Sistem Konversi Business Process Model Notation ke Unified Modelling Language

Anak Agung Angga Wijaya [1] **Anjik Sukmaaji** [2] **Sholiq** [3] **Teguh Sutanto** [3]
Program Studi/Jurusan Sistem Informasi
Universitas Dinamika
Jl. Raya Kedung Baruk 98 Surabaya, 60298
Email : 1) anggawijaya4567@gmail.com, 2) anjik@dinamika.ac.id, 3) sholiq@is.its.ac.id 3) teguh@dinamika.ac.id

Abstract : *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 Unified Modeling Language (UML). Saat ini tidak ada sistem yang dapat mengotomatasiasi BPMN kedalam simbol pemrograman, yakni UML. Metode konversi tersebut menurut Rhazali dapat dilkakukan 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 sudah dibuat harus sudah mengikuti aturan yang telah dibuat oleh Rhazali. Keluaran sistem ini adalah file DrawIO yang memiliki dua diagram yaitu usecase 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 harus 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 degenerate menjadi kode program Bahasa pemrograman.*

Keywords: BPMN, Website, UML , Konversi, MDA

Business Process Model Notation (BPMN) merupakan suatu cara untuk mengambarkan proses bisnis yang didasarkan pada teknik diagram alur, dirangkai untuk membuat model grafis dari operasi-operasi bisnis yang terdapat aktivitas-aktivitas dan control-kontrol alur yang mendefinisikan urutan kerja (Ramdhani, 2015). Sebelum membuat BPMN harus membuat dan memahami proses bisnis yang ada karena BPMN berfungsi sebagai penjembatan antara perancangan dan implementasi proses bisnis (Helmi, Aknuranda, & Saputra, 2018). Pemodelan proses bisnis dapat digunakan untuk mengidentifikasi kebutuhan sistem sebagai dasar pengembangan sistem informasi (Sari & Tj, 2015).

BPMN tidak bisa dijadikan dasar dari pengembangan suatu aplikasi. Karena berdasarkan MDA (*Model Driven Architecture*) BPMN berada pada level *Computation Independent Model* (CIM) perlu untuk mencapai PIM (*Platform Independent Model*). CIM(*Computation Independent Model*) adalah model yang digunakan oleh manajer bisnis dan análisis untuk mengambarkan proses bisnis. CIM tidak menunjukan detail struktur melainkan menjembatani kedua orang ahli (Betari, Filali, Azzaoui, & Amine Boubnad, 2018) .Berarti ketika seseorang ingin membangun suatu sistem maka perlu dibuat juga UML. Karena UML akan dijadikan rancangan dasar dalam pengembangan sistem (Suendri, 2018).

Unified Modeling Language (UML) telah menjadi standar de facto 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).

Model UML memiliki banyak jenis diagram. UML memliki 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, 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).

Permasalahannya adalah dalam proses konversi dari BPMN ke UML 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).

Solusi dari permasalahan diatas adalah membuat suatu aplikasi yang dapat menkonversi BPMN ke UML. Dengan aplikasi ini dalam proses konversi tidak terpengaruh kualtias dan sudah terstandarisasi karena menggunakan suatu aturan yang sama.

Aturan yang digunakan pada aplikasi ini menggunakan aturan dari (Rhazali, Mouloudi, & Hadi, 2014). Didalam jurnal yang dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014) dijelaskan bagaimana proses membuat proses BPM sampai dengan bagaimana mentransformasikannya menjadi UML (*Class* Diagram dan *Usecase* diagram).

Untuk aturan dalam membangun bpmn akan disebutkan sebagai berikut :

1. Aturan membangun Diagram Kolaborasi BPMN
   a. Menjelaskan secara sederhana dan untuk pembuatan sub proses berkisar 4 sampai 12 *task.*
   b. Jika sub proses kurang dari 4 *task* atau merupakan operasi pelengkap maka akan digabungkan
   c. Hindari penyajian manual
   d. Model hanya menyajikan deskripsi kegiatan yang umum
   e. Fokus pada sub proses dan urutannya
   f. Identifikasi Aktor yang terlibat
   g. Hindari pengunaan *Gateway*.
2. Aturan untuk membangun Diagram Proses BPMN
   a. Detailkan sub proses menjadi beberapa *task*
   b. Jangan mewakili tugas manual dari diagram kolaborasi bpmn
   c. Memiliki *Gateway*
   d. Menambahkan *dataobject* pada setiap *task*

Sedangkan untuk aturan dalam melakukan transformasi dari BPMN ke UML akan disebutkan sebagai berikut :

1. Aturan CIM ke Use Case
   a. Setiap tugas dalam proses level akan dijadikan menjadi usecase
   b. Pada kolaborasi level bpm, *task* yang berada dalam suatu maka *pool* tersebut akan menjadi aktor *use case*
   c. Jika terdapat gerbang eksklusif diantara maka akan menjadi "*extend*" pada *usecase*
   d. Jika terdapat tugas yang berurutan maka akan menjadi "*include*" pada *usecase*
   e. Dalam menerjemahkan harus berurutan dan tidak boleh berjalan mundur
   f. Setiap *task* yang ada di kolaborasi *level* akan diterjemahkan sebagai *package*
2. Aturan dari Process Level ke Class
   a. Dalam proses level, *dataobject* akan diubah menjadi *classes*
   b. Jika ada *dataobject* yang memiliki nama sama maka hanya akan dijadikan satu *class* saja

Tetapi didalam jurnal yang dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014) masih di Bahasa ATL (*Atlas Transformation Language*). ATL ini belum bisa digunakan seperti aplikasi pada umumnya.

Pada jurnal ini, penulis menggunakan Bizagi Modeler sebagai aplikasi pembuat BPM (Input) dan menggunakan DrawIO sebagai keluaran untuk menvisualisasikan UML.Untuk DrawIO dan Bizagi Modeler menggunakan XML sebagai penyimpanan obyek.

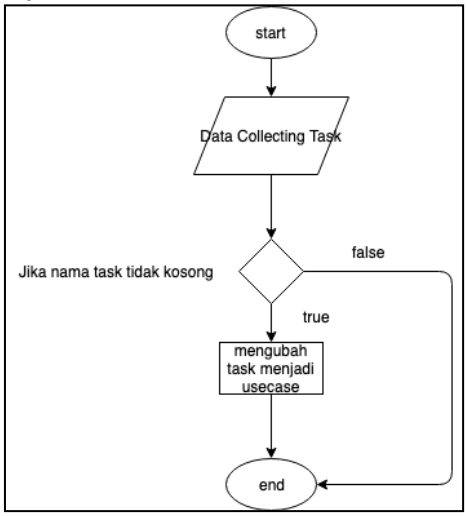
Data yang didapatkan dari Bizagi Modeler berupa XML. Penulis memutuskan untuk melakukan konversi berupa JSON. Karena JSON dapat di-*parsing* dan diperlukan sebagai *array*.

Hasil dari aplikasi ini adalah dapat mengeluarkan UML dengan memanfaatkan DrawIO. DrawIO ini adalah *platform* independen yang berfungsi menvisualisasikan XML. UML yang dibangun didasarkan dengan aturan-aturan (Rhazali, Mouloudi, & Hadi, 2014)

dan tambahan aturan dari penulis, sehingga aplikasi ini bisa terstandar.

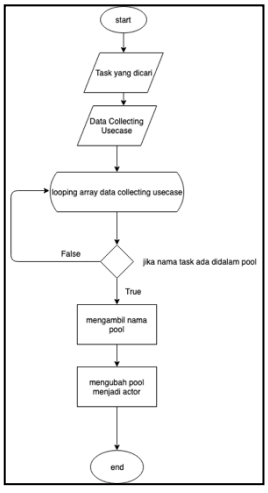
Sebelum penulis membuat sistem ini, penulis berinisiatif untuk membuat *flowchart* terlebih dahulu agar lebih mudah memahami konsep dari aturan (Rhazali, Mouloudi, & Hadi, 2014)

a. Setiap tugas dalam proses level akan d



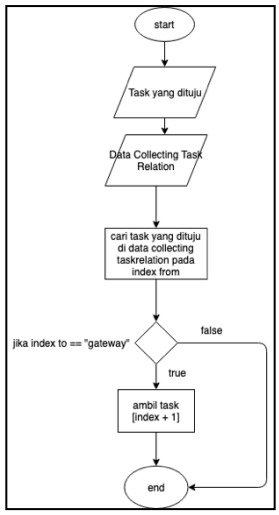
Gambar 3 : *Flowchart Task to Usecase*

b. Pada kolaborasi level bpm, *task* yang berdada dalam suatu *pool* maka *pool* tersebut akan menjadi aktor *usecase*



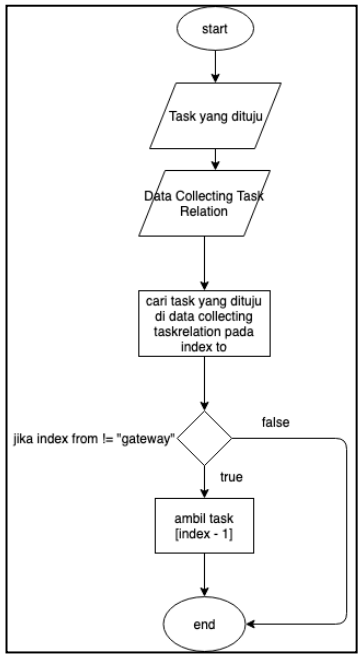
Gambar 1 : *Flowchart Pool to Usecase*

c. Jika terdapat gerbang eksklusif diantara maka akan menjadi "*extend*" pada *usecase*



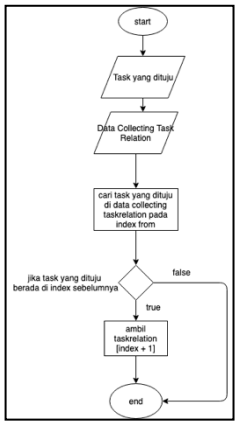
Gambar 2 : *Flowchart Gateway to Extend*

d. Jika terdapat tugas yang berurutan maka akan menjadi "*include*" pada *usecase*



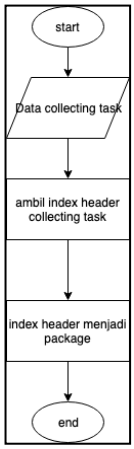
Gambar 4 : *Flowchart Sequence to Include*

e. Dalam menerjemahkan harus berurutan dan tidak boleh berjalan mundur



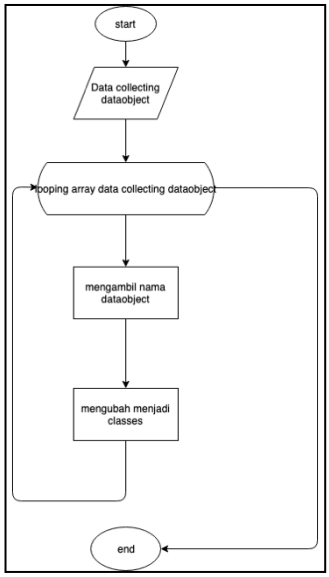
Gambar 5 : *Flowchart No Turn Back*

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



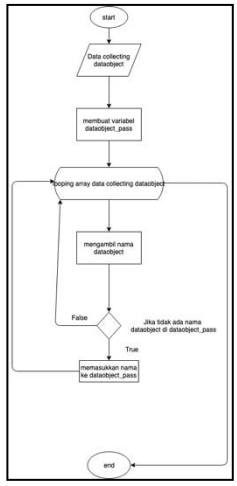
Gambar 6 : *Flowchart Task to Package*

g. Dalam proses level, *dataobject* akan diubah menjadi *clases*



Gambar 7 : *Flowchart Dataobject to Classes*

h. Jika ada *dataobject* yang memiliki nama sama maka hanya akan dijadikan satu *class* saja

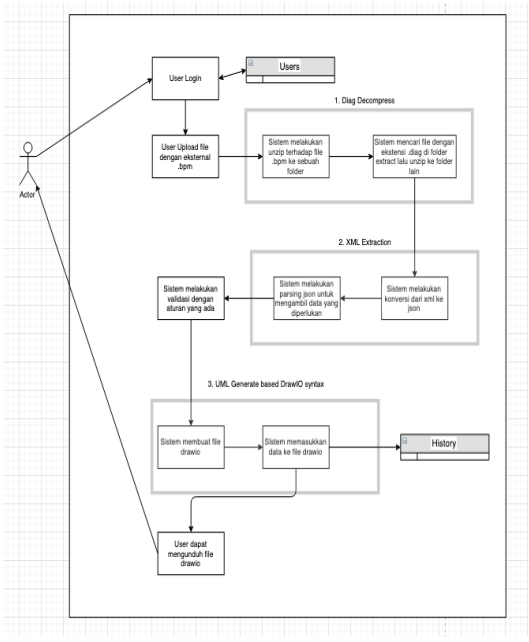


Gambar 8 : *Flowchart Clean Dataobject*

METODE

Didalam sistem aplikasi konversi BPMN ke UML melalui beberapa tahapan.Tahapan yang dilalui melalui 3 area

besar.3 area besar itu adalah *Diag decompress, XML Extraction,* dan *UML generate based DrawIO syntax.* Untuk tabel yang digunakan ada dua, yaitu tabel *users* dan tabel *histories*. Tabel *users* digunakan untuk menyimpan *credential* pengguna dan tabel *histories* digunakan untuk menyimpan *file* bpm dan hasil konversinya yang sudah pernah dilakukan oleh pengguna. Tahapan yang dilakukan oleh sistem ini dapat dilihat pada gambar 9.



Gambar 9 : Konversi BPMN ke UML

Diag Decompress

Diag Decompress merupakan suatu fase yang bertujuan untuk mengeluarkan *file-file* yang berada di dalam *file* bpm yang telah di*upload* oleh pengguna.Fase ini merupakan langkah awal dari aplikasi ini karena fase ini bertugas untuk mengolah *file* bpm menjadi diagram.xml.

Di diagram.xml memiliki data-data yang berguna untuk melakukan konversi ke UML.Untuk mendapatkan *file* diagram.xml ini sistem melakukan dua kali *unzip* yaitu *unzip* file bpm untuk mendapatkan *file* dengan ekstensi .diag. setelah *file* dengan ekstensi .diag didapatkan lalu dilakukan *unzip* lagi untuk mendapatkan *file* diagram.xml.

Jika di BPMN memiliki dua diagram maka terdapat juga dua *file* diag.Di satu *file* diag juga terdapat satu *file* diagram.xml.Jadi satu diagram di Bizagi Modeler diwakili oleh satu diagram.xml.

Setelah *file* diagram.xml yang didapat dari proses *diag decompression*. Langkah selanjutnya adalah melakukan *xml extraction*.

XML Extraction

XML Extraction merupakan suatu fase yang bertujuan untuk mendapatkan data-data yang berasal dari *file* bpm yang telah di*upload* oleh pengguna.Fase ini merupakan langkah kedua dari aplikasi ini.

Didalam fase ini, sistem melakukan pengolahan data dari diagram.xml. Agar sistem dapat membaca dan mengambil data dari diagram.xml. Langkah yang diambil oleh penulis adalah melakukan konversi dari XML menjadi JSON. Penulis beranggapan bahwa JSON lebih mudah digunakan karena JSON dapat diperlakukan sama seperti array.

Data yang diperoleh dari hasil konversi XML ke JSON tergolong cukup banyak. Data yang diambil akhirnya ada 4 yaitu Data *Task, TaskRelation, Usecase,* dan *Dataobject.* Data *Task* akan digunakan untuk menampilkan *usecase* di *Usecase* diagram, Data *TaskRelation* digunakan untuk menentukan relasi antar *usecase*, Data *dataobject* digunakan untuk menampilkan *classes* pada *class* diagram, dan Data *Usecase* digunakan untuk mengetahui aktor yang terlibat dalam suatu *task*.

Semua data yang diperoleh diatas dari proses iterasi dikarenakan data yang didapat berupa array. Data yang sudah didapat tadi disimpan pada attribut kelas "*Files*" sebelum nanti digunakan oleh kelas lain.

Data yang sudah didapat melalui proses "*Diag Decompress*" akan di pindahkan ke kelas *"Validation"*. Didalam kelas "*Validation*" terjadi proses *cleansing*. Yang dimaksud proses *cleansing* disini adalah melakukan pembersihan terhadap *dataobject* yang memiliki nama yang sama, penghapusan *non-task* yang ada didalam data *task*, dan *merger array* di data *usecase.*

UML Generate Based DrawIO Syntax

UML Generated Based DrawIO Syntax merupakan suatu fase yang bertujuan untuk memasukkan data yang diperoleh dari proses sebelumnya dan dimasukkan ke dalam *file* XML.Fase ini merupakan langkah terakhir dari aplikasi ini.

Didalam fase ini, sistem melakukan *generate* XML yang datanya berasal dari Kelas

"*Files*" dan "*Validation*". Sebelum memasuki pada fase ini, sistem perlu melalui "*Files*" dan "*Validation*" karena data perlu dipastikan sudah sesuai kebutuhan dan sesuai dengan aturan dari (Rhazali, Mouloudi, & Hadi, 2014) dan aturan tambahan yang dibuat oleh penulis.

Didalam fase "*Create File*" ini melakukan *crafting string* yang sesuai dengan format dari DrawIO. Format dari DrawIO terdiri dari 3 bagian. 3 bagian yang ada didalam di DrawIO adalah *header,content,* dan *footer.*

Tahap *header* ini adalah melakukan *crafting string*. Di *header* menyimpan data-data yang nanti nya juga akan digunakan pada bagian *content.*Data-data yang dimaksud adalah perangkat yang digunakan untuk akses, versi drawio, dan jumlah halaman.

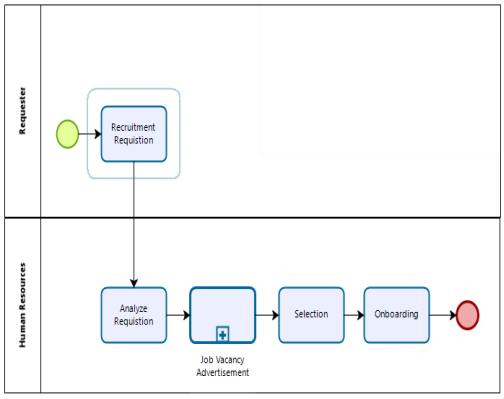
Tahap *content* ini adalah melakukan *crafting string*. Di *content* menyimpan data-data diagram UML untuk nantinya ditampilkan di DrawIO. Karena aplikasi ini menghasilkan keluaran UML (Usecase diagram & Class Diagram) maka pada DrawIO dapat dipastikan total halaman akan genap. Diagram dengan halaman ganjil adalah *Usecase diagram* dan halaman genap adalah *Class diagram.*

Didalam tahap ini ada dua *method* yaitu classGen dan usecaseGen. *Method* classGen berfungsi untuk men*generate class* diagram yang sesuai dengan sintaks DrawIO dan didasarkan pada data yang didapat dari "*collecting dataobject*". Pada akhir ketika sudah dihasilkan *crafted string* maka akan di *encode* melalui tiga tahap. Tiga tahap yaitu encodeURIComponent, gzdeflate, dan base64encode.

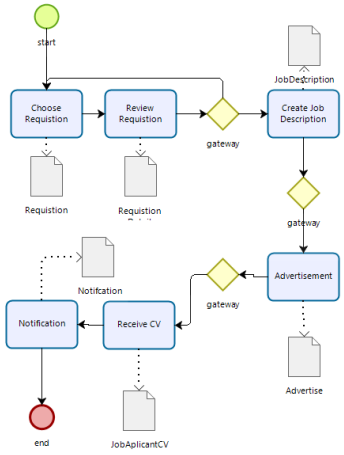
Method usecaseGen berfungsi untuk men*generate usecase* diagram yang sesuai dengan sintaks DrawIO dan didasarkan pada data yang didapat dari "*collecting task*", "*collecting usecase*", "*collecting taskrelastion*".Didalam *method* usecaseGen masih terdapat dua method yang dibutuhkan yaitu firstStageUsecase dan scndStageUsecase. *Method* firstStageUsecase berfungsi untuk mendefiniskan *task* mana saja yang memiliki relasi asosiasi ke aktor. Dan *method* scndStageUsecase berfungsi untuk mendefinisikan *task* mana saja yang memiliki relasi *extend* atau *include* ke *task* yang tergolong firstStageUsecase. Pada akhir ketika sudah dihasilkan *crafted string* maka akan di *encode* melalui tiga tahap. Tiga tahap yaitu encodeURIComponent, gzdeflate, dan base64encode.

Tahap *footer* ini adalah melakukan *crafting string*. Di *footer* bertujuan untuk menutup *tag* mxfiles. Hal ini diperlukan karena berdasarkan konsep XML semua *tag* perlu ditutup kembali.

HASIL DAN PEMBAHASAN



Gambar 10 : *Collaboration Diagram Recruitment*



Gambar 11 : *Process Diagram Job Vacancy Advertisement*

BPMN diatas menjelaskan tentang alur proses bisnis recruitment. Proses bisnis ini menjelaskan dari tahap pengajuan penambahan pegawai hingga penempatan. Proses bisnis ini diawali oleh *Requester* yang nanti nya akan diteruskan ke *Human Resources.*Di *human resources* sendiri melalui beberapa *task*. *Task-*

task yang dilalui oleh *Analyze Requistion, Job Vacancy Advertisement, Selection,* dan *Onboarding.* Tetapi pada *collaboration diagram* tidak akan membahas secara detail. Di dalam *collaboration diagram* hanya akan membahas garis besar dari suatu proses bisnis. Untuk proses keseluruhan dapat dilihat pada gambar 10.

Process Level BPM menjelaskan tentang alur proses bisnis dari *Job Vacancy Advertisement.* Proses bisnis ini menjelaskan dari tahap *choose requisition* sampai *notification*. Proses bisnis ini diawali oleh *Human resources* yang memilih *requisition dan diakhir di notification.* Di *process diagram* akan membahas secara detail. Di dalam *process diagram* membahas cukup detail bahkan terdapat *gateway* yang menunjukan terdapat proses yang bercabang. Untuk proses keseluruhan dapat dilihat pada gambar 13.

Sampel ini nantinya akan diupload ke aplikasi. Aplikasi ini sebenarnya terbagi kedalam 3 area besar, yaitu *Diag decompress, XML Extraction,* dan *UML generate based DrawIO syntax*. Proses keseluruhan dapat dilihat pada gambar 1.

Diag Decompress

Pada tahap *Unzip* file bpm mengalami dua kali. Pertama untuk mendapatkan file dengan ekstensi .diag dan yang kedua untuk mendapatkan file "Diagram.xml". Setelah "Diagram.xml" didapatkan maka akan dikonversi keJSON agar lebih mudah dibaca dan diakses dalam bentuk array.

Berdasar data yang diperoleh dari "Diagram.xml" maka bisa dilanjutkan pada tahap kedua yaitu "Extract Data". Pada tahap ini JSON tadi akan di *parsing* untuk mendapatkan data yang diperlukan seperti *Task, TaskRelation, DataObject,* dan Usecase.

XML Extraction

Pada proses "*collecting task*" aplikasi ini berfokus pada *array* "WorkFlow Activity". Di array tersebut tersimpan semua *task* yang berada di satu "Diagram.xml". Hasil proses ini dapat dilihat pada gambar 12.

```
Array
(
    [0] => Array
        (
            [Job Vacancy  Advertisement] => Array
                (
                    [0] => start
                    [1] => Choose Requistion
                    [2] => Review Requistion
                    [3] => gateway
                    [4] => Create Job Description
                    [5] => gateway
                    [6] => Advertisement
                    [7] => gateway
                    [8] => Receive CV
                    [9] => Notification
                    [10] => end
                )

        )

)
```

Gambar 12 *Result Collecting Task*

Pada proses "*collecting taskrelation*" aplikasi ini berfokus pada *array* "WorkFlow Transition". Di array tersebut tersimpan semua Relasi dari satu task ke task lain yang berada di satu "Diagram.xml". Hasil proses ini dapat dilihat pada gambar 11.

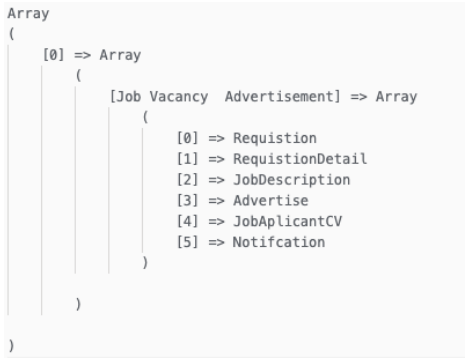
```
Array
(
    [0] => Array
        (
            [from] => start
            [to] => Choose Requistion
            [diagram] => Job Vacancy  Advertisement
        )

    [1] => Array
        (
            [from] => Choose Requistion
            [to] => Review Requistion
            [diagram] => Job Vacancy  Advertisement
        )

    [2] => Array
        (
            [from] => Review Requistion
            [to] => gateway
            [diagram] => Job Vacancy  Advertisement
        )
```

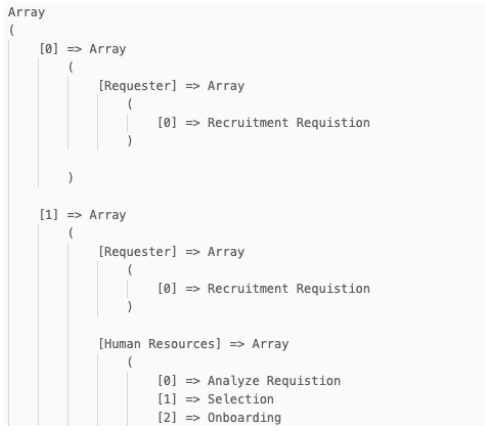
Gambar 13 : *Result Collecting TaskRelation*

Pada proses "*collecting dataobject*" aplikasi ini berfokus pada *array* "WorkFlow DataObject". Di array tersebut tersimpan semua *dataobject* yang berada di satu "Diagram.xml". Hasil proses ini dapat dilihat pada gambar 14.



Gambar 14 : *Result Collecting Dataobject*

Dan yang terakhir dalam tahap "extract data" yaitu "Collecting usecase". Tahap ini bertujuan untuk mengambil actor yang terlibat dalam *task.*Tahap ini berfokus pada array "WorkFlow @attribute". Hasil pengambilan data ini dapat dilihat pada gambar 15.



Gambar 15 : *Result Collecting Usecase*

Setelah data yang didapatkan tadi lalu akan di bersihkan yaitu memastikan dataobject tidak boleh ganda dan task yang ada harus bersih dari "start", "gateway", dan "end".

UML Generate based DrawIO Syntax

Pada tahap *Create* file adalah suat tahapan yang menerjemahkan dari data yang didapat menjadi DrawIO. Dalam hal ini karena DrawIO menggunakan XML maka aplikasi harus dapat melakukan *crafting string* agar cocok dengan sintaksis yang diterima oleh DrawIO.

Didalam file DrawIO terbagi menjadi tiga bagian. Tiga bagian itu adalah *header*, *content*, dan *footer*. Pada bagian Men*generate* header disini berfungsi untuk mendefinisikan dimana file drawio dibuat, diakses dengan perangkat apa, tanggal dibuat, versi drawio yang digunakan, dan jumlah halaman yang ada . Ketika sudah di*generate* oleh aplikasi , *crafted string* tadi akan dimasukkan file. Hasil dari Generate dapat dilihat pada 16.

Gambar 16 : *Result Generate Header*

Mengenerate diagram disini berfungsi untuk menampilkan data-data yang sudah didapat dari tahap *Collecting Data*. Dalam mengenerate diagram, ada dua *method* yaitu classGen dan usecaseGen. *Method* classGen berfungsi untuk men*generate* class diagram yang sesuai dengan sintaks DrawIO dan didasarkan pada data yang didapat dari "collecting dataobject".

Gambar 17 : *Result Class Diagram* belum di *encode*

Tetapi pada gambar 17 belum bisa digunakan oleh DrawIO Dikarena kan harus di enskripsi melalui tiga tahap, yaitu


```
7ZhRb5swEMc/DY9IYKckfQyhXZVpm9RK214dOMCSwdSYhOzTzwYTIF03VVMlV+Ilwf873/nOP1sJD
9aOPfCarPphUzQpJf8UXiiv8CInlEbmkA9c5ScM0mruRjzsoRYzjQiBD/N3VLO5lkrksEL4Skm7KX
00TCdw7eCc5l/1S0O2C6d0NfGu/bo+vWoVjv97T6ST6vIXVNMAfdv2XWpQoBpfyv6OQv046ENaZtD
01sUlopUsYzcy8WFUCYoypnjLz3WU+DMIjPDe0lpSXg0mVd7h2V1p1reVCr35Ackjkv54zocc/1ps
RikehZXXinrSEup2jwcpryU5mT5yIzvSUGZPpMPwI6go+p1yoJpp0tunRDaq5PwD4Z8w7beOH0tAC
+xaK/Y9BcQSSULawbCXL/o0lMCObYd7zQwR1LGi1XMvWooywJShjm1HeJt3u17BQbCXF2LeE4pXNF
HF+rdLbJuyl89xs=
```


Gambar 18 : *Result Class Diagram* sudah di *encode*

encodeURIComponent, *gzdeflate*, dan *base64*. Ketika sudah melalui 3 hal tersebut maka hasilnya dapat dilihat pada gambar 10 dan sudah sesuai standar DrawIO.

Method usecaseGen berfungsi untuk men*generate usecase* diagram yang sesuai dengan sintaks DrawIO dan didasarkan pada data yang didapat dari "*collecting task*", "*collecting usecase*", "*collecting taskrelastion*".Didalam *method* usecaseGen masih terdapat dua method yang dibutuhkan yaitu firstStageUsecase dan scndStageUsecase. *Method* firstStageUsecase berfungsi untuk mendefiniskan *task* mana saja yang memiliki relasi asosiasi ke aktor. Dan *method* scndStageUsecase berfungsi untuk mendefinisikan *task* mana saja yang memiliki relasi *extend* atau *include* ke *task* yang tergolong firstStageUsecase.Hasil dari *method* dan telah melewati 3 tahap *encode* dapat dilihat pada gambar 18.

Dan yang terakhir adalah men*generate footer*. Langkah ini sebenernya adalah langkah

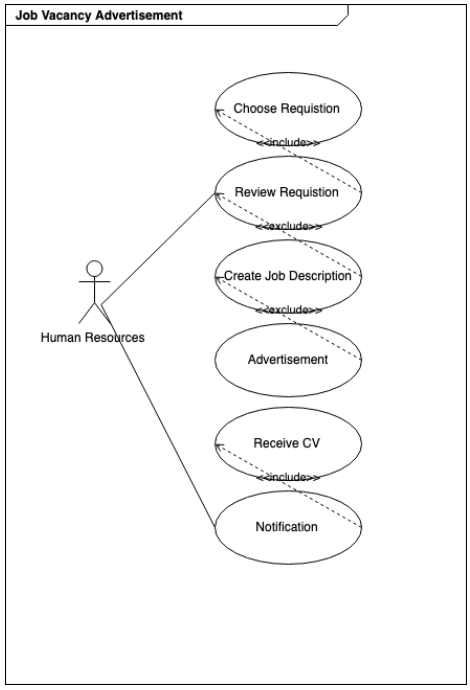

```
7VnLkps4FP0alu0CxKO9tN151FQy6UlXdTKzk+EaNA2IBmHjfH0kkHjbcTr98MJe2OhIukLnnoMkr
wR36ABHWJFsSHvNeQURoxkvZBjyYJeKyH4Syju36zDY36o6Y4gBFw5+FIoTO7xb8Rn4USN5x5W/ER
k71Znt1uxpahiyOeMHgl3W1Zr4/HMI4KUSKM0hYb1AZdSqk45q+41gW2swNWNtXcowtjgo1hulEPN
```


Gambar 21 : Result *Usecase diagram* sudah di *encode*

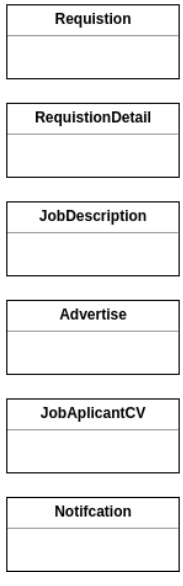

```
Yb1B3Mt1eo4EZuYZEG1PLBwsoUTFWVN+S86cLbt6QLZCdav/s6Ps+gwoc0eU/U0Z2TDF8hyd6Z/Do
f4kQHYkXZy6QS/d0ScTHYnmfZELhWWQcRFue2fOKcS3A5yS4l4VCo52caBh5aKUs9e9hof/4bhnMG
K/Y7cJiUAyij9E1iHzfJqxsCnWgI80mGMJxnU//c0Wd652P2xHuNfuGFI0YQbzx+J/TQFy9jisl3e
```


terakhir dan paling mudah karena dilangkah ini hanya memambahkan penutup *tag* dari mxfiles. Setelah tahap-tahap diatas sudah selesai maka hasil konversi pun sudah dapat dilihat. Hasil dari konversi dapat dilihat pada gambar 19 dan gambar 21.

Pada gambar 19 merupakan hasil dari gambar 10 dan gambar 13. Gambar 19 didapat dari *collecting task, collecting taskrelation,* dan *collecting usecase.review requisition* ini mempunyai relasi sekuensial dari *choose requisition* jadi *review requisition* memiliki relasi *include* ke *choose requisition*. *Review requisition* juga memiliki relasi ke *gateway* yang berhulu ke *create job description* maka kedua *usecase* itu memiliki relasi *extend.*



Gambar 20 : *Result Usecase Diagram*



Pada gambar 21 adalah hasil dari konversi yang keluaran nya adalah *class* diagram.*Class* diagram ini didapat dari *collecting*

Gambar 19 : Result *Class Diagram*

dataobject.Hasil dari *collecting dataobject* dapat dilihat pada gambar 14 untuk melihat kecocokan nya.

SIMPULAN

Berdasarkan dari gambar 19 dan gambar 20 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) dan menggunakan aturan tambahan dari penulis untuk membantu menyelesaikan masalah dari aplikasi ini.

RUJUKAN

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. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 1467-1471.

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.

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* (pág. 248). Heidelberg: Springer-Verlag Berlin.

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

Sari, S. K., & Tj, A. (2015). Analisis dan Pemodelan Proses Bisnis Prosedur Pelaksanaan Proyek Akhir Sebagai Alat Bantu Identifikasi Kebutuhan Sistem. *Jurnal Infotel*, 143-152.

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*, 4184-4191.

Suendri. (2018). Implementasi Diagram UML (Unified Modelling Language) Pada Perancangan Sistem Informasi Remunerasi Dosen Dengan Database Oracle (Studi Kasus: UIN Sumatera Utara Medan). *Jurnal Ilmu Komputer dan Informatika*, 1-9.

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

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

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.