untuk membantu penulis dalam proses pembuatan aplikasi, pembuatan sample, dan lain sebagainya.

3.3. Model Konversi BPMN ke UML

Pada tahap ini penulis membuat rancangan konversi Business Process Model Notation (BPMN) ke Unified Modeling Language (UML). Hal ini dibutuhkan agar pembaca dapat memahami dengan baik rancangan yang akan dibuat berdasarkan observasi dan studi literatur yang dilakukan oleh penulis. Pada gambar 3.1 menunjukan alur dari BPMN Bizagi ke UML.

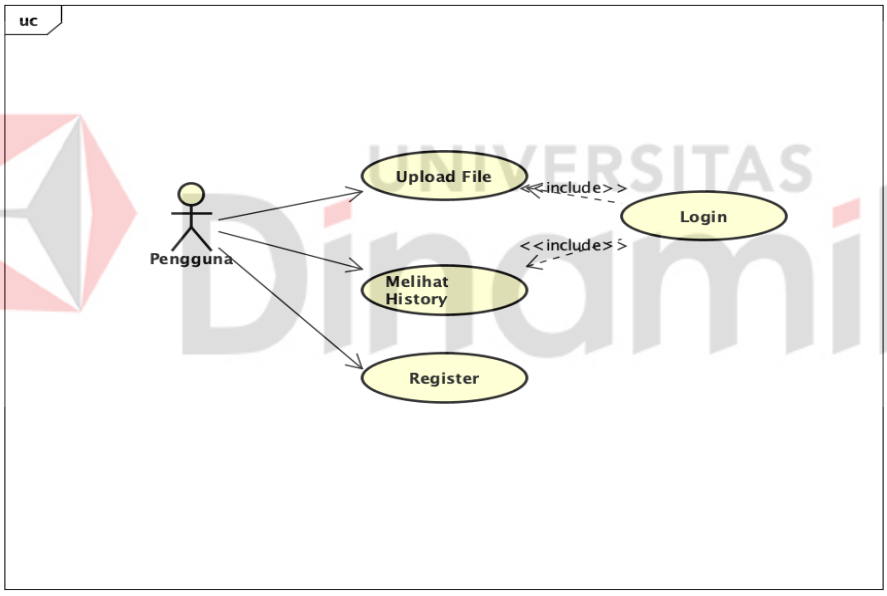
Gambar 3.1 Konsep Konversi

Pada gambar 3.1 menjelaskan proses konversi diawali dari BPMN bizagi yang akan diambil XMLnya.XML yang didapat dari BPMN tersebut nantinya akan

diubah menjadi JSON. Untuk proses *Generate* UML menggunakan aturan yang dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014). Setelah mempelajari aturan-aturan dari (Rhazali, Mouloudi, & Hadi, 2014) , penulis memutuskan untuk mengambil data-data yang di BPMN yaitu :

1. *Collecting Usecase*
2. *Collecting Task*
3. *Collecting TaskRelation*
4. *Collecting Dataobject*

A. Usecase



Gambar 3.2 Use case Aplikasi Konversi

Use Case Aplikasi untuk Konversi BPMN ke UML Berbasis Web terdapat pada gambar 3.2. *Use case* terdiri dari satu aktor yaitu pengguna. Dan memiliki fitur *register*,*upload* file bpm,dan melihat *history*(*log*).

B. Konsep

Penulis akan menjelaskan proses konversi BPMN ke UML, Data yang dibutuhkan oleh sistem ini adalah *file* BPM (Bizagi Modeler) :

B.1. BPM *to* XML

Pada tahap ini akan melakukan ekstraksi *file* BPM. Tujuan dari tahap ini adalah untuk mendapat diagram.xml.

1. Dari *file* BPM yang di *upload* oleh pengguna , *file* BPM tersebut akan disimpan dalam suatu *folder.* Proses ini dapat dilihat pada gambar L1.19
2. Didalam *folder* tersebut, sistem akan melakukan ekstraksi *file* BPM untuk mendapatkan *file* dengan ekstensi .diag. Proses ini dapat dilihat pada gambar L1.25
3. Sistem akan melakukan pendeteksian terkait jumlah *file* dengan ekstensi .diag dan melakukan ekstraksi terhadap setiap *file* dengan ekstensi .diag untuk mendapatkan *file* diagram.xml. Proses ini dapat dilihat pada gambar L1.26

B.2. *Collecting Data*

Pada tahap ini sistem akan melakukan ekstrasi data diagram.xml yang nanti nya akan digunakan untuk membangun *use case* diagram dan *class* diagram.

1. Dari diagram.xml, sistem akan melakukan pengumpulan *task* yang di bagian "*Activity"* dan untuk *component* proses di *use case*. Dari proses ini akan menghasilkan data *task*. Data *task* yang didapatkan dikelompokan berdasarkan nama diagram dan ada tambahan *start, end* dan *gateway*. Proses ini dapat dilihat pada gambar L1.30.

2. Dari diagram.xml, sistem akan melakukan pengumpulan *pool* untuk mengetahui aktor dari suatu *task*. *Pool* ini diambil dari bagian "*Workflowprocess task"*. Dari proses ini akan menghasilkan data *use case.* Proses ini dapat dilihat pada gambar L1.28.
3. Dari diagram.xml, sistem akan melakukan pengumpulan *task relation* untuk mengetahui relasi antar *task. Task relation* berada di bagian "*transition"* tetapi didalam "*transition"* masih berupa *uid* sehingga perlu proses *translate* untuk mendapatkan nama *activity.* Proses *translate* ini melakukan pencocokan dari *uid* di "*transition"* dengan *uid* di "*activity"*. Dari proses ini akan menghasilkan data *task relation*. Proses ini dapat dilihat pada gambar L1.31.
4. Dari diagram.xml, sistem akan melakukan pengumpulan *dataobject*. *Dataobject* ini nantinya akan diperlukan untuk *class* diagram. Dari proses ini akan menghasilkan data *dataobject*. Proses ini dapat dilihat pada gambar L1.29.

B.3. Validasi

Pada tahap ini akan mengolah data – data yang didapat dari tahap *collecting data*. Dari data yang didapat dari tahap *collecting data*, data tersebut akan dibersihkan agar tahap *generate uml* tidak terganggu.

1. Dari data *dataobject*, sistem akan melakukan pengecekan dan pembersihan terhadap data tersebut sehingga tidak ditemukan kembali nama *dataobject* yang sama. Sehingga didapatkan data *dataobject* yang bersih. Proses ini dapat dilihat pada gambar L1.30.
2. Dari data *task,* sistem akan melakukan pengecekan dan pembersihan terhadap data tersebut sehingga tidak ditemukan kembali nama *start, end,* dan *gateway*.

Sehingga didapatkan data *dataobject* yang bersih. Proses ini dapat dilihat pada gambar L1.31.

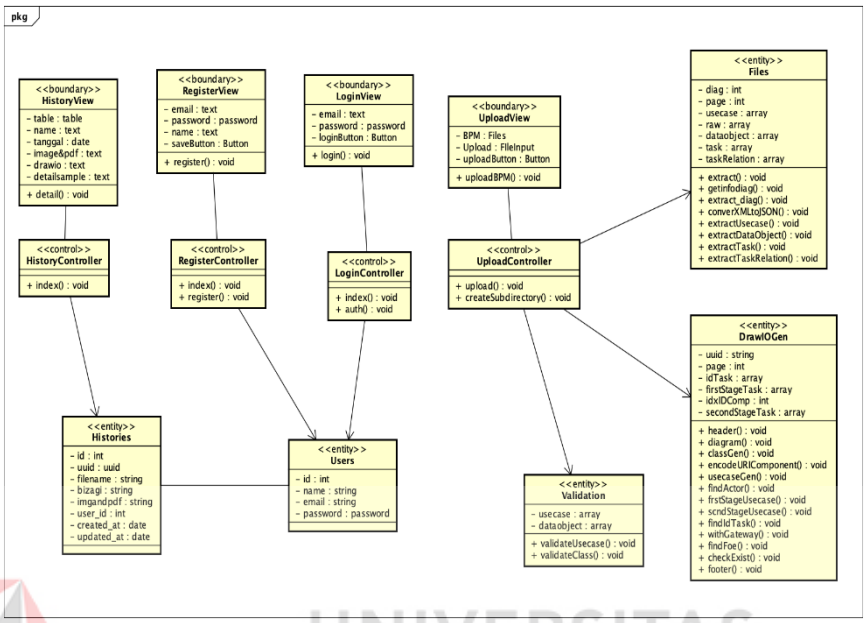
B.4. *Generate UML*

Pada tahap ini, sistem akan mengolah data – data yang didapat dari tahap validasi. Dikarenakan data yang didapat sudah bersih maka pada tahap ini sistem tidak akan memperhatikan kualitas dari data yang didapat.

1. *Header* dari data *task* akan dijadikan *package* pada *usecase* diagram. Proses ini dapat dilihat pada gambar L1.39.
2. Data *usecase* yang didapat akan dicocokan dengan nama *task* jika ditemukan sama maka data *usecase* dari *task* tersebut akan menjadi aktor. Proses ini dapat dilihat pada gambar L1.41.
3. Data *task* akan dirubah menjadi proses di *use case* diagram. Proses ini dapat dilihat pada gambar L1.41.
4. Mencari *task* yang dekat dengan *end* dan *start* , ketika sudah ditemukan maka akan dihubungkan ke aktor dengan relasi asosiasi. Proses ini dapat dilihat pada gambar L1.41.
5. Berdasarkan *task* yang memiliki relasi asosiasi akan diperiksa. Proses ini dapat dilihat pada gambar L1.41.
    a. Jika ada *task* sebelum task yang memiliki relasi asosiasi akan dihubungkan dengan relasi *include* dari *task* yang memiliki relasi asosiasi.
    b. Jika ditemukan gateway didepan *task* yang memiliki relasi asosiasi akan meloncati relasi yang ada sampai menemukan *task* jika sudah ditemukan maka *task* yang ditemukan akan memiliki relasi *extend* ke task yang memiliki relasi asosiasi

6. Semua *dataobject* akan diubah menjadi *classes.* Proses ini dapat dilihat pada gambar L1.40

C. *Class* Diagram



Gambar 3.3 Class Diagram Aplikasi Konversi

Gambar 3.3 merupakan *class diagram* Aplikasi Konversi BPMN ke UML yang menggambarkan relasi antar *class* yang mempunyai peran-peran tersendiri seperti *login, register,* pengambilan data, *validation, generate* DrawIO.

Setelah melakukan analisis dari hasil observasi dan ujicoba di temukan beberapa kekurangan diantara lain :

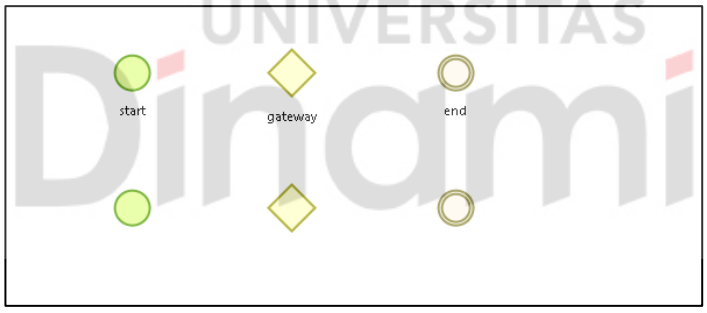
1. Hanya *Component* yang berupa *task* saja yang memiliki nama
2. Tidak ada konektivitas antara *Collaboration Diagram* dan *Process Diagram*
3. Penyimpanan pada *taskrelation* tidak berurutan (sesuai dengan tahapan yang dilakukan *creator*)
4. Tidak ada aturan tertulis yang mengatur relasi asosiasi pada Aturan yang dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014)

5. Tidak ada aturan tertulis yang mengatur arah dari dari relasi *extend* maunpun *include* pada Aturan yang dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014)

Untuk mengatasi kekurangan diatas, maka penulis membuat aturan tambahan agar file BPM dapat diproses oleh aplikasi ini. Proses ini merupakan bagian pengembangan yang dilakukan penulis. Berikut adalah aturan aturan tersebut :

a. Setiap *component* non-*task* harus diberi nama

Component didalam process-level harus lah diberi nama sesuai dengan nama *component* nya . Yang dimaksud *component* adalah seperti *start, end, gateway,* dan *dataobject.* Jika tidak ditambahkan nama *component* tersebut maka aplikasi tidak dapat mengidentifikasi *component* tersebut. Hal tersebut dapat mempengaruhi hasil dari aplikasi ini. Contoh pemberian nama dapat dilihat pada gambar 3.4.

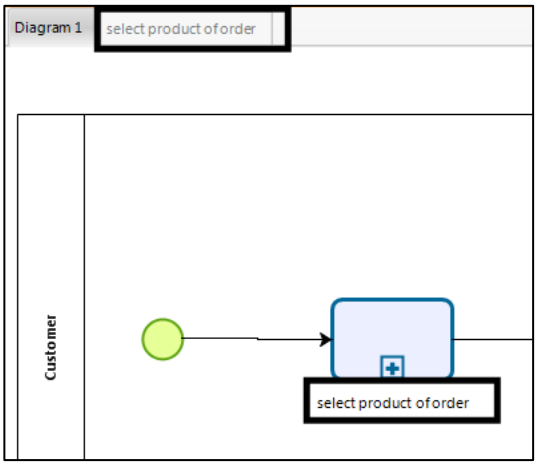


Gambar 3.4 Penambahan nama pada *component* non *task*

b. Pembuatan harus dibuat secara berurutan

Dalam proses pembuatan bpmn di Bizagi Modeler harus berurutan. Yang dimaksud berurutan adalah setiap penambahan *start,task,relation* dan *end* harus dari hulu ke hilir. Hal tersebut diperlukan agar aplikasi dapat memahami maksud dari bpmn yang dibuat.

c. Penamaan sub diagram



Gambar 3.5 Penamaan Diagram *Sub process*

Dalam proses pembuatan bpmn di Bizagi Modeler, pengguna harus memperhatikan nama diagram pada *process level*.Pada pembuatan *process level* diagram nama dari diagram harus diganti dengan nama task yang diwakili di *collaboration* diagram.Contoh hal ini bias dilihat di gambar 3.5.

d. Relasi Asosiasi akan ditentukan berdasarkan *start* dan *end*

Dikarenakan terdapat beberapa hal yang tidak muat pada *paper* (Rhazali, Mouloudi, & Hadi, 2014). Penulis berusaha mengatasi kekurangan yang terjadi dengan menambahkan beberapa aturan di sistem. Berdasarkan observasi yang dilakukan oleh penulis, penulis beranggapan bahwa *usecase* yang memiliki relasi asosiasi dengan aktor adalah *usecase*/*task* yang berada didekat *start* dan *end* pada BPMN. Observasi ini didasari pada lampiran 32.

e. Arah Relasi *extend* dan *include*

Dikarenakan terdapat beberapa hal yang tidak muat pada *paper* (Rhazali, Mouloudi, & Hadi, 2014). Penulis berusaha mengatasi kekurangan yang terjadi dengan menambahkan beberapa aturan di sistem. Berdasarkan observasi yang dilakukan oleh penulis, penulis beranggapan bahwa relasi *include* jika ada di proses level ada dua *task* yang terhubung dengan *sequence* maka *task* kedua menuju ke

task kesatu. Untuk relasi *extend* jika ada di proses level ada *task* yang melewati *gateway* dan setelah *gateway* ada *task* maka *task* setelah *gateway* akan menuju ke *task* sebelum *gateway* dan jika setelah *gateway* berujung ke dua *task* maka dua *task* itu akan menuju ke *task* sebelum *gateway*. Observasi ini didasari pada lampiran 32.

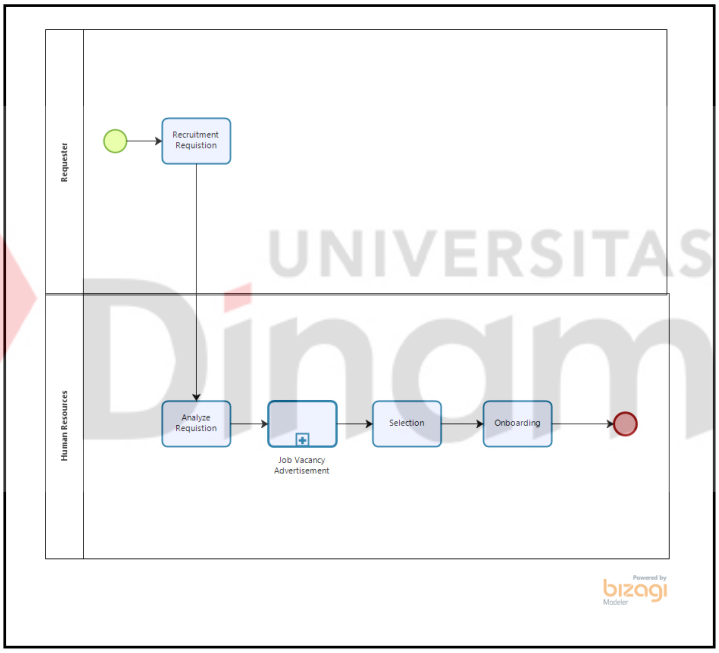
BAB IV

HASIL DAN PEMBAHASAN

4.1. Hasil

Pada tahap ini penulis akan menjelaskan hasil dari apa yang dilakukan oleh penulis. Penulis juga akan melakukan pengujian terhadap *fungsional* sistem dan menguji terhadap aturan dari (Rhazali, Mouloudi, & Hadi, 2014).

4.1.1. Contoh *Sample Recruitment*



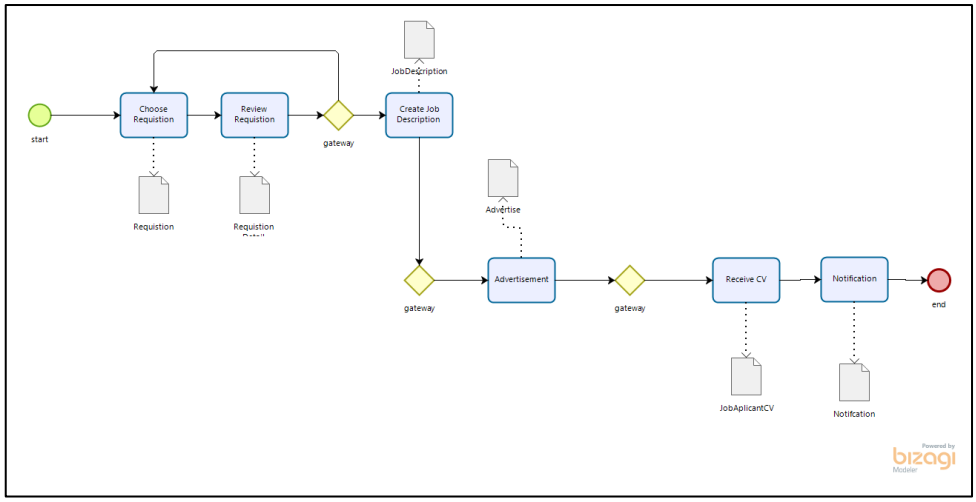
Gambar 4.1 *Collaboration Level Sample*

Pada sample ini menggambarkan proses bisnis rekrutmen dari mulai requester mengajukan penambahan pegawai sampai penerimaan pegawai. Proses bisnis ini diawali dari requester mengajukan daftar penambahan pegawai yangdiinginkan.Lalu daftar tersebut akan dianalisis oleh human resource. Setelah dianalisis dan diperiksa maka akan melakukan advertisement.

Setelah diiklankan pihak *human resource* akhirnya memilih para *job applicant* yang akan di melalui proses *selection.* Jika *job applicant* sudah melalui proses *selection* dan diterima kan langkah selanjutnya adalah *onboarding*. Proses dapat dilihat pada gambar 4.1.

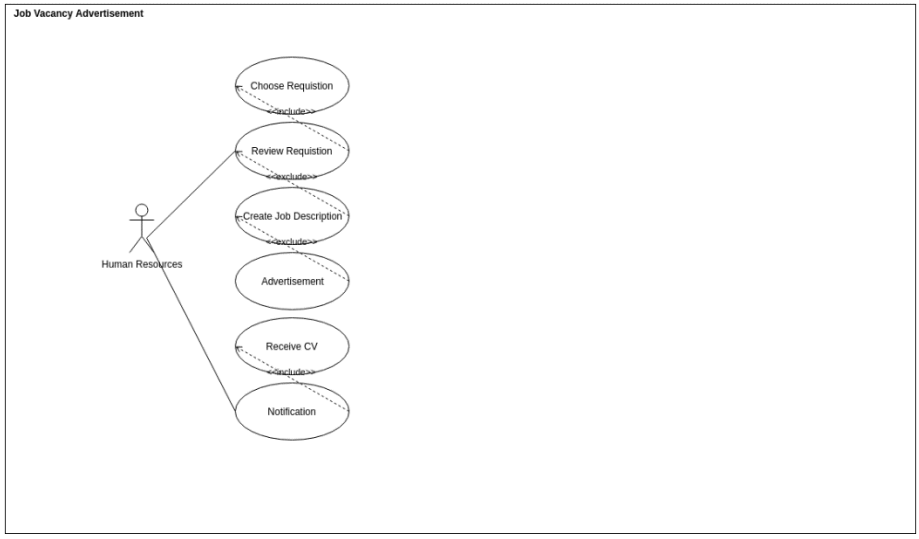
Tetapi *collaboration* diagram tidak menggambarkan proses yang lebih detail. Oleh karena itu jika ingin lebih menjelaskan lebih detail lagi akan dijelaskan di *process level* diagram. Pada *sample* ini akan men-detailkan satu *task* yaitu "*job vacancy advertisement*".

Pada process level "job vacancy advertisement", dimulai dari memilih requisition . lalu human resource akan mereview requisition jika diterima human resource pihak human resource bisa menambahkan job description.Jika human resource sudah cocok dengan requistion dapat melakukan advertising.Dalam waktu advertising dipublikasikan akan ada job applicant yang mengirimkan CV dan lalu mengirimkan notifikasi ke human resource untuk melakukan pengecekan. Proses yang ada dapat dilhat pada gambar 4.2.

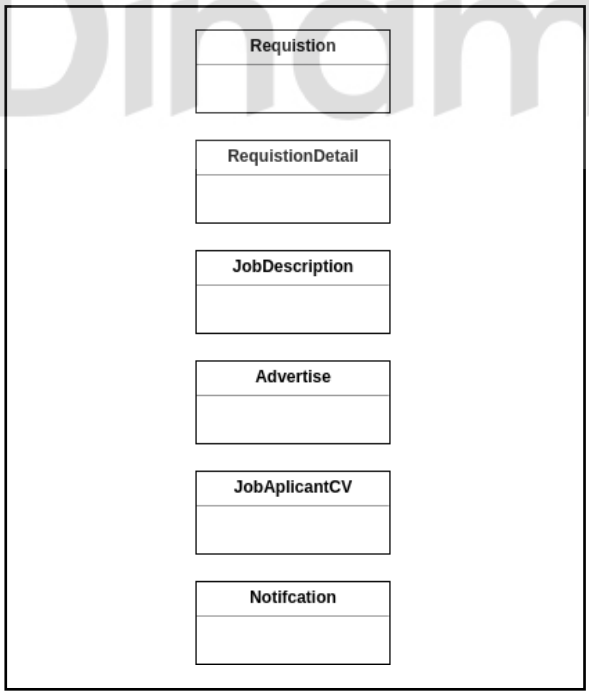


Gambar 4.2 *Process level* diagram "*job vacancy advertisement*"

Setelah BPMN tersebut dikonversi maka hasilnya dapat dilihat pada gambar 4.3 dan gambar 4.4.Tahapan dari proses konversi dapat dilihat pada lampiran 2.



Gambar 4.3 Hasil *Usecase* diagram *Recruitment*



Gambar 4.4 Hasil *Class* Diagram *Recruitment*

Dari BPMN diatas penulis akan melakukan pengujian terhadap tahapan-tahapan yang sudah dibuat :

Header dari data *task* akan dijadikan *package* pada *usecase* diagram jika melihat pada gambar 4.2 bahwa gambar tersebut merupakan gambaran BPMN dari proses level diagram "*job vacancy advertisement*" maka seharusnya pada gambar 4.3 pada bagian *package* adalah *job vacancy advertisement.*

1. Data *usecase* yang didapat akan dicocokan dengan nama *task* jika ditemukan sama maka data *usecase* dari *task* tersebut akan menjadi aktor.

Pada gambar 4.1 menjelaskan *collaboration level* diagram BPMN. Di gambar tersebut ada satu *task* yang memiliki *subproses* yaitu *job vacancy advertisement*. *Task job vacancy advertisement berada di* dalam *pool human resource.* Berdasarkan aturan yang disebutkan sebelumnya bahwa aktor dari *job vacancy advertisement* adalah *human resources*

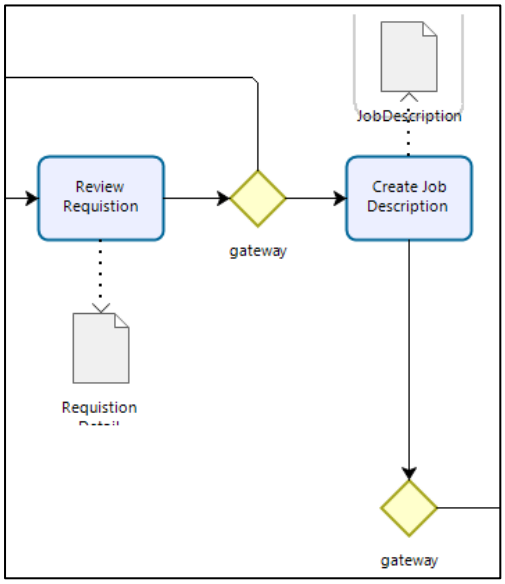
2. Data *task* akan dirubah menjadi *usecase*

Pada gambar 4.2 menunjukkan terdapat ada 6 task yaitu choose requisition, review requistion, create job description, advertisement, receive cv, dan notification.Dikarenakan semua task akan menjadi usecase maka didalam usecase diagram juga seharusnya terdapat 6 usecase. Pada gambar 4.3 dapat dilihat bahwa usecase diagram hasil konversi memiliki 6 usecase.

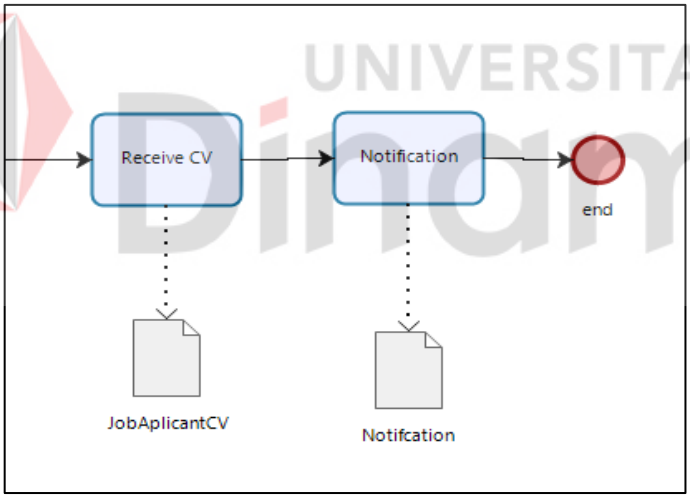
3. Mencari *task* yang dekat dengan *end* dan *start* , ketika sudah ditemukan maka akan dihubungkan ke aktor dengan relasi asosiasi.

Berdasarkan gambar 4.2 dapat dilihat bahwa terdapat 1 *start* dan 1 *end* maka yang diambil adalah *review requisition dan notification*. Pada gambar 4.3 dapat dibuktikan kedua *task* tersebut memiliki relasi asosiasi.

4. Berdasarkan *task* yang memiliki relasi asosiasi akan diperiksa.



Gambar 4.5 BPM Fokus Relasi *Gateway*



Gambar 4.6 BPM Fokus Relasi *Sequence*

a. Jika ada *task* sebelum task yang memiliki relasi asosiasi akan dihubungkan dengan relasi *include* dari *task* yang memiliki relasi asosiasi.

Gambar 4.6 menjelaskan bahwa *task* "*ReceiveCV*" sudah selesai , maka *human resource* mendapat *notification*. Berarti seharusnya "*notification* " memiliki

relasi *include* ke "*receive cv*".Karena *cv job applicant* harus diterima terlebih dahulu baru bisa mendapatkan notifikasi. Setelah dicek pada gambar 4.3 hasilnya pun sesuai dengan ekspetasi maka sudah bisa disimpulkan bahwa aturan ini telah sukses dieksekusi oleh program.

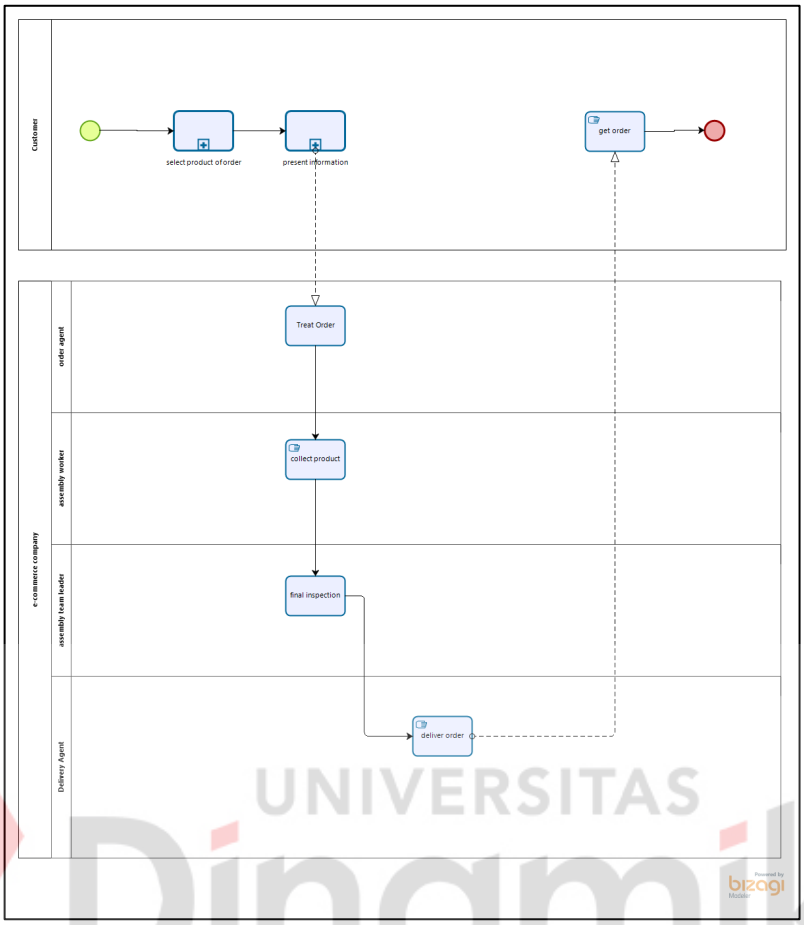
b. Jika ditemukan gateway didepan *task* yang memiliki relasi asosiasi akan meloncati relasi yang ada sampai menemukan *task* jika sudah ditemukan maka *task* yang ditemukan akan memiliki relasi *extend* ke task yang memiliki relasi asosiasi

Gambar 4.5 menjelaskan bahwa *task* "*Review Requistion*" terhubung ke gateway dan ujung dari gateway itu bermuara ke *task* "*change product quantity*". Berarti seharusnya "*Create job description*" memiliki relasi *extend* ke "*review requistion*".Setelah dicek pada gambar 4.3 hasilnya pun sesuai dengan ekspetasi maka sudah bisa disimpulkan bahwa aturan ini telah sukses dieksekusi oleh program.

5. Semua *dataobject* akan diubah menjadi *classes*

Berdasarkan process level diagram yang dapat dilihat pada gambar 4.2 terdapat 6 dataobject. Dataobject tersebut adalah Requistion ,RequistionDetail , JobDescription , Advertise , JobApplicantCV , dan Notifcation. Berarti seharusnya 6 dataobject diatas menjadi kelas-kelas.Dan pada gambar 4.4, 6 dataobject tersebut sudah menjadi classes. Maka sudah bisa disimpulkan bahwa aturan ini telah sukses dieksekusi oleh program.

4.1.2. Contoh *Sample E-Commerce*

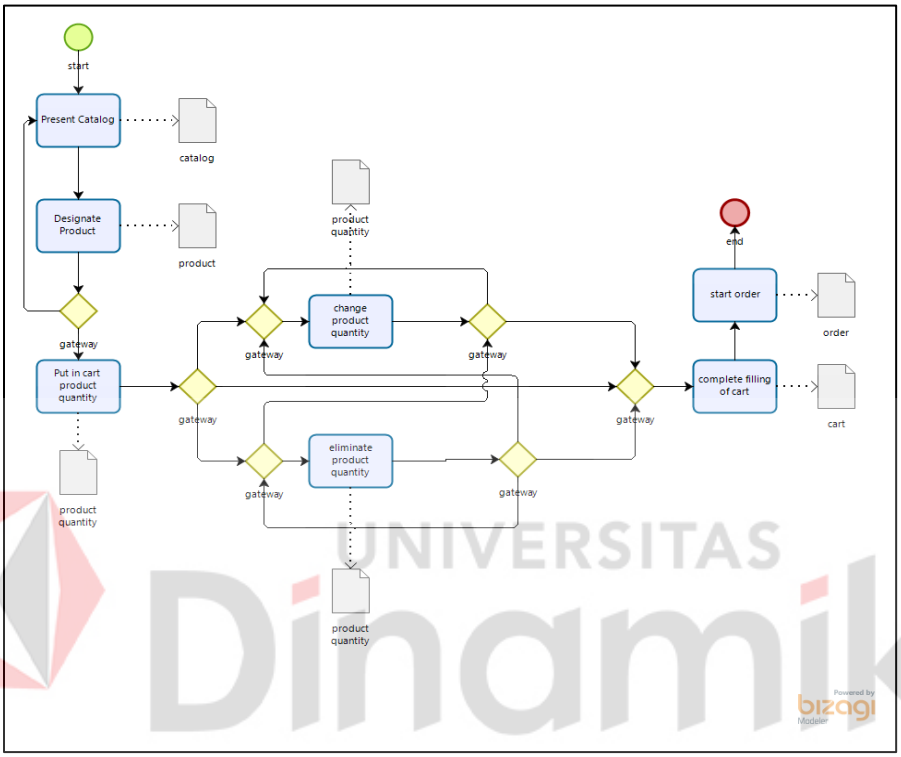


Gambar 4.7 *Collaboration Level Sample E-Commerce*

Pada *sample* ini menggambarkan proses bisnis *e-commerce* dari mulai *customer* memilih produk sampai produk tersebut diterima. Proses bisnis ini diawali dari *customer* memilih produk/barang yang diinginkan.Lalu ketika sudah dipilih akan munculkan detail produk/barang yang diinginkan setelah itu pesanan tersebut akan diurus oleh *order agent.* Pesanan yang sudah diurus oleh *order agent* akan diambil oleh *assembly worker.*

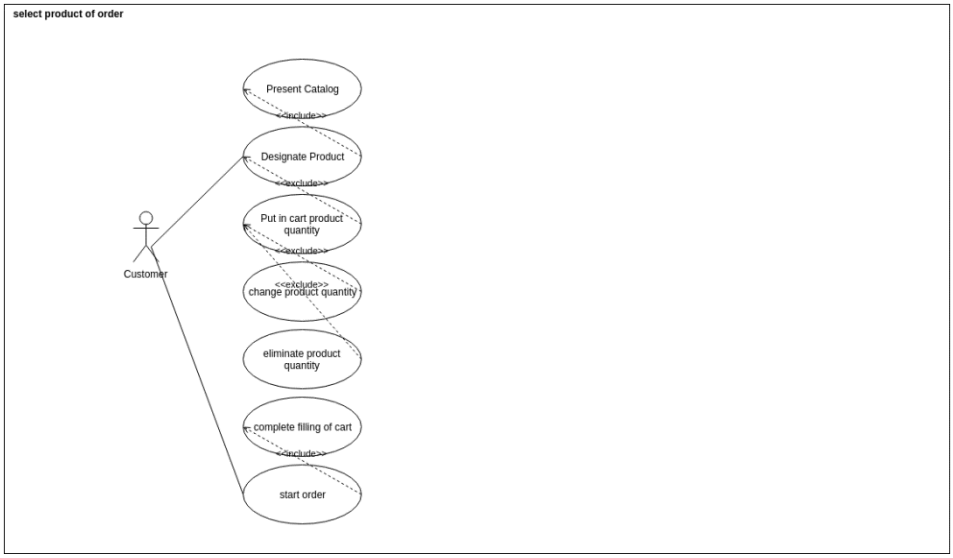
Setelah diambil pesanan tersebut, maka akan diperiksa oleh *assembly worker leader. Deliver agent* akan mengirimkan ke *customer* ketika paket sudah dicek oleh *assembly worker leader*.Dan proses berakhir ketika paket diterima oleh *customer*. Proses dapat dilihat pada gambar 4.7.

Tetapi *collaboration* diagram tidak menggambarkan proses yang lebih detail. Oleh karena itu jika ingin lebih menjelaskan lebih detail lagi akan dijelaskan di *process level* diagram. Pada sample 1 ini akan men-detailkan satu *task* saja yaitu "*select product of order*".

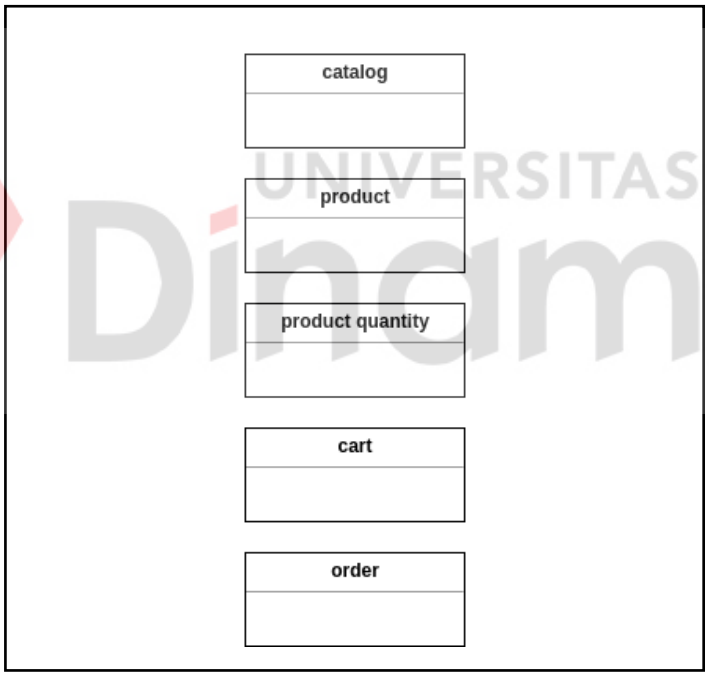


Gambar 4.8 *Process level* diagram "*select product of order*"

Pada *process level* "*select product of order*", dimulai dari menampilkan katalog.lalu *customer* juga bisa meng-*custom* warna,ukuran,dan hal lainnya jika dirasa tidak cocok *customer* dapat memilih produk yang lain.Jika *customer* sudah cocok dengan pilihan dan detail barangnya dapat melakukan melakukan jika tidak *customer* dapat merubah detail detail tersebut. Hal ini dapat dilihat pada gambar 4.8.



Gambar 4.9 Hasil *Usecase* diagram *E-Commerce*



Gambar 4.10 Hasil *Usecase* diagram *E-Commerce*

Dari BPMN diatas penulis akan melakukan pengujian terhadap tahapan-tahapan yang sudah dibuat :

1. *Header* dari data *task* akan dijadikan *package* pada *usecase* diagram

Jika melihat pada gambar 4.8 bahwa gambar tersebut merupakan gambaran BPMN dari proses level diagram "*select product of order*" maka seharusnya pada gambar 4.9 pada bagian *package* adalah *select product of order.*

2. Data *usecase* yang didapat akan dicocokan dengan nama *task* jika ditemukan sama maka data *usecase* dari *task* tersebut akan menjadi aktor.

Pada gambar 4.7 menjelaskan *collaboration level* diagram BPMN. Di gambar tersebut ada satu *task* yang memiliki *subproses* yaitu *select product of order*. *Task job vacancy advertisement berada di* dalam *pool customer.* Berdasarkan aturan yang disebutkan sebelumnya bahwa aktor dari *select product of order* adalah *customer.*

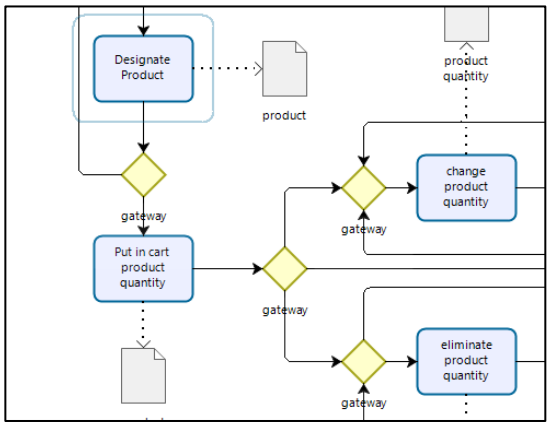
3. Data *task* akan dirubah menjadi *usecase*

Pada gambar 4.8 menunjukkan terdapat ada 7 task yaitu present catalog, designate catalog, put in cart product quantity, change product quantity, eliminate product quantity, complete filling of cart, dan notification.Dikarenakan semua task akan menjadi usecase maka didalam usecase diagram juga seharusnya terdapat 7 usecase. Pada gambar 4.9 dapat dilihat bahwa usecase diagram hasil konversi memiliki 7 usecase.

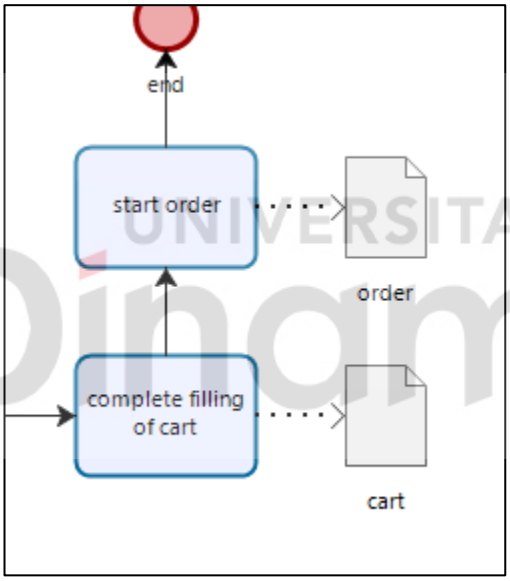
4. Mencari *task* yang dekat dengan *end* dan *start* , ketika sudah ditemukan maka akan dihubungkan ke aktor dengan relasi asosiasi

Berdasarkan gambar 4.8 dapat dilihat bahwa terdapat 1 *start* dan 1 *end* maka yang diambil adalah *start order* dan *designate product*. Pada gambar 4.9 dapat dibuktikan kedua *task* tersebut memiliki relasi asosiasi.

5. Berdasarkan *task* yang memiliki relasi asosiasi akan diperiksa.



Gambar 4.11 BPM Fokus Relasi *Gateway Sample E-Commerece*



Gambar 4.12 BPM Fokus Relasi *Sequence Sample E-commerce*

a. Jika ada *task* sebelum task yang memiliki relasi asosiasi akan dihubungkan dengan relasi *include* dari *task* yang memiliki relasi asosiasi.

Gambar 4.11 menjelaskan bahwa *task* "*complete filling of cart*" sudah selesai , maka *customer* dapat memesannya . Berarti seharusnya "*start order* " memiliki relasi *include* ke "*complete filling of cart*".Karena *cart* harus diisi terlebih dahulu baru bisa memesan. Setelah dicek pada gambar 4.9 hasilnya pun sesuai dengan

ekspetasi maka sudah bisa disimpulkan bahwa aturan ini telah sukses dieksekusi oleh program.

b. Jika ditemukan gateway didepan *task* yang memiliki relasi asosiasi akan meloncati relasi yang ada sampai menemukan *task* jika sudah ditemukan maka *task* yang ditemukan akan memiliki relasi *extend* ke task yang memiliki relasi asosiasi

Gambar 4.10 menjelaskan bahwa *task* "*put in cart product quantity*" terhubung ke gateway dan ujung dari gateway itu bermuara ke dua *task* . Task tersebut adalah "*change product quantity*" dan "*eliminate product quantity*". Berarti seharusnya "*change product quantity*" dan "*eliminate product quantity*" memiliki relasi *extend* ke "*put in cart product quantity*".Setelah dicek pada gambar 4.9 hasilnya pun sesuai dengan ekspetasi maka sudah bisa disimpulkan bahwa aturan ini telah sukses dieksekusi oleh program.

6. Semua *dataobject* akan diubah menjadi *classes*

Berdasarkan data yang didapat terdapat 5 *dataobject. Dataobject* tersebut adalah *catalog ,product ,product quantity ,cart, dan order.*Berarti seharusnya 5 *dataobject* diatas menjadi kelas-kelas.Dan pada gambar 4.20, 5 dataobject tersebut sudah menjadi *classes.* Maka sudah bisa disimpulkan bahwa aturan ini telah sukses dieksekusi oleh program.

4.1.3. Software Testing

1. Output Unit Testing Iterasi 1

Pada tahap ini penulis akan menunjukan hasil dari unit test yang mengunakan phpunit. *Output* unit testing iterasi 1 melakukan pengetesan pada iterasi 1. *Output* dari unit ditunjukkan oleh tabel 4.1.

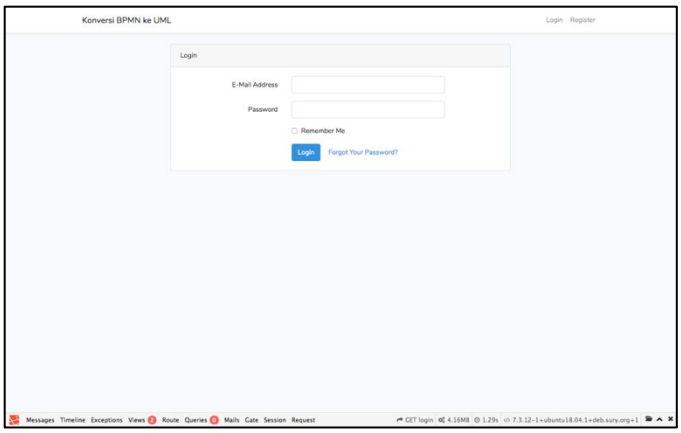
Tabel 4.1 *Output* Unit *Testing* Iterasi 1

No	Tes	Test 1	Test 2	Presentase
1.	*Log in* email dan password benar	*pass*	*pass*	100%
2.	*Log in* email dan password salah	*pass*	*pass*	100%
3.	Masuk tanpa *Log in*	*pass*	*pass*	100%
4.	Register Tanpa Diisi semua	*pass*	*pass*	100%
5.	Register dengan diisi semua	*pass*	*pass*	100%
6.	Upload File dengan ekstensi bpm	*pass*	*pass*	100%
7.	Upload File dengan ekstensi exe	*pass*	*pass*	100%

Penulis tidak membuat testing pada iterasi 2 dan iterasi 3 dikarena kan pada iterasi 2 dan iterasi 3 membahas proses konversi . Jadi iterasi 2 dan iterasi 3 akan dijelaskan pada bagian "Contoh *Sample*" yang berada pada bab 4.

4.1.4. Tampilan Aplikasi Konversi BPMN ke UML

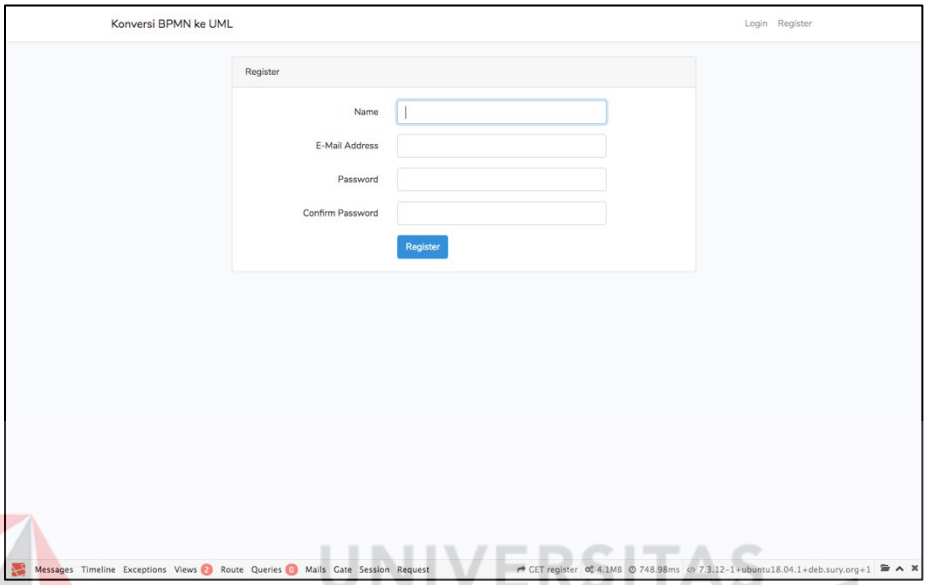
4.1.4.1. Login



Gambar 4.13 Halaman UI *Login*

Halaman *Login* merupakan halaman dimana pengguna dapat memasukkan *email* dan *password* yang telah didaftarkan. UI dari halaman *login* dapat dilihat pada gambar 4.14.

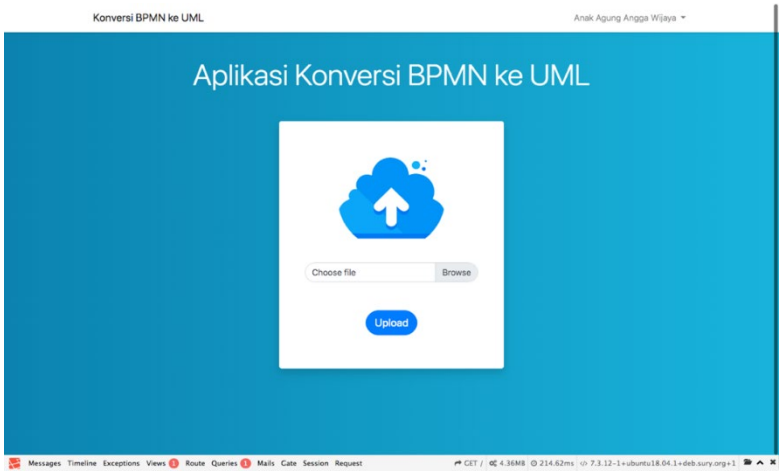
4.1.4.2. Register



Gambar 4.14 Halaman UI *Register*

Halaman *Register* merupakan halaman dimana pengguna dapat mendaftarkan *email* nya yang diperuntukan untuk dapat *login* dan menggunakan aplikasi ini. UI dari halaman *register* dapat dilihat pada gambar 4.15.

4.1.4.3. Upload File

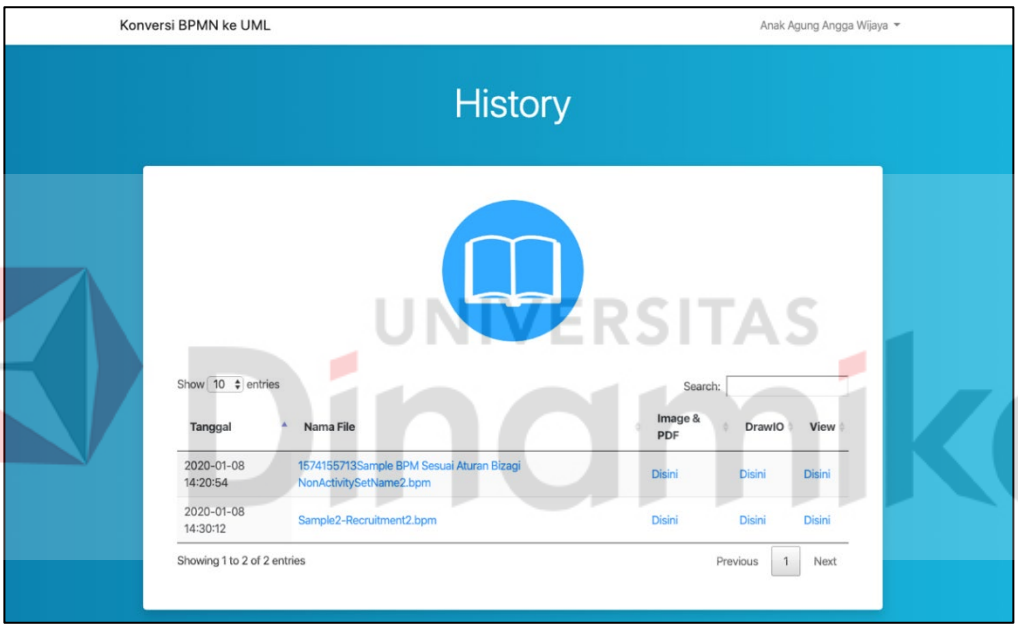


Gambar 4.15 Halaman UI *Upload File*

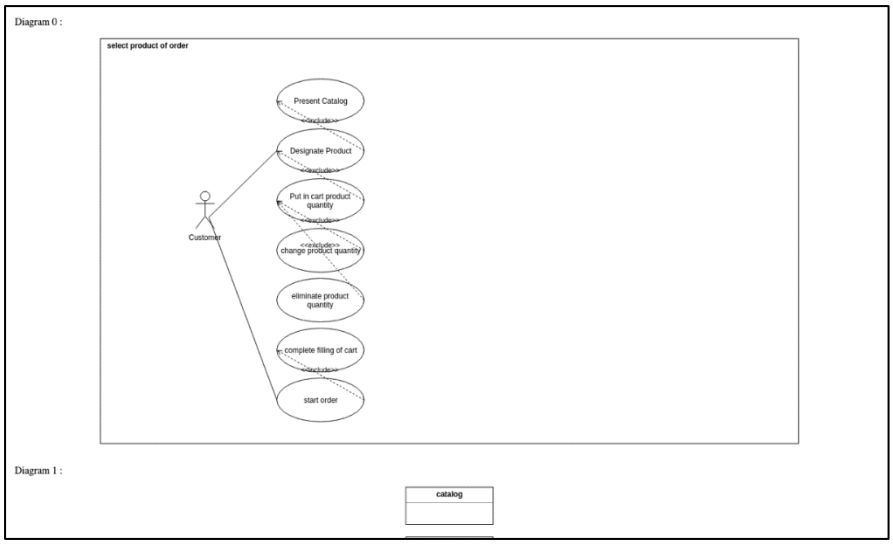
Halaman *Upload File* merupakan halaman dimana pengguna dapat mengupload file BPM yang akan dikonversi kan menjadi UML(DrawIO). UI dari halaman *Upload File* dapat dilihat pada gambar 4.16.

4.1.4.4. History

Halaman *History* merupakan halaman dimana pengguna dapat melihat file bpm dan hasilnya yang telah di unggah sebelumnya.UI dari halaman *History* dapat dilihat pada gambar 4.17.



Gambar 4.16 Halaman UI *History*



Gambar 4.17 Tampilan "*View*" pada Halaman *History*

Ketika pengguna mengklik "Disini" pada kolom View maka akan muncul tampilan yang menampilkan Usecase diagram dan Class diagram dari sample yang sudah dibuat. Tampilan dapat dilihat pada gambar 4.18.

4.2. Pembahasan

Berdasarkan dari data yang didapat pada sub bab 4.1.2 bisa disimpulkan bahwa aplikasi ini sudah sesuai dengan aturan aturan yang dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014).Adapun kekurangan yang dimiliki dari aplikasi ini yang menurut penulis perlu diperbaiki dikemudian hari adalah tata letak *use case* yang tidak rapi.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan *sample* yang dipunya oleh penulis, aplikasi dapat mengkonversi dari BPMN (Bizagi Modeler) ke UML (DrawIO). Hasil tersebut menunjukan bahwa penelitian ini telah berhasil mengimplementasikan Aturan-aturan yang dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014). Tetapi sistem ini menggunakan bantuan dari aturan tambahan untuk menopang proses konversi yang dilakukan oleh sistem. Di *use case* diagram yang dihasilkan oleh sistem ini memiliki kekurangan dimana *use case* yang ditampilkan belum rapi.

5.2. Saran

Sistem ini dapat dikembangkan lebih lanjut dengan menambahkan modul aplikasi yang dapat dibaca menggunakan StarUML karena StarUML menggunakan JSON, agar kode UML dapat langsung di*generate* menjadi kode program bahasa pemrograman dan perlunya mencari aturan yang tambahan terkait relasi agar sistem yang akan dikembangkan agar lebih *solid* lagi.Selain penambahan modul StarUML juga perlu dilakukan perapian *use case* diagram supaya *use case* yang dihasilkan lebih rapi.

Daftar Pustaka

Ramdhani, M. A. (2015). Pemodelan Proses Bisnis Sistem Akademik Menggunakan Pendekatan Business Process Modelling Notation (BPMN) (Studi Kasus Intitusi Perguruan Tinggi XYZ). *Jurnal Informasi*.

Ambler, S. (2012). *Model Storming: An Agile Best Practice*. Diambil kembali dari Agile Modeling: http://agilemodeling.com/essays/modelStorming.htm

Ambler, S. W. (2012). *Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development*. Dipetik Desember 14, 2016, dari http://www.agilemodeling.com/essays/amdd.htm

Arevalo, C., Escalona, M. J., Ramos, I., & Domínguez-Muñoz , M. (2016). A metamodel to integrate business processes time perspective in BPMN 2.0. *Information and Software Technology*, 17-33.

Astamal, R. (2013). *Rancang Bangun Aplikasi Belajar Web Hacking Berbasis Jejaring Sosial (Facebook).* Surabaya: Stikom Surabaya.

Awaludin, R. (2015, January 30). *Medium Laravel Indonesia*. Diambil kembali dari 7 Alasan Menggunakan Framework Laravel dibandingkan native PHP: https://medium.com/laravel-indonesia/7-alasan-menggunakan-framework-laravel-dibandingkan-native-php-89462abd806

B, G. (2019, January 18). *Hostinger*. Diambil kembali dari What is Apache? An In-Depth Overview of Apache Web Server: https://www.hostinger.in/tutorials/what-is-apache

Barjtya, S., Sharma, A., & Rani, U. (2017). A detailed study of Software Development Life Cycle (SDLC) Models. *International Journal Of Engineering And Computer Science*, 22097-22100.

Betari, O., Filali, S., Azzaoui, A., & Amine Boubnad, M. (2018). Applying a Model Driven Architecture Approach: Transforming CIM to PIM Using UML. *International Journal Of Online and Biomedical Engineering*, 171.

Cohen, W. W., Ravikumar, P., & Fienberg, S. E. (2003). A Comparison of String Metrics for Matching Names and Records. *American Association for Artificial Intelligence*.

Datanyze. (2019). *MySQL Market Share*. Diambil kembali dari Datanyze: https://www.datanyze.com/market-share/databases/mysql-market-share

Dharwiyanti, S., & Wahono, R. S. (2003). Pengantar Unified Modeling Language (UML). *Kuliah Umum IlmuKomputer.Com* .

Doan, A., Halevy, A., & Ives, Z. (2012). *Principles of Data.* Waltham: Elsevier.

DreamHost. (2018, Oktober 30). *Web server performance comparison*. Diambil kembali dari DreamHost: https://help.dreamhost.com/hc/en-us/articles/215945987-Web-server-performance-comparison

Dunia Online. (2013, September 16). *Mengenal Jenis-Jenis Website Berdasarkan Fungsinya*. Diambil kembali dari dunia online: http://www.dunia-online.net/article/mengenal_jenisjenis_website_berdasarkan_fungsinya

Dykes, L., & Tittel, E. (2005). *XML For Dummies 4th Edition.* Indianapolis: Wiley Publishing,Inc.

Edy, Ferdiansyah, Prahmusinto, W., & Waluyo, S. (2019). Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order. *Jurnal Rekayasa Sistem dan Teknologi Informasi*, 108.

Firman, A., Wowor, H., & Najoan, X. (2016). Sistem Informasi Perpustakaan Online Berbasis Web . *E-journal Teknik Elektro dan Komputer vol.5 no.2* , 30.

Fong, J. (2018, August 30). *Are Containers Replacing Virtual Machines?* Diambil kembali dari Docker Inc: https://www.docker.com/blog/containers-replacing-virtual-machines/

Gafur, A., Yulianti, S., & Hudayat, N. (2008). *Cara Mundah Mendapatkan Beasiswa.* Jakarta: Penebar Plus.

Grossman, M., Aronson, J., & McCarthy, R. (2005). Does UML make the grade? Insights from the software development community. *Information and Software Technology*, 383-397.

Harfiansyah, I. (2016, July 18). *Mengenal Teknologi Docker*. Diambil kembali dari Codepolitan: https://www.codepolitan.com/mengenal-teknologi-docker

Helmi, A. T., Aknuranda, I., & Saputra, M. C. (2018). Analisis Dan Pemodelan Proses Bisnis Menggunakan Business Process Improvement (BPI) Pada Lembaga Bimbingan Belajar (Studi Kasus: Lembaga Bimbingan Belajar Prisma). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 4186.

Junaedi, M. (2003). *Pengantar XML.* Diambil kembali dari Ilmu Komputer: http://naeli.staff.gunadarma.ac.id/Downloads/files/16859/pengantar+xml.pdf

Kornain, A., Yansen, F., & Tinaliah. (2014). Penerapan Algoritma Jaro-Wwinkler Distance untuk sistem pendeteksi plagiarisme pada dokumen teks berbahasa Indonesia. *STMIK MDP*.

Kurniawati, A., Puspitodjati, S., & Rahman, S. (2014). Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia. *Skripsi Program Studi Sistem Informasi*.

Lee, S. (2012). Unified Modeling Language (UML) for Database Systems and Computer Applications. *International Journal of Database Theory and Application*, 157.

Lucid Software Inc. (2019). *BPMN & BPMN 2.0 Tutorial*. Diambil kembali dari LucidChart: https://www.lucidchart.com/pages/bpmn-bpmn-2.0-tutorial

Rhazali, Y., Mouloudi, A., & Hadi, Y. (2014). Transformation Method CIM to PIM: From Business Processes Models Defined in BPMN to Use Case and Class Models Defined in UML. *World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering*.

Rouse, M. (2018, July). *WhatIs*. Diambil kembali dari Apache HTTP Server: https://whatis.techtarget.com/definition/Apache-HTTP-Server

Sari, S. K., & Tj, A. (2015). Analisis dan Pemodelan Proses Bisnis Prosedur Pelaksanaan Proyek Akhir Sebagai Alat Bantu Identifikasi Kebutuhan Sistem. *Jurnal Informatika,Telekomunikasi dan Elektronika*.

Sholiq, & Robandi, I. (2010). *Analisis dan Perancangan Berorientasi Obyek.* Bandung: Muara Indah.

Sholiq. (2010). *Analisis dan Perancangan Berorientasi Obyek.* Bandung: CV. Muara Indah.

Singh, Y., & Sood, M. (2010). The Impact of the Computational Independent Model for Enterprise Information System Development . *International Journal of Computer Applications*, 22.

Stenberg, D. (2015, September). *Everything-curl*. Diambil kembali dari curl: https://curl.haxx.se/

Sugiarti, Y. (2013). *Analisis dan Perancangan UML Generated VB.6.* Yogyakarta: Graha Ilmu.

Wati, E. F., & Kusumo, A. A. (2016). Penerapan Metode Unified Modeling Language (UML) Berbasis Desktop Pada Sistem Pengolahan Kas Kecil Studi Kasus Pada PT Indo Mada Yasa Tangerang. *UNSIKA Syntax Jurnal Informatika*, 25.

Wei, Z., Mei, H., Zhao, H., & Jie Yang. (2005). Transformation from CIM to PIM: A Feature-Oriented Component-Based Approach. *international conference on Model Driven Engineering Languages and Systems* (hal. 248). Heidelberg: Springer-Verlag Berlin.

Wibowo, W. (2017, April 22). *TDD the Series : Part 1 - Apa itu Test Driven Development (TDD)*. Diambil kembali dari Medium: https://medium.com/koding-kala-weekend/tdd-the-series-part-1-apa-itu-test-driven-development-tdd-ff92c95c945f

Widodo, B. P., & Purnomo, H. D. (2016). Perancangan Aplikasi Pencarian Layanan Kesehatan Berbasis HTML 5 Geolocation. *Jurnal SIstem Komputer*.

Yang, D. (2018). *The 9 Best Programming Languages to Learn in 2018*. Diambil kembali dari fullstack academy: https://www.fullstackacademy.com/blog/nine-best-programming-languages-to-learn-2018

Yaqin, M., Adawiyah, R., Ayu Ningrum, W., & Janan, A. (2019). Simulasi Model Proses Bisnis pada Permainan Travel Agency. *SENIATI 2019.* Malang.

Zabir, O. A. (2008). *Building Web 2.0 Portal With ASP.NET 3.5.* Sebastopol: O'Reilly.