

Sistem Konversi Business Process Model Notation ke Unified Modelling Language

by Anak Agung Angga Wijaya

Submission date: 24-Jan-2020 03:17PM (UTC+0700)

Submission ID: 1245798569

File name: Jurnal-16410100123-Draft-15_Januari_2020-Rev1-Rev2.docx (628.34K)

Word count: 2930

Character count: 17977

Sistem Konversi Business Process Model Notation ke Unified Modelling Language

Anak Agung Angga Wijaya¹⁾ Anjik Sukmaaji²⁾ Sholiq³⁾ Teguh Sutanto³⁾

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@its.ac.id 3) teguh@dinamika.ac.id

Abstract : In the concept of Model Driven Architecture (MDA), There is an Computation Independent Model or known as BPMN and Platform Independent Model or known as UML. In the process of conversion from BPMN to UML is still done manually and it is feared that the quality is not maintained. Because the quality of the results of the conversion depends on who is doing it. This research aims to build a website that can convert it automatically. The process of conversion from BPMN to UML using Business Process Modelling (BPM) data with the Bizagi Modeler format as input. Before becoming UML, the data is converted to DrawIO with XML type. From DrawIO, UML can be visualized. The process of visualization refers to rules made by Rhazali. The system created can visualize UML from BPM data with BPM input must comply with rules of Thazali, the output will also follow from the above rules and DrawIO is actually a standalone platform but still be displayed in PNG format.

Keywords: BPMN, Website, UML, Conversion, MDA

Business Process Model Notation (BPMN) merupakan suatu cara untuk menggambarkan 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 memahami proses bisnis yang ada karena BPMN berfungsi sebagai penjematan 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 analisis untuk menggambarkan proses bisnis. CIM tidak menunjukkan 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 (Sendri, 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 memiliki 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 bilang 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). Atau pun dalam prosesnya tidak menggambarkan

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 kualitas 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 penggunaan *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 memvisualisasikan UML. Untuk DrawIO dan Bizagi Modeler menggunakan XML sebagai penyimpanan obyek.

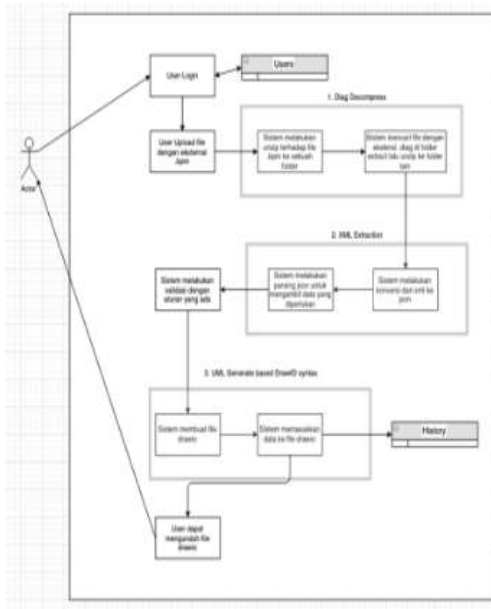
Data yang didapatkan dari Bizagi Modeler berupa XML. Penulis memutuskan untuk melakukan konversi berupa JSON. Karena JSON dapat di-parsing dan diperlukan sebagai *array*.

Hasil dari aplikasi ini adalah dapat mengeluarkan UML dengan memanfaatkan DrawIO. DrawIO ini adalah *platform* independen yang berfungsi memvisualisasikan XML. UML yang dibangun didasarkan dengan aturan-aturan (Rhazali, Mouloudi, & Hadi, 2014) dan tambahan aturan dari penulis, sehingga aplikasi ini bisa terstandar.

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 pengguna¹⁹. Tahapan yang dilakukan oleh sistem ini dapat dilihat pada gambar 1.



Gambar 1 : 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 diupload 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 diupload 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 atribut 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 ganjil adalah *Class diagram*.

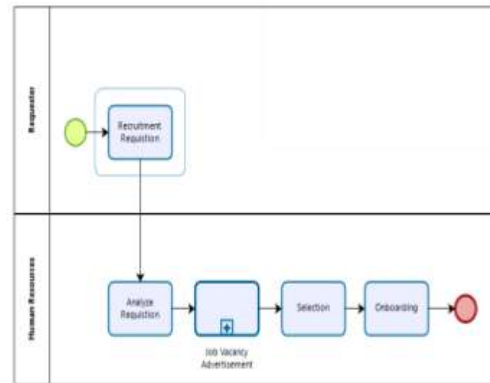
Didalam tahap ini ada dua *method* yaitu *classGen* dan *usecaseGen*. *Method classGen* berfungsi untuk *generate class* diagram yang sesuai dengan sintaks DrawIO dan didasarkan pada data yang didapat dari “*collecting dataobject*”. Pada akhir ketika sudah dihasilkan *crafted string* maka akan di *encode*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

Method usecaseGen berfungsi untuk *generate usecase* diagram yang sesuai dengan sintaks DrawIO dan didasarkan pada data yang didapat dari “*collecting task*”, “*collecting usecase*”, “*collecting taskrelation*”. Didalam *method usecaseGen* masih terdapat dua *method* yang dibutuhkan yaitu *firstStageUsecase* dan *scndStageUsecase*. *Method firstStageUsecase* berfungsi untuk mendefinisikan *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.

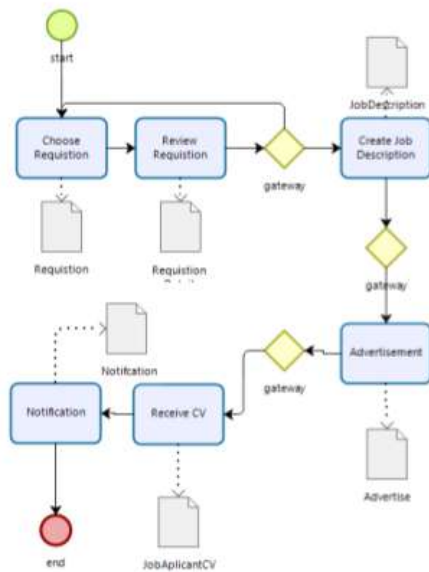
HASIL DAN PEMBAHASAN



Gambar 2 : Collaboration Diagram Recruitment

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 Requisition*, *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 2.

BPMN diatas 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 menunjukkan terdapat proses yang bercabang. Untuk proses keseluruhan dapat dilihat pada gambar 3.



Gambar 5 : Process Diagram Job Vacancy Advertisement

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 5.



Gambar 4 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 4.



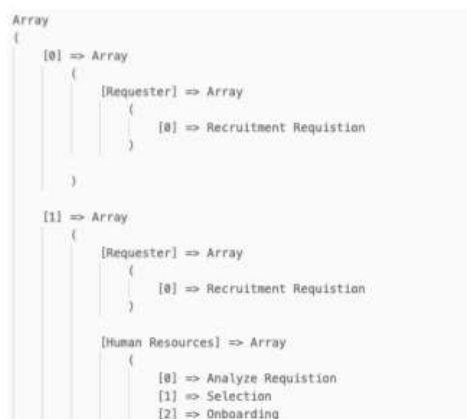
Gambar 3 : Result Collecting TaskRelation

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 6.



Gambar 6 : 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 `21` `orkFlow @attribute`. Hasil pengambilan data ini dapat dilihat pada gambar 7.



Gambar 7 : 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 suatu 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 *Mengenerate 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 digenerate oleh aplikasi, *crafted string* tadi akan dimasukkan file. Hasil dari *Generate* dapat dilihat pada 8.

```
<mxfile host="localhost" modified="2019-10-24T12:58:01.452Z" agent="directory"
```

Gambar 8 : 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 mengenerate class diagram yang sesuai dengan sintaks DrawIO dan didasarkan pada data yang didapat dari “collecting dataobject”.

```
<mxGraphModel dx="786" dy="488" gridSize="10" guides="1" tooltips="
<root>
<mxCell id="u000--sbr733ipka7ef-0" />
<mxCell id="u000--sbr733ipka7ef-1" parent="u000--sbr733ipka7ef-0" />
<mxCell id="u000--sbr733ipka7ef-8" value="<tip style=<quot;:margin
<mxGeometry x="148" y="78" width="148" height="68" as="geometry"
</mxCell>
<mxCell id="u000--sbr733ipka7ef-1" value="<tip style=<quot;:margin
<mxGeometry x="148" y="158" width="148" height="68" as="geometry"
</mxCell>
<mxCell id="u000--sbr733ipka7ef-2" value="<tip style=<quot;:margin
<mxGeometry x="148" y="238" width="148" height="68" as="geometry"
</mxCell>
<mxCell id="u000--sbr733ipka7ef-3" value="<tip style=<quot;:margin
<mxGeometry x="148" y="318" width="148" height="68" as="geometry"
</mxCell>
<mxCell id="u000--sbr733ipka7ef-4" value="<tip style=<quot;:margin
<mxGeometry x="148" y="398" width="148" height="68" as="geometry"
</mxCell>
<mxCell id="u000--sbr733ipka7ef-5" value="<tip style=<quot;:margin
<mxGeometry x="148" y="478" width="148" height="68" as="geometry"
</mxCell>
</root>
</mxGraphModel>
```

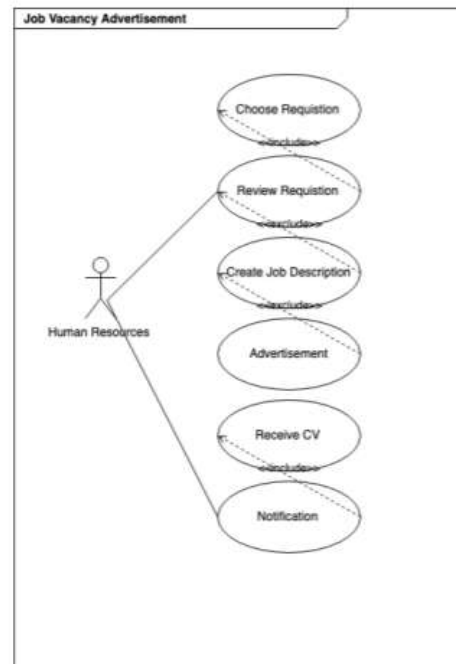
Gambar 9 : Result Class Diagram belum di encode

Tetapi pada gambar 9 belum bisa digunakan oleh DrawIO. Dikarena kan harus di enkripsi melalui tiga tahap, yaitu *encodeURIComponent*, *gzdeflate*, dan *base64*. Ketika sudah melalui 3 hal tersebut maka hasilnya dapat dilihat pada gambar 10 dan sudah sesuai standar DrawIO.

7Zn8s5wPmc/DY91YKckf0yKZVpn9K214d0MCsWdSYh0zTzwYTF83VM1V+I1wF873/n0P1s3C
9a0PFCarPphUz0pJ7BUX11v8C1n1EbkaA9c55cM8erUj)zsoRYzjQ180/NSVL051krksEL45ka70
00TCdW7eCc51/15002C6d0NfGu/ba+vWvVjv97T65T6vIXVNPafdvZ0p008pTyv60Qv046Eha2t1
01sUlopUsYzcy0WfUCYoypnjLz3WU+0M1jPDe0lp5Xg8mVd7hZV1p1reVCr35Ackjv54zoccc/3ps
RLkehZXXInrSEup2jwcpYU5eT5yIzvSUGZPpMPw16go+p1yoJpp8tun8Daq5Pw04Z8w7be0M8TAC
+xak/Y98cQ5SULawCXL/o0UNC0bYd7z0wRLG1LDMVWooywJ5hje1HeJt3u17B0bCXF2Le4Ep0B
HF+rdLbJuy189xs=

Gambar 10 : Result Class Diagram sudah di encode

Method usecaseGen berfungsi untuk mengenerate usecase diagram yang sesuai dengan sintaks DrawIO dan didasarkan pada data yang didapat dari "collecting task", "collecting usecase", "collecting taskrelation". Didalam method usecaseGen masih terdapat dua method yang dibutuhkan yaitu firstStageUsecase dan scndStageUsecase. Method firstStageUsecase berfungsi untuk mendefinisikan 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 [12](#) method dan telah melewati 3 tahap encode dapat dilihat pada gambar 11.



Gambar 11 : Result Usecase Diagram

7VnLaps4FP0a1u0Ck09tN151FQy6U1XDTKzK+EaNA21BmHj fH0kktj)bcT98MJe20hTukLno0Kz
wR36ABHw3FsShWdeURoxkvZBjyY2ekYH45yju36zDY36oEY4g8Pw5+7ToT07xbRn4U5H5a5M/ER
k71Znt1uxpah.iyoHig13W1Zr4/HPI4KUSKPHYb1A2d5qk45q+41gW2svMMNTXcovt jgo3hu1EPN
A/LZ1VbAXSRf6vYfWen351cLfw5Z1znEgg3Zh0xHkbgWp1A41770Ysow+HrCokxKLoF5e80WpS
+brxepPg7ZY00KPh05P1dGe0Nak0Wv0ES0d0g86AxsEzwK8sgw617YWe02ee23X0pa0Kdrzn2
f6Z8EgTe0pw0gKra0YjRkzy+Z5oRk)nmHTT0xFjTr1Qu00/UpubbcvZcL29Jhu08T0Kj2R11jR
Yb1B3H1Eop4E2uYZEG1PCBws0fFWN+S66cCb16QLZCdav7s6Ps+gw0c8eU/U8Z2fDFRhyd6Z7Dd
f4k0HYKXZy605/d05cTHYnmTZE1Hw0cRfue2f0Kc53A5y5414VCo52caBh5akUs9e9horf/4bhoHG
k/Y7c310Ay1j9E11Hzf3qcsCnmg1B8mGJxnl//c0Wd652P2xHuNfUGFT8YQbxx+1/TQFy9j1s13e

Gambar 12 : Result Usecase diagram sudah di encode

Dan yang terakhir adalah mengenerate footer. Langkah ini sebenarnya adalah langkah terakhir dan paling mudah karena dilangkah ini hanya menambahkan penutup tag dari mxfiles. Setelah tahap-tahap diatas sudah selesai maka hasil konv [16](#) pun sudah dapat dilihat. Hasil dari konversi dapat dilihat pada gambar 12 dan gambar 13.

[2](#) Pada gambar 12 merupakan hasil dari gambar 2 dan gambar 3. Gambar 12 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 13 : Result Class Diagram

Pada gambar 13 adalah hasil dari konversi yang keluaran nya adalah class diagram. Class diagram ini didapat dari collecting dataobject. Hasil dari collecting dataobject dapat

dilihat pada gambar 6 untuk melihat kecocokannya.

SIMPULAN

Berdasarkan dari gambar 12 dan gambar 13 aplikasi dapat mengkonversi dari BPMN (Bizagi Modeler) ke UML (DrawIO). Hasil tersebut menunjukkan bahwa penelitian ini telah berhasil mengimplementasikan Aturan-aturan yang dibuat oleh (Rhazali, Mouloudi, & Hadi, 2014) dan menggunakan aturan tambahan dari penulis untuk menyokong 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*, 71.
- 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 Intituti 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*, 14(2).
- 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.
- Sholih, & 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.

Sistem Konversi Business Process Model Notation ke Unified Modelling Language

ORIGINALITY REPORT

18%

SIMILARITY INDEX

14%

INTERNET SOURCES

5%

PUBLICATIONS

17%

STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Forum Perpustakaan Perguruan Tinggi Indonesia Jawa Timur Student Paper	5%
2	Submitted to Universitas Brawijaya Student Paper	2%
3	Submitted to Study Group Australia Student Paper	1%
4	widuri.raharja.info Internet Source	1%
5	Submitted to Universidad de Lima Student Paper	1%
6	jurnal.uinsu.ac.id Internet Source	1%
7	Oualid Betari, Saida Filali, Amine Azzaoui, Mohamed Amine Boubnad. "Applying a Model Driven Architecture Approach: Transforming CIM to PIM Using UML", International Journal of Online Engineering (iJOE), 2018 Publication	1%

8	Internet Source	1 %
9	pt.scribd.com Internet Source	1 %
10	repository.ub.ac.id Internet Source	1 %
11	Submitted to Syiah Kuala University Student Paper	1 %
12	id.123dok.com Internet Source	1 %
13	id.scribd.com Internet Source	1 %
14	Submitted to STIKOM Surabaya Student Paper	<1 %
15	aepnurulhidayat.wordpress.com Internet Source	<1 %
16	repository.ipb.ac.id Internet Source	<1 %
17	Artūrs Solomencevs. "Comparing Transformation Possibilities of Topological Functioning Model and BPMN in the Context of Model Driven Architecture", Applied Computer Systems, 2016 Publication	<1 %
18	digilib.unila.ac.id Internet Source	<1 %

19	www.scribd.com Internet Source	<1 %
20	pt.slideshare.net Internet Source	<1 %
21	rekayantianwar.blogspot.com Internet Source	<1 %
22	eprints.kfupm.edu.sa Internet Source	<1 %
23	trepo.tuni.fi Internet Source	<1 %

Exclude quotes Off
Exclude bibliography Off

Exclude matches Off