TUGAS AKHIR – KI141502

DEKOMPOSISI PROSES BISNIS UNTUK OPTIMASI PROSES MINING

YUTIKA AMELIA EFFENDI
NRP 5112 100 176

Dosen Pembimbing I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.

Dosen Pembimbing II
Nurul Fajrin Ariyani, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

BUSINESS PROCESS DECOMPOSING FOR OPTIMATION PROCESS MINING

YUTIKA AMELIA EFFENDI
NRP 5112 100 176

Supervisor I
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.

Supervisor II
Nurul Fajrin Ariyani, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DEKOMPOSISI PROSES BISNIS UNTUK OPTIMASI PROSES MINING

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
YUTIKA AMELIA EFFENDI
NRP. 5112 100 176

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Prof. Drs. Ec. Ir. RIYANARTO SARNO, M.Sc., Ph.D.
NIP: 19590803 198601 1 001 ..... (pembimbing 1)

NURUL FAJRIN ARIYANI, S.Kom., M.Sc.
NIP: 19860722 201504 2 003 ..... (pembimbing 2)

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INFORMASI
JURUSAN TEKNIK INFORMATIKA

SURABAYA
DESEMBER, 2015

DEKOMPOSISI PROSES BISNIS UNTUK OPTIMASI PROSES MINING

Nama : **Yutika Amelia Effendi**
NRP : **5112100176**
Jurusan : **Teknik Informatika – FTIf ITS**
Dosen Pembimbing I : **Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.**
Dosen Pembimbing II : **Nurul Fajrin Ariyani, S.Kom., M.Sc.**

Abstrak

Proses mining atau proses discovery adalah metode yang secara otomatis dapat menemukan dan merekonstruksi alur kerja proses bisnis dari event log. Proses mining penting untuk menemukan aktivitas-aktivitas yang dieksekusi bersamaan karena banyak dari aktivitas tersebut terkandung dalam proses bisnis. Karena proses discovery mendapat perhatian di antara para peneliti dan praktisi, maka kualitas proses model yang dihasilkan akan diperlukan. Model proses bisnis biasanya mengandung input, output, dan relasi, baik sequence maupun paralel. Namun, sistem informasi enterprise yang ada sekarang sudah kompleks dan disusun oleh jumlah komponen besar yang biasanya diimplementasikan di lingkungan sebuah perusahaan. Oleh karena itu, proses mining digunakan untuk men-discover model struktural untuk workflow yang kompleks dari event log yang berasal dari berbagai departemen yang ada di dalam sebuah perusahaan.

Dalam skripsi ini, diusulkan metode untuk menemukan dan menggabungkan proses model untuk sistem informasi yang kompleks dengan menggunakan top-to-down proses mining berdasarkan perbaikan petri nets dari multi-source event log serta algoritma proses mining yang populer, yaitu algoritma Heuristics Miner, berdasarkan modifikasi interval waktu. Dengan menggunakan informasi waktu untuk aktivitas yang dilakukan memungkinkan algoritma untuk menghasilkan model alur kerja

yang lebih baik. Algoritma yang diusulkan dapat secara efektif membedakan relasi single choice XOR, conditional OR dan parallel AND. Interval ambang batas (threshold) ditentukan berdasarkan rata-rata ukuran dependency graph.

Hasil eksperimen menunjukkan bahwa algoritma yang diusulkan dapat menemukan proses bisnis kompleks dari sebuah perusahaan yang berasal dari berbagai departemen dan proses model yang dibentuk oleh parallel AND dan conditional OR serta mengevaluasi metode lain dari algoritma proses discovery, seperti algoritma Heuristics Miner, algoritma Process Model Discovery Based on Activity Lifespan dan algoritma Heuristics Miner for Time Intervals. Setelah itu, kami menunjukkan perbandingan dari kualitas fitness masing-masing algoritma, validitas dari proses model dengan algoritma yang diusulkan, kelengkapan dari proses model dengan algoritma yang diusulkan dan optimasi waktu dan biaya produksi dengan menggunakan pemrograman non-linear.

***Kata kunci:** Process Mining, Process Discovery, Discovery Parallel Activity OR, Modified Time-Based Heuristics Miner, Time–cost Optimization, Double Timestamp Event Log, Non-Linear Programming, Activity Lifespan, Quality Fitness, Validity, Completeness, CPM Crashing Project*

BUSINESS PROCESS DECOMPOSING FOR OPTIMATION PROCESS MINING

Student Name	**: Yutika Amelia Effendi**
NRP	**: 5112100176**
Major	**: Teknik Informatika – FTIf ITS**
Supervisor I	**: Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.**
Supervisor II	**: Nurul Fajrin Ariyani, S.Kom., M.Sc.**

Abstract

Process Mining or Process Discovery is a method to automatically discover and reconstruct the workflow of a business process from logs of activities. Process mining for discovering concurrent activities is important since there are many of them contained in business processes. Since researchers and practitioners are giving attention to the process discovery, then the quality of the discovered process models is required. Business process model contains input, output, and relation (sequence and parallel traces). However today's enterprise information systems are complex and composed of a large number of components, which are usually implemented on an organization. So, process mining is used to discover the structural model for a complex workflow from multi-source event log collected from distributed department.

In this undergraduate thesis book, we propose a method to obtain and to merge process model for complex workflow systems using top-to-down process mining based on refinement of petri nets from multi-source event log and a modification to one of process mining algorithms, Heuristics Miner, to Modified Time-Based Heuristics Miner. We propose modification method because using time-based information for the activities in event log enable the algorithm to produce better process models. The proposed algorithm can effectively distinguish single choice XOR, conditional OR and parallel AND. The threshold intervals are

determined based on average dependency measure in dependency graph.

The experimental results show that the proposed algorithm can discover the complex business process from an organizational that is collected from distributed department and the concurrent business processes formed by parallel AND and conditional OR and evaluate other methods for process discovery techniques, such as original Heuristics Miner algorithm, Process Model Discovery Based on Activity Lifespan and Heuristics Miner for Time Intervals. After that, we show the evaluation of quality fitness, the validity, the completeness and time-cost optimization using non-linear programming.

Keywords: Process Mining, Process Discovery, Discovery Parallel Activity OR, Modified Time-Based Heuristics Miner, Time–cost Optimization, Double Timestamp Event Log, Non-Linear Programming, Activity Lifespan, Quality Fitness, Validity, Completeness, CPM Crashing Project

DAFTAR ISI

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji bagi Allah SWT, Tuhan semesta alam yang telah melimpahkan rahmat dan hidayah-Nya kepada penulis, sehingga tugas akhir berjudul "Dekomposisi Proses Bisnis untuk Optimasi Proses Mining" ini dapat selesai sesuai dengan waktu yang telah ditentukan.

Pengerjaan tugas akhir ini menjadi sebuah sarana untuk penulis memperdalam ilmu yang telah didapatkan selama menempuh pendidikan di kampus perjuangan Institut Teknologi Sepuluh Nopember Surabaya, khususnya dalam disiplin ilmu Teknik Informatika. Terselesaikannya buku tugas akhir ini tidak terlepas dari bantuan dan dukungan semua pihak. Pada kesempatan kali ini penulis ingin mengucapkan terima kasih kepada:

1. Bapak, Ibu, adik dan keluarga yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Bapak Riyanarto Sarno dan Ibu Nurul Fajrin Ariyani selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan Tugas Akhir ini.
3. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
4. Seluruh staf dan karyawan FTIf ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
5. Segenap dosen rumpun mata kuliah Manajemen Informasi.
6. Rekan-rekan laboratorium Manajemen Informasi dan laboratorium Komputasi Cerdas dan Visi.

7. Teman-teman ‘Yuk Mari’ dan TC Angkatan 2012 yang selalu mendukung, menyemangati, membantu dan mendengarkan suka duka selama proses pengerjaan tugas akhir.
8. Serta semua pihak yang turut membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih memiliki banyak kekurangan. Dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Desember 2015

DAFTAR GAMBAR

DAFTAR KODE SUMBER

DAFTAR ISTILAH

activity
Merupakan bagian dari *case* yang merupakan sub proses dalam pembuatan suatu barang atau dalam suatu proses tertentu.

case
Suatu kasus tertentu yang ada pada *event log*. Kasus tertentu tersebut dapat berupa suatu kasus dalam memproduksi suatu barang tertentu, karena *event log* dapat terdiri dari catatan dari proses eksekusi pembuatan banyak barang atau proses eksekusi dari banyak kasus proses.

cost slope
Perbandingan antara pertambahan biaya dengan percepatan waktu penyelesaian proyek.

CPM *(Critical Path Method)*
Teknik untuk menganalisis jaringan kegiatan/aktivitas-aktivitas ketika menjalankan proyek dalam rangka memprediksi durasi total.

crash cost
Biaya yang dikeluarkan dengan penyelesaian proyek dalam jangka waktu sebesar durasi *crash*-nya. Biaya setelah di *crashing* akan menjadi lebih besar dari biaya normal.

crash time
Waktu yang dibutuhkan suatu proyek dalam usahanya mempersingkat waktu yang durasinya lebih pendek dari *normal duration*.

dependency graph
Graph dari hubungan antar aktivitas yang sesuai dengan kriteria yang diinginkan.

discovery
Salah satu teknik *process mining* yang bertujuan untuk mendapatkan proses model dengan menggali informasi dari *event log*.

durasi normal
Waktu yang dibutuhkan untuk menyelesaikan suatu aktivitas atau

kegiatan dengan sumber daya normal yang ada tanpa adanya biaya tambahan lain dalam sebuah proyek.

event log

Suatu set proses eksekusi yang mengambil dari data aktivitas proses bisnis yang dilakukan dalam konteks tertentu.

frekuensi aktivitas

Jumlah kejadian hubungan antar aktivitas dalam *event log*.

heuristics miner

Salah satu algoritma yang digunakan untuk melakukan proses *discovery*.

Length One Loop

Merupakan hubungan dimana memungkinkan suatu proses dari suatu aktivitas kembali ke aktivitas itu sendiri.

Length Two Loop

Merupakan hubungan bolak balik antar dua aktivitas.

makespan

Durasi total terpanjang dari suatu proses bisnis.

normal cost

Biaya yang dikeluarkan dengan penyelesaian proyek dalam waktu normal. Perkiraan biaya ini adalah pada saat perencanaan dan penjadwalan bersamaan dengan penentuan waktu normal.

paralel

Suatu rangkaian proses bisnis dimana eksekusi dari aktivitas yang terdapat dalam rangkaian tersebut dapat dilakukan secara bersamaan maupun berurutan.

parallel measure

Perhitungan untuk menentukan batas yang berguna dalam penentuan jenis paralel yang digunakan dalam proses bisnis.

process discovery

Salah satu proses yang paling menantang dari rangkaian *process mining*, tujuan dari proses ini adalah untuk membentuk model dengan cara menggali informasi dari data yang tercatat dalam suatu *event log*.

process mining

Teknik yang dapat digunakan untuk mendapatkan model

sesungguhnya dari proses bisnis yang terjadi dalam sebuah sistem informasi berdasarkan data yang berasal dari *event log*.

proses bisnis

Sekumpulan aktivitas yang dibuat untuk menghasilkan keluaran spesifik dengan tujuan tertentu.

double timestamp event log

Event log yang memiliki dua jenis informasi waktu, yaitu *start time* dan *finish time.*

threshold

Ambang batas yang digunakan untuk menjadi penentu dari penyaringan hubungan antar aktivitas.

timestamp

Suatu informasi yang menyimpan data tanggal dan waktu suatu kejadian dilakukan.

trace

Alur dari aktivitas yang dijalankan dalam suatu proses.

UML (*Unified Modelling Language*)

Salah satu bahasa pemodelan proses bisnis.

YAWL (*Yet Another Workflow Language*)

Salah satu bahasa pemodelan proses bisnis.

DAFTAR TABEL

NOMENKLATUR

$X =>_w Y$	: nilai *dependency* dari aktivitas X ke Y.
$\|X >_w Y\|$	: frekuensi aktivitas X yang diikuti secara langsung oleh Y.
$\|Y >_w X\|$	: frekuensi aktivitas Y yang diikuti secara langsung oleh X.
$X =>_w X$	: nilai *dependency Length One Loop.*
$\|X >_w X\|$	: frekuensi aktivitas X yang diikuti secara langsung oleh X.
$max\{\|X >_w A\|\| A \in e\}$	: frekuensi aktivitas X yang diikuti oleh aktivitas A dimana aktivitas A merupakan bagian dari aktivitas yang ada di *event log*.
$X =>_{2w} Y$	: nilai *dependency Length Two Loop.*
$\|X \gg_w Y\|$	: frekuensi dari *trace* dalam bentuk XYX yang muncul dalam *event log.*
$\|Y \gg_w X\|$	: frekuensi dari *trace* dalam bentuk YXY yang muncul dalam *event log.*
$X =>_w Y \wedge Z$	: *parallel measure* antara Y dan Z dimana *split* terjadi di X.
$\|Y >_w Z\|$	: frekuensi aktivitas Y yang diikuti secara langsung oleh Z.

$\|Z >_w Y\|$	: frekuensi aktivitas Z yang diikuti secara langsung oleh Y.
$\|X >_w Y\|$	: frekuensi aktivitas X yang diikuti secara langsung oleh Y.
$\|X >_w Z\|$	: frekuensi aktivitas X yang diikuti secara langsung oleh Z.
Z	: nilai fungsi tujuan.
RBT	: *Relative-to-best threshold.*
$Avg\ PDM$	: rata-rata positive dependency measure yang lebih dari 0 pada *matrix dependency*.
DM	: *dependency measure.*
$DM_{a=>b}$	: *dependency measure* antar aktivitas dalam dependency matriks.
$SD\ PDM$	: standar deviasi *positive dependency measure* pada *matrix dependency.*
POT	: *Positive observations threshold.*
$f_{a=>b}$	: frekuensi antar aktivitas pada matriks frekuensi.
DT	: *dependency threshold.*
$X =>_w Y \wedge Z$	: *parallel measure* antara aktivitas pada percabangan dari X yaitu Y dan Z.
$\|Y \gg>_w Z\|$	: *undirect and direct followed frequency* aktivitas Y dan Z.
$\|Z \gg>_w Y\|$	: *undirect and direct followed frequency* aktivitas Z dan Y.
$\|X >_w Y\|$	: frekuensi aktivitas X yang diikuti secara langsung oleh

Y.

$\|X >_w Z\|$	: frekuensi aktivitas X yang diikuti secara langsung oleh Z.
$\|Y \gg> not_w Z\|$	: frekuensi eksekusi aktivitas Y yang tidak diikuti aktivitas Z.
$\|Z \gg> not_w Y\|$	: frekuensi eksekusi aktivitas Z yang tidak diikuti aktivitas Y.
PM	: *Parallel Measure.*
Minimum PDM	: nilai minimum dari *positive dependency measure* pada *matrix dependency.*
c	: jumlah dari *trace* yang berhasil diuraikan dengan benar.
t	: total jumlah *trace* di dalam *event log.*
PMw	: *parsing measure* untuk mengukur kualitas *fitness.*
e	: total jumlah dari aktivitas di dalam *event log.*
m	: jumlah dari input yang hilang (misalnya aktivitas yang tidak berhasil diurai di *trace).*
r	: jumlah dari *output* yang tersisa (misalnya aktivitas yang mengikuti hingga akhir *trace).*
$CPMw$	: *continuous parsing measure* untuk mengukur kualitas *fitness.*
X_i	: *time slope* yang terjadi untuk penyelesaian aktivitas ke – i

dimana i ∈ {Aktivitas} (variable keputusan).

T_{ni} : durasi normal aktivitas ke – i dimana i ∈ {Aktivitas}.

T_{ci} : *crash* durasi aktivitas i dimana i ∈ {Aktivitas}.

S_i : *Cost slope* aktivitas ke – i dimana i ∈ {Aktivitas}.

C_{ni} : *cost* normal aktivitas ke – i dimana i ∈ {Aktivitas}.

C_{ci} : *cost crash* aktivitas ke – i dimana i ∈ {Aktivitas}.

ed : durasi eksekusi.

$et_{activity_{output}}$: waktu eksekusi *output* aktivitas.

$et_{activity}$: waktu eksekusi aktivitas.

ed_k : waktu eksekusi aktivitas pada *case* ke-k.

$cost_k$: biaya aktivitas pada *case* ke-k.

h : banyaknya aktivitas dieksekusi dalam *event log* (banyaknya *case* yang mengandung aktivitas).

P : *place* pada *Petri Net.*

T : transisi (aktivitas) pada *Petri Net.*

$X \rightarrow w^{Y\wedge Z}$: hasil perhitungan *parallel measure* pada tahap *mining parallel activity*, dimana aktivitas X memiliki *split* atau *join* ke aktivitas Y dan Z.

$|Y\ ||_w Z|$: perhitungan untuk jumlah (per *case* aktivitas) dimana

	aktivitas Y dan Z yang memiliki relasi paralel dalam *event log*.
$\sum^{'}_{TPM}$	: *Top level* proses model pada metode *top-to-down process mining*.
m	: *message* antar departemen pada metode *top-to-down process mining*.
$\sum^{'}_{BPM}$	: *bottom level* proses model pada metode *top-to-down process mining*.
e_i	: *dependency measure* pada *dependency matrix*.
n_e	: jumlah *dependency measure*.
$Avg\ PM$	:rata-rata dari *parallel measure* yang memiliki induk aktivitas yang sama.
n_{PM}	:jumlah *parallel measure* yang memiliki *input* atau *output* yang sama.

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pengerjaan Tugas Akhir, dan sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Tujuan dari proses *discovery* adalah untuk menentukan model proses bisnis dari *event log*. Proses *discovery* adalah salah satu tantangan di dalam proses *mining*. Proses *discovery* adalah suatu teknik yang secara otomatis membangun proses model dari aktivitas saat ini dan variasi utama aktivitas sebuah perusahaan. Teknik ini menggunakan *event log* dari sekumpulan aktivitas yang dieksekusi dalam sebuah perusahaan. Model proses bisnis dianalisis untuk menunjukkan kompleksitas masalah dan bagaimana menyelesaikannya. Isu-isu masalah ini terdapat di bidang apapun, misalnya bisnis (Olatunde T. Baruwa, Miquel A. Piera, 2015), lingkungan (Adnen Sanaa, Samir Ben Abid, Abdennacer Boulila, Chokri Messaoud, Mohammed Boussaid, Najeh Ben Fadhel, 2015) (Jie Yuan, David Oswald, Wei Li, 2015), *smartphone* (Victor R.L. Shen, Horng-Yih Lai, Ah-Fur Lai, 2015), *fraud*(kecurangan) (Riyanarno Sarno, Rahadian Dustrial Dewandono, Tohari Ahmad, Mohammad Farid Naufal, dan Fernandes Sinaga, 2015), dll. Setiap teknik *discovery* memiliki kelebihan dan kekurangan yang berbeda, sehingga menyebabkan aktivitas-aktivitas dari perusahaan yang ingin dicapai tidak dapat disajikan dalam proses model yang ditemukan. Untuk mengatasi masalah ini, kami meningkatkan salah satu teknik *discovery* agar dapat menyajikan proses model lebih baik.

Sistem informasi *enterprise* yang ada sekarang sudah kompleks dan disusun oleh jumlah komponen besar yang biasanya diimplementasikan di lingkungan sebuah perusahaan dengan berbagai departemen. Oleh karena itu, proses *mining*

digunakan untuk men-*discover* model struktural untuk *workflow* yang kompleks dari *event log* yang berasal dari berbagai departemen yang ada di dalam sebuah perusahaan. Ada beberapa kriteria untuk mengetahui *workflow* dari *multi-source event log*, yaitu: (1) *workflow* dari *event log* yang digunakan untuk proses *mining* tersebar di server-server yang berbeda; (2) *workflow* dari *event log* yang tersimpan di server yang berbeda memiliki struktur *event log* yang berbeda; (3) *workflow* dari *event log* yang disimpan oleh perusahaan tidak dapat diakses dengan alasan keamanan; (4) *workflow* dari *event log* sebuah departemen dapat menjadi patokan proses bisnis dari sebuah perusahaan dengan mengetahui interaksi satu departemen dengan departemen yang lainnya. Metode yang digunakan dalam Tugas Akhir ini yaitu dengan menerapkan kriteria yang terakhir. Dengan menggunakan *top-to-down proses mining*, langkah awal adalah men-*discover top* dan *bottom level* dari proses model kemudian menggabungkan proses model *top* dan *bottol level* menjadi proses model kompleks yang dapat menggambarkan proses bisnis sebuah perusahaan. Metode ini mengasumsikan *event log* terletak di server yang sama (Qingtian Zeng, Hua Duan, Cong Liu, 2015).

Process discovery dapat menggunakan banyak algoritma, seperti algoritma *alpha*, *alpha+*, *alpha++*, dan *Heuristics miner* (Sofie De Cnudde, Jan Claes, Geert Poels, 2014). Pada awalnya, diantara banyak algoritma tersebut, tidak ada algoritma yang fokus menemukan relasi *conditional OR*. Algoritma-algoritma yang ada hanya menemukan relasi *conditional OR* sebagai relasi *parallel AND* atau relasi *single choice XOR*. Penemuan relasi paralel yang tidak tepat dari suatu proses model akan mengubah hasil *discovery*. Tugas Akhir ini akan memodifikasi algoritma *Heuristics Miner*. Telah terbukti bahwa algoritma *Heuristics Miner* yang telah dimodifikasi dapat menemukan proses bisnis yang mengandung relasi paralel *conditional OR*, dimana relasi paralel *conditional OR* ini tidak dapat ditemukan dengan menggunakan algoritma *Heuristics miner* yang ada saat ini. Algoritma *Heuristics Miner* yang telah dimodifikasi, untuk

menemukan sebuah proses model secara umum sama dengan algoritma *Heuristics Miner* yang ada sekarang; perbedaannya terletak pada penemuan relasi aktivitas paralel. Algoritma *Heuristics Miner* yang telah dimodifikasi menggunakan frekuensi aktivitas paralel yang tidak diikuti oleh aktivitas lain, tidak hanya frekuensi aktivitas yang diikuti oleh aktivitas lain saja. Aktivitas-aktivitas paralel diambil dari *causal matrix* yang telah dihitung sebelumnya, kemudian perhitungan untuk paralel menggunakan sebuah persamaan baru. Oleh karena itu, interval *threshold* dari rata-rata pengukuran paralel digunakan untuk menentukan tipe relasi paralel *split* dan *join*. Hasil penelitian menunjukkan bahwa algoritma *Heuristics Miner* yang telah dimodifikasi dapat menemukan proses model dengan benar, yang mengandung relasi paralel *single choice* XOR, *parallel* AND, dan *conditional* OR.

Setelah memperoleh proses bisnis yang lengkap, kualitas *fitness*, *validity*, *completeness* dan biaya efektif di dalam proses bisnis merupakan isu yang sangat penting untuk di evaluasi dan di optimasi (J. W. Zhang Jingwen, Y. Xu and Z. W. He, 2006). Sebuah proses model dengan *fitness* yang baik memungkinkan perilaku dari aktivitas dapat dilihat dari *event log* (yang "cocok" dari proses model dengan "realita"). Jika semua *trace* di *event log* tercantum di dalam proses model dari awal hingga akhir, maka proses model tersebut memiliki *fitness* yang sempurna. Tetapi jika *event log* memiliki banyak *trace* yang tidak tercantum di dalam proses model, maka *fitness* dari proses model tersebut tidak sempurna. *Completeness* didefinisikan sebagai dependensi penting yang tidak ditampilkan di dalam proses model yang ditemukan. Masalah dari *completeness* fokus pada perhitungan dependensi dari *Length Two Loop*. *Validity* fokus pada proses model semantik yang dihasilkan oleh algoritma *Modified Time-Based Heuristics Miner*. *Validity* dari proses model semantik tidak dianggap valid karena tidak berdasarkan pada aturan teoritis yang didefinisikan dalam paper (Sofie De Cnudde, Jan Claes, Geert Poels, 2014) (A.J.M.M. Weijters, J.T.S. Ribeiro, 2010). Valid berarti proses model dihasilkan yang menggambarkan

transformasi dari *dependency graph* ke proses model semantik. Untuk membangun sebuah proses model semantik, kita perlu mengetahui dua langkah utama. Pertama, memodelkan *dependency graph* menjadi *causal net* dan yang kedua, memodelkan *causal net* menjadi proses model semantik.

Manajemen waktu proyek dan manajemen biaya proyek adalah dua domain penting dalam pencapaian biaya proyek yang efektif. Optimasi waktu dan biaya adalah sebuah masalah yang menunjukkan dua domain inti (J. W. Zhang Jingwen, Y. Xu and Z. W. He, 2006) (X. L. Wu and Z. Yin, 2007). *Critical Path Method* (CPM) dikembangkan pada akhir 1950-an, TCO telah mendapat perhatian dari para peneliti dan telah menghasilkan sejumlah besar publikasi meliputi bidang metode optimasi dan model optimasi. CPM *crashing project* merupakan salah satu metode yang digunakan untuk menyelesaikan masalah optimasi waktu dan biaya. Masalah optimasi waktu dan biaya itu sendiri memiliki dua aspek utama: pertama adalah durasi kegiatan yang perlu dikurangi dan kedua, jumlah durasi kegiatan yang perlu dikurangi (J. E. Kelley and M. R. Walker, 1959) (S. E. Elmaghraby, 2000). Berbagai model optimasi yang berbeda telah diusulkan untuk memecahkan TCO menggunakan CPM. Di dalam Tugas Akhir ini, penyelesaian optimasi waktu dan biaya menggunakan *Non-Linear programming*. Ada banyak model data yang mewakili proses bisnis dalam optimasi waktu dan biaya, salah satunya adalah *event log* dengan *double timestamp*. *Event log* dengan *double timestamp* memiliki data aktivitas yang ada dalam proses bisnis, seperti waktu pelaksanaan aktivitas dalam setiap *case* dan data biaya dari setiap aktivitas dalam setiap *case*, tetapi *event log* dengan *double timestamp* tidak memiliki data yang digunakan untuk menghitung *crashing project*, oleh karena itu di dalam Tugas Akhir ini akan membahas suatu metode untuk menghitung data dari *event log* dengan *double timestamp* yang digunakan untuk menghitung *crashing project*.

Event log yang diolah dalam tugas akhir ada 2 jenis yaitu untuk permasalahan *top-to-down process mining* menggunakan

event log produksi benang, sedangkan untuk permasalahan modifikasi algoritma *heuristics miner* dan optimasi waktu-biaya menggunakan *event log* dengan *double timestamp*. *Event log* 1 terdiri dari 11 aktivitas, 18 *case, start time, end time,* dan mengandung *length one loop* dan *length two loop*. *Event log* 2 memiliki 6 *case,* 5 aktivitas, *start time, end time, originator* dan *cost*. *Event log* 3 dan *event log* 5 memiliki 10 *case,* 10 aktivitas, *start time, end time, originator* dan *cost*. *Event log* 4 dan *event log* 6 memiliki 13 *case,* 11 aktivitas, *start time, end time, originator* dan *cost*.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana menemukan proses bisnis suatu departemen dari sebuah perusahaan dengan *multi-source event log* dan teknik proses *mining?*
2. Bagaimana menemukan proses model dari *top level* dan proses model dari *bottom level?*
3. Bagaimana menggabungkan proses model dari *top level* dan *bottom level* yang sudah diketahui menjadi proses bisnis kompleks?
4. Metode apakah yang dapat digunakan untuk membantu men-*discover* proses bisnis yang kompleks dengan benar?
5. Bagaimana membuat aktivitas sebagai interval waktu dengan menggunakan algoritma *heuristics miner?*
6. Bagaimana memodifikasi algoritma *heuristics miner* agar dapat memodelkan proses bisnis dengan *threshold* otomatis?
7. Bagaimana memodifikasi algoritma *heuristics miner* agar dapat memodelkan relasi paralel *conditional* OR?
8. Bagaimana hasil evaluasi metode yang diusulkan (*modified time-based heuristics miner*) jika dibandingkan dengan metode lain seperti *heuristics miner*, *heuristics*

miner for time intervals dan *process model discovery based on activity lifespan*?

9. Bagaimana hasil kualitas *fitness* dari algoritma *modified time-based heuristics miner, heuristics miner, heuristics miner for time intervals* dan *process model discovery based on activity lifespan*?
10. Bagaimana hasil *validity* dari proses model dengan algoritma yang diusulkan (*modified time-based heuristics miner*)?
11. Bagaimana hasil *completeness* dari proses model dengan algoritma yang diusulkan (*modified time-based heuristics miner*)?
12. Bagaimana hasil optimasi waktu dan biaya dari proses model dengan algoritma yang diusulkan (*modified time-based heuristics miner*)?
13. Bagaimana mendapatkan nilai durasi dan tambahan biaya yang paling optimal dari proses model?
14. Bagaimana hasil keluaran dari perangkat lunak yang dibangun?

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Bahasa pemrograman yang digunakan adalah Python.
2. Aplikasi yang dikembangkan merupakan aplikasi *dekstop*.
3. *Library* Python yang digunakan adalah *numpy, xlrd, dan xlwt.*
4. Data masukan berupa *event log* dalam bentuk Excel.
5. Data keluaran proses *discovery* dalam bentuk *input, output,* relasi, dan *short loop* dari proses *mining.*

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Menemukan proses bisnis dari salah satu departemen dalam sebuah perusahaan dengan *multi-source event log* dan teknik proses *mining.*
2. Menemukan proses model dari *top level* dan *bottom level.*
3. Menggabungkan proses model *top level* dan *bottom level* sehingga menjadi proses bisnis kompleks dengan proses *refinement of petri nets.*
4. Menggunakan metode *top-to-down process mining based on refinement of petri nets from multi-source event log* untuk menemukan proses bisnis kompleks dengan benar.
5. Menentukan aktivitas yang bertindak sebagai interval waktu dengan algoritma *heuristics miner.*
6. Menentukan *threshold heuristics miner* sehingga dapat membentuk model proses bisnis yang ideal tanpa harus memasukkan secara manual.
7. Memodifikasi algoritma *heuristics miner* sehingga dapat men-*discover* bentuk percabangan *conditional* OR.
8. Melakukan evaluasi terhadap metode yang diusulkan (*modified time-based heuristics miner)* jika dibandingkan dengan metode lain seperti *heuristics miner*, *heuristics miner for time intervals* dan *process model discovery based on activity lifespan.*
9. Menentukan hasil kualitas *fitness* algoritma *modified time-based heuristics,original heuristics miner, heuristics miner for time intervals* dan *process model discovery based on activity lifespan.*
10. Menerapkan dua langkah utama untuk memodelkan *dependency graph* menjadi proses model semantik dengan benar agar sesuai dengan konsep *validity.*
11. Menerapkan konsep *completeness* agar proses model dapat mengatasi permasalahan *Length Two Loop.*
12. Menggunakan rata-rata dan standar deviasi dari setiap aktivitas yang ada pada *event log* untuk menghitung data optimasi.

13. Mendapatkan nilai durasi dan tambahan biaya yang paling optimal dengan menggunakan *non-linear programming.*
14. Hasil keluaran dari perangkat lunak yang dibangun adalah *input, output,* relasi antar aktivitas dan *short loop.*

1.5 Manfaat

Manfaat Keilmuan

Manfaat yang dihasilkan dari pengerjaan Tugas Akhir ini adalah terbentuknya suatu metode yang dapat menemukan proses bisnis kompleks dari suatu perusahaan dengan memanfaatkan proses model dari *top level* dan *bottom level* berbagai departemen dan suatu metode yang dapat membuat aktivitas sebagai interval waktu, men-*discover* proses model yang mengandung relasi paralel *conditional* OR, membandingkan hasil pengerjaan dengan metode *heuristics miner, heuristics miner for time intervals* dan *Process Model Discovery Based on Activity Lifespan*, mendapatkan hasil kualitas *fitness, validity, completeness* dan optimasi waktu-biaya yang lebih baik. Untuk mendapatkan proses model yang mengandung relasi paralel *conditional* OR dalam Tugas Akhir ini maka proses model dapat mengembalikan *event log* sesuai dengan *event log* yang digali. Pemodelan relasi paralel *conditional* OR dilakukan dengan memodifikasi algoritma yang sudah ada yaitu *heuristics miner*. Selain pemodelan relasi paralel *conditional* OR terdapat manfaat lainnya yaitu optimasi dari *double timestamp event log* yang menggunakan *non-linear programming.* Untuk optimasi dari *double timestamp event log* diperlukan perhitungan data durasi. Dalam Tugas Akhir ini, data durasi dihitung dengan memodifikasi metode *time prediction* dari Van Der Aalst karena durasi yang dihitung dari *time prediction* merupakan durasi per *trace* sedangkan yang diperlukan untuk optimasi adalah durasi per aktivitas.

Manfaat Praktis

Manfaat dari proses *discovery* sendiri adalah agar dapat menemukan proses yang sebenarnya terjadi, untuk penemuan aktivitas sebagai interval waktu dan relasi paralel *conditional* OR

dimaksudkan utntuk mendapatkan proses model yang benar-benar dapat mengembalikan data yang ada dalam *event log*. Dari proses model yang didapatkan, dapat dilihat bagaimana model proses bisnis yang sesuai untuk SOP ke depannya, hal ini dapat dilihat dengan mendapatkan proses model secara periodik dari *event log*. Sedangkan untuk kualitas *fitness, validity, completeness*, dan optimasi waktu dan biaya produksi lebih membantu mengevaluasi dan memprediksi hasil metode yang diusulkan ini lebih baik daripada 3 metode yang dijadikan pembanding (*heuristics miner, heuristics miner for time intervals* dan *process model discovery based on activity lifespan).*

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1. **Studi literatur**

   Dalam pembuatan Tugas Akhir ini telah dipelajari tentang hal-hal yang dibutuhkan sebagai ilmu penunjang dalam penyelesaiannya. Pertama adalah tentang metode *top-to-down process mining based on refinement of petri nets from multi-source event log*, struktur utama dari *Heuristics Miner, Heuristics Miner for Time Intervals, Activity Lifespan, Fitness, Validity, Completeness,* CPM *Crashing Project*, dan Optimasi dengan *Non-Linear Programming*. Kemudian adalah menentukan batasan-batasan pemetaan. Selain itu, juga dibantu beberapa literatur lain yang dapat menunjang proses penyelesaian Tugas Akhir ini.

2. **Pemodifikasian Metode**

   Pada tahap ini penulis menjabarkan cara pemecahan masalah yang terdapat dalam rumusan masalah. Selain itu penulis juga menjabarkan tentang modifikasi yang telah dilakukan pada algoritma sebelumnya sebagai salah satu kontribusi dalam Tugas Akhir ini.

3. **Analisis dan Perancangan Sistem**

Aktor yang menjadi pelaku adalah pengguna perangkat lunak yang dibangun oleh penulis. Kemudian beberapa kebutuhan fungsional dari sistem ini adalah sebagai berikut:

a. Melakukan proses *mining* untuk menemukan proses model dari *top level* dan *bottom level*.
b. Melakukan proses *mining* untuk menggabungkan proses model dari *top level* dan *bottom level* sehingga menghasilkan proses bisnis kompleks dengan proses *refinement of petri nets*.
c. Melakukan proses *discovery* dari *file event log* menjadi model proses bisnis.
d. Melakukan evaluasi terhadap metode yang diusulkan (*modified time-based heuristics miner)* dengan metode *heuristics miner*, *heuristics miner for time intervals* dan *process model discovery based on activity lifespan.*
e. Melakukan perhitungan kualitas *fitness,* evaluasi *validity*, dan evaluasi *completeness* dari proses model yang ditemukan.
f. Melakukan perhitungan untuk penentuan data optimasi berupa durasi dan biaya.
g. Melakukan optimasi minimum durasi proses bisnis dan minimum biaya tambahan yang dibutuhkan.

4. **Implementasi**

Pada tahap ini dilakukan pembuatan elemen perangkat lunak yang merupakan implementasi dari rancangan pada proses *discovery* dari *file event log* menjadi model proses bisnis dengan menggunakan metode *modified time-based heuristics miner*.

5. **Pengujian dan evaluasi**

Pada tahap ini dilakukan pengujian terhadap elemen perangkat lunak dengan menggunakan data uji yang telah dipersiapkan. Pengujian dan evaluasi

perangkat lunak dilakukan untuk mengevaluasi jalannya perangkat lunak, mengevaluasi fitur utama, mengevaluasi fitur-fitur tambahan, mencari kesalahan yang timbul pada saat perangkat lunak aktif, dan mengadakan perbaikan jika ada kekurangan. Tahapan-tahapan dari pengujian adalah sebagai berikut:

a. pencocokan hasil pada setiap *step* pengerjaan perangkat lunak dengan hasil pengerjaan pada Excel,
b. pencocokan hasil *discovery* uji sistem dengan hasil uji pada eksperimen yang telah dilakukan.

6. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan pendokumentasian dan pelaporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan Tugas Akhir.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan yang menjadi dasar dari pembuatan Tugas Akhir ini.

Bab III Metode Pemecahan Masalah

Bab ini membahas cara penulis memecahkan masalah yang ada. Penjelasan tentang algoritma yang dikembangkan penulis dan langkah-

langkahnya sehingga dapat memecahkan masalah yang ada.

Bab IV Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan alur, proses dan perancangan antarmuka pada perangkat lunak.

Bab V Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak perangkat lunak dan implementasi fitur-fitur penunjang perangkat lunak.

Bab VI Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian subjektif untuk mengetahui penilaian aspek kegunaan (*usability*) dari perangkat lunak dan pengujian fungsionalitas yang dibuat dengan memperhatikan keluaran yang dihasilkan serta evaluasi terhadap fitur-fitur perangkat lunak.

Bab VII Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir. Teori-teori tersebut meliputi *Process Mining*, *Process Discovery*, model proses bisnis, *event log*, *top level* proses model, *bottom level* proses model, proses *refinement of petri nets*, *framework* untuk *top-to-down* proses *mining*, jenis *split* dan *join*, aktivitas *event log* sebagai interval waktu, algoritma *Heuristics Miner*, *Causal Net* (C-Net), proses *mining* untuk relasi aktivitas paralel, kualitas *fitness* dari suatu algoritma, *validity* dari proses model, *completeness* dari proses model, optimasi *time* dan *cost*, dan *Critical Path Method* (CPM).

2.1 Model Proses Bisnis

Proses bisnis merupakan sekumpulan aktivitas yang dibuat untuk menghasilkan *output* spesifik dengan tujuan tertentu (Wang, He, Wen, Wu, ter Hofstede, & Su, 2010). Dari proses model tersebut juga dapat diketahui informasi di mana dan kapan suatu aktivitas dilakukan, kondisi awal sebelum aktivitas dilakukan, kondisi akhir setelah aktivitas dilakukan, serta *input* dan *output* yang jelas. Adapun ciri-ciri dari proses bisnis itu sendiri adalah sebagai berikut:

1. Mempunyai tujuan tertentu
2. Mempunyai *input* (masukan) yang spesifik.
3. Mempunyai *output* (keluaran) yang spesifik.
4. Memanfaatkan *resource*.
5. Memiliki aktivitas yang dapat dieksekusi dengan urutan tertentu.
6. Dapat melibatkan lebih dari satu organisasi.

Model proses bisnis merupakan representasi dari proses bisnis. Sehingga sebuah model proses bisnis harus secara jelas mendefinisikan setiap ciri-ciri yang harus dimiliki oleh suatu proses bisnis. *Unified Modeling Language* (UML) merupakan

salah satu representasi dasar dari proses bisnis. Saat ini representasi dari model proses bisnis itu sendiri sudah banyak berkembang dan banyak jenisnya. Mulai dari *Causal Net*, UML, *Business* BPEL, *Business Process Model and Notation* (BPMN), EPC, PNML, dan masih banyak lagi. Tetapi, masing-masing jenis tersebut juga memiliki kegunaan dan fungsi sendiri-sendiri.

2.2 *Event Log*

Di dalam proses *mining*, untuk menganalisis suatu proses bisnis digunakan *event log* dari proses bisnis tersebut sebagai acuannya. *Event log* didefinisikan sebagai suatu set proses eksekusi yang mengambil data aktivitas proses bisnis yang dilakukan dalam konteks tertentu (Wen, Aalst, Jianmin, & Sun, 2012). Atau dengan kata lain, *event log* merupakan catatan dari eksekusi aktivitas dalam suatu proses bisnis. Catatan eksekusi ini dapat menyimpan data berupa waktu dilaksanakannya suatu aktivitas, *resource* yang melaksanakan aktivitas, dan lain-lain sesuai dengan kebutuhan dari perusahaan yang menjalankan *event log*-nya. Dalam *event log* dapat terdiri dari berbagai macam *case*, *trace*, dan *activity* (van der Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, 2011).

- *Case* dan *Trace*

Case merupakan suatu kasus tertentu yang ada pada *event log*. Kasus tersebut dapat berupa suatu kasus dalam memproduksi suatu barang tertentu, karena *event log* dapat terdiri dari catatan dari proses eksekusi pembuatan banyak barang atau proses eksekusi dari banyak kasus proses. Sedangkan *trace* merupakan alur dari aktivitas yang dijalankan dalam suatu proses. Misal dalam suatu *event log (E)*:

$$E = [< a, c, d >^{45}, < b, c, d >^{42}, < a, c, e >^{38}, < b, c, e >^{22}]$$

Dalam *event log* tersebut:

- Terdapat 4 *trace* yaitu (a,c,d), (b,c,d), (a,c,d), (b,c,e).

- Terdapat 147 *case* karena (a, c, d) dilakukan sebanyak 45 kali, (b, c, d) sebanyak 42 kali, (a, c, e) sebanyak 38 kali, dan (b, c, e) sebanyak 22 kali.

➨ *Activity*

Merupakan bagian dari *case* yang merupakan sub proses dalam pembuatan suatu barang atau dalam suatu proses tertentu. Misal pada *event log (E):*

$L = [< a, c, d >^{45}, < b, c, d >^{42}, < a, c, e >^{38}, < b, c, e >^{22}]$

Dalam *event log* tersebut:
Terdapat 5 aktivitas yaitu {a, b, c, d, e}.

2.3 *Process Mining*

Process mining adalah penelitian baru berfokus antara *machine learning* dan *data mining* yang menangani pemodelan proses dan analisisnya. Tujuan *process mining* adalah menemukan, mengawasi, dan mengembangkan proses asli yang didapat dengan menggali informasi dari *event log* yang tersedia pada sistem saat ini (Aalst, 2010). *Event log* yang didapat dari sistem informasi saat ini sangat banyak dan sebagian besarnya tidak terstruktur. Sehingga pengekstrakan informasi dari sistem membutuhkan usaha yang lebih. *Process mining* terdiri dari 3 teknik yaitu, *discovery*, *conformance*, dan *enhancement.*

- *Discovery*

Salah satu teknik *process mining* yang bertujuan untuk mendapatkan proses model dengan menggali informasi dari *event log*. *Process discovery* terdiri dari beberapa algoritma seperti *alpha*, *alpha+*, *alpha++*, dan *genetic algorithm.*

- *Conformance*

Salah satu teknik *process mining* yang bertujuan membandingkan hasil proses model dari *event log* dengan proses model yang sudah ada. Kemudian, perbedaan diantaranya akan dianalisis lebih lanjut. Teknik ini dapat berguna dalam pengecekan kecurangan dalam proses bisnis (Riyanarno Sarno,

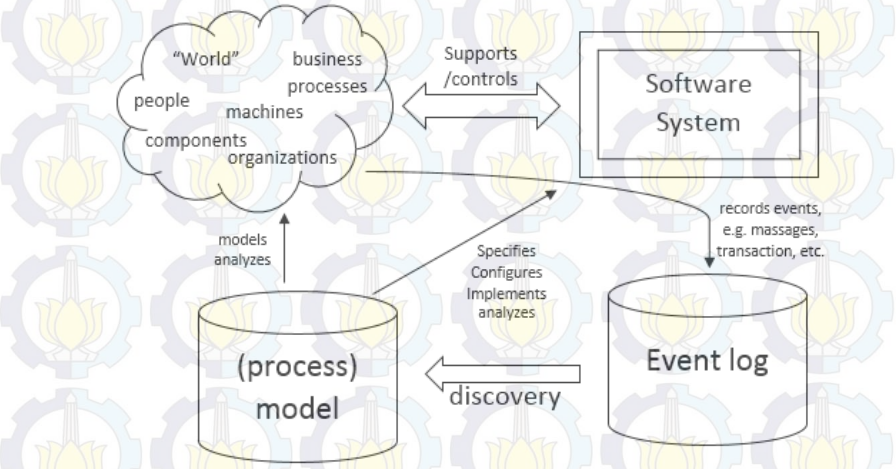
Rahadian Dustrial Dewandono, Tohari Ahmad, Mohammad Farid Naufal, dan Fernandes Sinaga, 2015) (Solichul Huda, Riyanarno Sarno, Tohari Ahmad, Heru Agus Santoso, 2014).

- *Enhancement*

Salah satu teknik *process mining* yang bertujuan untuk mengembangkan proses bisnis yang sudah ada. Proses model dari *event log* dianalisis untuk mendapatkan bagian proses yang masih dapat atau perlu dikembangkan.

2.4 *Process Discovery*

Process discovery merupakan salah satu proses yang paling menantang dari rangkaian *process mining*. Tujuan dari proses ini adalah untuk membentuk model dengan cara menggali informasi dari data yang tercatat dalam suatu *event log* (Goedertier, De Weerdt, Martens, Vanthienen, & Baesens, 2011). Dalam struktur proses bisnis, model dapat dianggap sebagai *graph* untuk mengandung satu set *node* dihubungkan dengan *edge* (Riyanarto Sarno, Hari Ginardi, Endang Wahyu Pamungkas, Dwi Sunaryono, 2013).

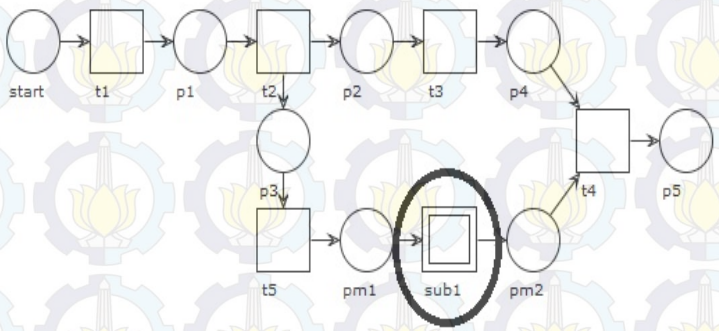


Gambar 2.1 Skema *Process Discovery*

Adapun algoritma yang digunakan dalam *Process Discovery* adalah algoritma *alpha miner, alpha plus miner, alpha plus plus miner, heuristics miner* (van der Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, 2011). Setiap algoritma memiliki pendekatan yang berbeda-beda dalam menganalisis proses yang terjadi. Dalam Tugas Akhir ini akan mengembangkan algoritma *heuristics miner* untuk melakukan *process discovery* karena algoritma ini merupakan algoritma yang dapat menangani adanya *noise* dalam *event log.*

2.5 *Top Level* Proses Model

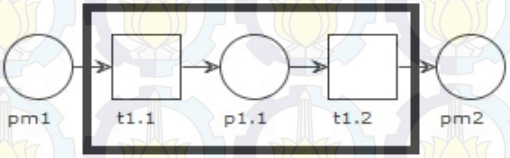
Top level proses model $\sum TPM$ adalah salah satu jenis dari perluasan *Petri nets* dengan *abstract transitions* (Qingtian Zeng, Hua Duan, Cong Liu, 2015), yaitu terdapat dua jenis transisi; yang pertama yaitu merepresentasikan aktivitas-aktivitas normal, dan yang kedua yaitu merepresentasikan *abstract procedures*. Untuk membedakannya dari *normal transitions*, sebuah *abstract transition* digambarkan dengan sebuah *double rectangle*. Sebagai contoh, sebuah *top level Petri net* ditunjukkan oleh Gambar 2.2, dimana sub1 adalah sebuah *abstract transition*.



Gambar 2.2 Contoh *Top Level* Proses Model

2.6 *Bottom Level* Proses Model

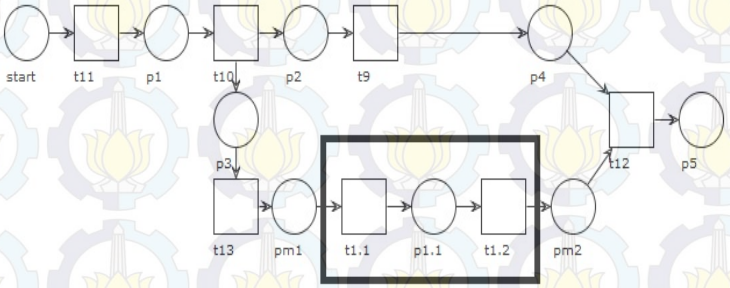
Bottom level proses model $\sum BPM$ didefinisikan berbeda dari *top level* proses model karena *bottom level* proses model tidak memiliki *abstract transition* (Qingtian Zeng, Hua Duan, Cong Liu, 2015). Oleh karena itu, aturan untuk *bottom level* proses model sama dengan standar *Petri net*. Sebagai contoh, sebuah *bottom level Petri net* ditunjukkan pada Gambar 2.3.



Gambar 2.3 Contoh *Bottom Level* Proses Model

2.7 Proses *Refinement of Petri Nets*

Proses *refinement* bertujuan untuk memperbaiki *abstract transition* pada *top level* proses model menjadi *bottom level* proses model. Struktur dari *bottom level* proses model akan menggantikan *abstract transition* dan bagian-bagian lain dari *top level* proses model (Qingtian Zeng, Hua Duan, Cong Liu, 2015). *Top level* proses model setelah proses *refinement* ditunjukkan pada Gambar 2.4. Hasilnya, *top level* proses model menjadi model *Petri Net* standar setelah mengganti sub1 dengan *bottom level* proses model.



Gambar 2.4 Contoh Proses Model setelah Proses *Refinement of Petri Nets*

2.8 *Framework Top-to-*Down Proses *Mining*

Framework untuk *top-to-down* proses *mining based on refinement of petri nets* terdiri dari empat langkah, yaitu (Qingtian Zeng, Hua Duan, Cong Liu, 2015):

Merekam *Event Log*.

Sementara alur kerja sistem berjalan pada beberapa *server* perusahaan, setiap *server* dapat merekam *event log* untuk setiap aktivitas dan menyimpannya ke dalam *database event log*. *Event log* yang sedang dieksekusi dikumpulkan dari *server multi-souce* yang digunakan untuk proses *mining top-to-down*.

Proses *Mining* dari *Top level* berdasarkan *Workflow Event Log*.

Menggunakan *event log* yang dikumpulkan, algoritma proses *mining top level* proses model mencoba untuk menemukan proses model *top level* dari alur kerja sistem. Hasil *mining* dapat direpresentasikan dalam bentuk formal/standar dari *Petri nets* dengan *abstract transitions*. Untuk melindungi keamanan, rincian isi *abstract transitions* tidak dapat diperoleh dalam langkah ini.

Proses *Mining* dari *Bottom level* berdasarkan *Workflow Event Log*.

Menggunakan *event log* yang dikumpulkan, algoritma proses *mining bottom level* proses model untuk menemukan proses model rinci untuk setiap *abstract procedure*. *Bottom level* proses model yang diperoleh ditampilkan dalam bentuk standar *Petri nets* tanpa *abstract transitions*.

Integrasi Model berdasarkan Proses *Refinement of Petri Nets.*

Setelah mendapatkan kedua proses model *top level* dan proses model *bottom level* dari *multi-source event log*, proses *refinement of petri nets* digunakan untuk memperbaiki *abstract transitions* pada proses model *top level* sesuai proses model *bottom level* untuk mendapatkan model yang terintegrasi dari seluruh alur kerja sistem.

2.9 *Causal Net* (C-Net)

Causal Net (C-Net) adalah suatu *graph* yang menggambarkan suatu proses bisnis, dimana *nodes* dari *graph* menggambarkan aktivitas yang terjadi pada proses bisnis dan *arcs* atau *edges* pada

gaph menggambarkan hubungan antar aktivitas yang ada pada proses bisnis. Dalam C-Net setiap aktivitas memiliki serangkaian *input bindings* dan *output bindings* (van der Aalst, Adriansyah, & van Dongen, Causal Nets: A Modeling Language Tailored Towards Process Discovery, 2011). Salah satu contoh dari *causal net* adalah sebagai berikut:

Gambar 2.5 Contoh *Causal Net*

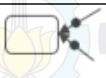
Pada contoh C-Net pada Gambar 2.5, aktivitas a memiliki *input bindings* kosong sehingga a merupakan *start activity* dan a memiliki *output bindings:* {b}, ini berarti setelah aktivitas a diikuti oleh aktivitas b. Aktivitas b memiliki *input bindings:*{a} dan *output bindings:* {c,d}, karena *output bindings* yang dimiliki b lebih dari satu maka menggunakan *split model* yang dimiliki C-Net, *input bindings* dan *output bindings* untuk C-Net begitu seterusnya sehingga akan menghasilkan *Causal Matrix.* Untuk Gambar 2.5 *Causal Matrix*-nya adalah sebagai berikut:

Tabel 2.1 *Causal Matrix* Gambar 2.5

Input	Aktivitas	Output
{}	a	b
a	b	c,d
b	c	e
b	d	e
c, d	e	f
e	f	{}

Bentuk model *split* dan *join* yang dimiliki C-Net adalah sebagai berikut (van der Aalst, Adriansyah, & van Dongen, Causal Nets: A Modeling Language Tailored Towards Process Discovery, 2011):

Tabel 2.2 C-Net *Split* dan *Join*

C-Net Split Model	C-Net Join Model
	
	
	

2.10 Jenis *Split* dan *Join*

Dalam suatu alur proses bisnis dimungkinkan untuk adanya percabangan sehingga membentuk suatu pilihan ataupun suatu proses paralel. Proses paralel sendiri terdapat dua jenis *split join* AND dan *split join* OR, sedangkan untuk pilihan menggunakan *split* dan *join* XOR (Weske, 2011). Sifat-sifat untuk masing-masing *split* dan *join* adalah sebagai berikut:

- *Single Choice* XOR

Single Choice XOR terjadi jika titik dalam proses alur kerja di mana satu cabang dibagi menjadi dua atau lebih tetapi trace hanya dapat memilih salah satu cabang saja. Semua jenis pemodelan dapat memodelkan XOR oleh karena itu hampir semua algoritma dapat memodelkan XOR (Weske, 2011).

- *Parallel* AND

Parallel AND terjadi jika *parallel split pattern* muncul. *Parallel split pattern* didefinisikan sebagai mekanisme yang memungkinkan dua kegiatan yang berbeda dilakukan secara bersamaan. Sifat dasar dari pola ini sendiri adalah semua aktivitas yang ada di percabangan harus dijalankan, baik itu dijalankan secara bersamaan atau secara bergantian (Weske, 2011).

- *Conditional* OR

Conditional OR digunakan ketika *multiple choice pattern* muncul. *Multiple choice pattern* pemilihan satu atau lebih aktivitas dalam percabangan untuk dijalankan. Dalam *multiple choice pattern* satu aktivitas dapat dijalankan sendiri tanpa harus menjalankan aktivitas lain yang ada dipercabangan, atau juga dapat menjalankan beberapa aktivitas baik secara bersamaan maupun tidak (Weske, 2011).

Dalam *process mining* banyak algoritma *process discovery* yang tidak dapat mngidentifikasi atau memodelkan secara benar suatu *event log* yang mengandung *multiple choice pattern* (Riyanarto Sarno, Putu Linda Indita Sari, Hari Ginardi, Dwi Sunaryono, Imam Mukhlash, 2013). Hal ini dikarenakan kebanyakan algoritma terutama algoritma yang menggunakan pemodelan dengan *Petri Net* tidak memiliki aturan dalam memodelkan *conditional* OR, hal ini dikarenakan bentuk pemodelan dari *Petri Net* sendiri memang tidak dapat memodelkan bentuk *split* dan *join* OR (Weske, 2011).

2.11 Algoritma *Heuristics Miner*

Algoritma *Heuristics miner* merupakan salah satu algoritma yang digunakan untuk melakukan proses *discovery* dalam proses *mining*. Algoritma *Heuristics miner* menggunakan model C-Net dalam merepresentasikan model proses bisnis yang dihasilkan. Algoritma ini menggunakan frekuensi dari suatu hubungan aktivitas dan urutannya untuk membangun sebuah model proses bisnis. Ide dasar dari algoritma ini adalah suatu urutan atau hubungan dari aktivitas yang jarang terjadi dalam *event log* tidak harus dimasukkan dalam model. Penggunaan frekuensi dalam pembentukan model membuat algoritma ini lebih akurat dibandingkan dengan algoritma yang lainnya. Selain itu juga membuat algoritma ini dapat menangani *noise*, walaupun belum ada algoritma yang data sepenuhnya menangani masalah *noise* (van der Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, 2011).

Dalam algoritma *heuristics miner* tedapat tiga langkah utama dalam pembentukan model dari proses bisnis (A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros), yaitu:

➨ *Mining Dependency Graph*

Langkah awal dari algoritma *heutistics miner* adalah membuat suatu *dependency graph* yang merupakan hasil dari perhitungan frekuensi-frekuensi dari hubungan antar aktivitas yang ada. Pertama yang dilakukan adalah menghitung frekuensi dari hubungan masing-masing aktivitas yang terhubung. Kemudian dari frekuensi tersebut akan menentukan hubungan ketergantungan dari masing-masing aktivitas atau dengan kata lain dapat dikatakan aktivitas yang dapat saling dihubungkan.

Perhitungan dari *dependency measure* yang digunakan untuk menentukan hubungan antar aktivitas dalam model dapat dilihat pada Persamaan 2.1.

$$X =>_w Y = \left(\frac{|X>_w Y| - |Y>_w X|}{|X>_w Y| + |Y>_w X| + 1}\right) \quad (2.1)$$

Keterangan:

$X =>_w Y$: nilai *dependency* dari aktivitas X ke Y.

$|X >_w Y|$: frekuensi aktivitas X yang diikuti secara langsung oleh Y.

$|Y >_w X|$: frekuensi aktivitas Y yang diikuti secara langsung oleh X.

Hasil dari Persamaan 2.1 menghasilkan suatu matriks yaitu matriks *dependency measure* yang akan digunakan untuk membentuk *dependency graph.* A. Weijters dan Sofie Cnude (A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros) (Sofie De Cnudde, Jan Claes, Geert Poels, 2014) menjelaskan bahwa dalam pemilihan *dependency measure* untuk pembentukan *dependency graph*, *heuristics miner* menggunakan beberapa jenis *threshold* yaitu:

a. *Relative-to-best threshold* (RBT)

Threshold ini menunjukkan bahwa suatu *edge* atau hubungan akan digunakan (yaitu untuk memasukkan *edge*/hubungan antar aktivitas ke dalam *control-flow network*) jika perbedaan antara nilai *dependency measure* yang dihitung untuk *edge* dan nilai terbesar dari *dependency measure* yang ada pada matriks *dependency graph* lebih rendah dari nilai parameter ini.

b. Positive observations threshold (POT)

Threshold ini mengontrol jumlah minimum berapa kali aktivitas memiliki ketergantungan hubungan dengan aktivitas lain. Relasi dianggap ketika banyak frekuensi hubungan ini berada di atas nilai atau sama dengan parameter ini.

c. Dependency threshold (DT)

Threshold ini berguna untuk mengabaikan semua hubungan yang nilai *dependency measure* berada dibawah nilai parameter. Dengan kata lain nilai *dependency measure* yang digunakan harus lebih besar atau sama dengan nilai *dependency threshold*.

Salah satu kekurangan dari algoritma *heuristics miner* adalah *threshold* yang tidak ditentukan sehingga pengguna harus menentukannya sendiri. Dengan kata lain, pengguna harus mengerti betul dengan cara kerja algoritma ini agar dapat menentukan *threshold* yang dapat menghasilkan proses bisnis yang ideal. Hal ini mengakibatkan pengguna awam kesulitan dalam menggunakan algoritma ini. Oleh karena itu, di dalam Tugas Akhir ini akan memodifikasi algoritma ini sehingga pengguna awam juga dapat menggunakan algoritma ini.

➨ *Mining Short Loop (Length One Loop* dan *Length Two Loop)*

Dalam proses, dimungkinkan untuk menjalankan kegiatan yang sama beberapa kali. Jika ini terjadi, ini biasanya mengacu pada *Loop* yang terjadi pada proses model. Untuk *Long Loop* sudah dapat diatasi dengan menggunakan *dependency measure* tetapi untuk *Short Loop* diperlukan pengecekan tersendiri.

Rumus untuk *Length One Loop:*

$$X =>_w X = \left(\frac{|X>_w X|}{\max\{|X>_w A|| \; A \in e\}}\right) \quad (2.2)$$

Keterangan:

$X =>_w X$: nilai *dependency Length One Loop*

$|X >_w X|$: frekuensi aktivitas X yang diikuti secara langsung oleh X.

$\max\{|X >_w A|| \; A \in e\}$: frekuensi aktivitas X yang diikuti oleh aktivitas A dimana aktivitas A merupakan bagian dari aktivitas yang ada di *event log*.

Hasil dari Persamaan 2.2 menghasilkan suatu matriks yaitu matriks *length one loop* yang digunakan untuk membentuk *one loop* pada *dependency graph.*

Rumus untuk *Length Two Loop:*

$$X =>_{2w} Y = \left(\frac{|X \gg_w Y|+|Y \gg_w X|}{|X \gg_w Y|+|Y \gg_w X|+1}\right) \quad (2.3)$$

Keterangan:

$X =>_{2w} Y$: nilai *dependency Length Two Loop*

$|X \gg_w Y|$: frekuensi dari *trace* dalam bentuk XYX yang muncul dalam *event log*

$|Y \gg_w X|$: frekuensi dari *trace* dalam bentuk YXY yang muncul dalam *event log*

Hasil dari Persamaan 2.3 menghasilkan suatu matriks yaitu matriks *length two loop* yang digunakan untuk membentuk *two loop* pada *dependency graph.*

- *Mining Process Parallel*

Untuk mendapatkan aktivitas paralel pada algoritma ini menggunakan *parallel measure* untuk menentukan suatu aktivitas tersebut paralel atau tidak. Sebelum menghitung nilai dari

parallel measure ditentukan terlebih dahulu proses mana saja yang paralel dengan menggunakan *causal matrix*.

Jika *input* dan *output*-nya lebih dari satu aktivitas maka ada kemungkinan paralel, sehingga nilai *parallel measure* dari aktivitas tersebut dihitung. Persamaan 2.4 merupakan rumus untuk penentuan dari *parallel measure.*

$$X =>_w Y \wedge Z = \left(\frac{|Y>_w Z|+|Z>_w Y|}{|X>_w Y|+|X>_w Z|+1}\right) \quad (2.4)$$

Keterangan:
$X =>_w Y \wedge Z$: *parallel measure* antara Y dan Z dimana *split* terjadi di X.
$|Y >_w Z|$: frekuensi aktivitas Y yang diikuti secara langsung oleh Z.
$|Z >_w Y|$: frekuensi aktivitas Z yang diikuti secara langsung oleh Y.
$|X >_w Y|$: frekuensi aktivitas X yang diikuti secara langsung oleh Y.
$|X >_w Z|$: frekuensi aktivitas X yang diikuti secara langsung oleh Z.

Kekurangan dari algoritma *heuristics miner* yang lainnya adalah hanya dapat men-*discover* jenis *split* dan *join* XOR dan AND, sementara untuk *conditional* OR tidak bisa walaupun sebenarnya C-Net dapat memodelkan bentuk *split* dan *join* OR. Oleh karena itu, di dalam Tugas Akhir ini akan memodifikasi algoritma ini sehingga dapat men-*discover split* dan *join OR.*

2.12 Aktivitas *Event Log* sebagai Interval Waktu

Tidak hanya mempertimbangkan waktu mulai tetapi juga waktu selesai dari aktivitas di dalam *event log*, algoritma *Heuristics Miner* juga mengenali setiap aktivitas sebagai *instantaneous event*. Fungsi asli dari algoritma *Heuristics Miner* dimodifikasi sehingga dapat menangani aktivitas berdasarkan interval waktu. Untuk itu, di dalam Tugas Akhir ini diberikan

definisi baru dari *direct relation* dalam konteks interval waktu (Andrea Burattin, Alessandro Sperduti, 2010).

Suatu *single event* direpresentasikan sebagai suatu aktivitas, sehingga terdapat sebuah definisi bahwa X >w Y iff $\exists \boldsymbol{\sigma} = t_1 \ldots, t_n$ dan $a \in \{1, \ldots, n-1\}$ sehingga $\boldsymbol{\sigma} \in E$, $t_i = X$ dan $t_{i+1} = Y$. Kami harus memodifikasi definisi ini untuk menangani aktivitas yang bertindak sebagai interval waktu. Langkah pertama, terdapat sebuah *event log* E, maka kita mendefinisikan *NameOfActivity[E],* nama aktivitas di dalam *event log,* dan *ClassTypeOf[E],* jenis kelas *event log* (*start* atau *end*). *Direct relation* yang baru antara dua aktivitas $X \geqslant \boldsymbol{w}$ Y didefinisikan lebih lengkap di Definisi 1:

Definisi 1 (*Direct relation,* $\geqslant$). Aktivitas X dan Y menjadi dua aktivitas yang bertindak sebagai interval waktu (bukan *instantaneous activity*) di dalam *event log* E, maka

$X \geqslant w$ Y iff $\exists \boldsymbol{\sigma} = t_1, \ldots, t_n$ dan
$a \in \{2, \ldots, n-2\}$, $b \in \{3, \ldots, n-1\}$
sehingga $\boldsymbol{\sigma} \in E$, $t_i = X_{end}$ dan $t_j = Y_{start}$ dan
$\forall \boldsymbol{k}$ sehingga $a < c < b$
memiliki definisi bahwa $\text{ClassTypeOf}[t_k] \neq \text{start}$

Berdasarkan definisi tersebut, dapat dikatakan bahwa terdapat beberapa kondisi yang harus dipenuhi oleh dua aktivitas sehingga mereka berada dalam *direct relation;* kondisi akhir aktivitas pertama terjadi sebelum dimulainya aktivitas kedua dan antara dua aktivitas, tidak ada aktivitas lain yang mulai. Selain itu, diperkenalkan juga konsep baru yaitu relasi paralel antara dua aktivitas. Terdapat aktivitas X dan Y dengan *instantaneous activities*, ketika aktivitas X dan Y dieksekusi tanpa melihat urutan, misalnya kadang-kadang X dieksekusi sebelum Y dan kadang-kadang Y dieksekusi sebelum X, $(X >_W Y) \wedge (Y >_W X)$, maka aktivitas X dan Y dianggap sebagai paralel. Definisi ini sebenarnya mungkin tampak aneh dan perlu dianalisis lebih jauh, tetapi tanpa menerapkan definisi ini sulit untuk mengekspresikan

relasi paralel tanpa gagasan durasi. Definisi ini lebih mudah dan lebih intuitif dalam konteks baru sebagai berikut:

Definisi 2 (*Parallelism relation*, k). Aktivitas X dan Y menjadi dua aktivitas yang bertindak sebagai interval waktu (bukan *instantaneous activity*) di dalam *event log* E, maka

$X \|_W Y$ iff $\sigma \in = t_1, \ldots, t_n$ dan a, b, e, f $\in \{1, \ldots, n\}$
dengan $t_i = X_{start}$, $t_j = X_{end}$ dan $t_u = Y_{start}$, $t_v = Y_{and}$
sehingga $e < a < f \vee a < e < b$

Seperti yang sudah dibahas sebelumnya, lebih mudah dan lebih intuitif jika dua aktivitas yang tumpang tindih atau jika satu aktivitas berisi aktivitas yang lain, definisi ini menunjukkan dua aktivitas sebagai hubungan paralel. (Andrea Burattin, Alessandro Sperduti, 2010) mengubah pengertian hubungan paralel antara dua aktivitas dan modifikasi rumus algoritma untuk *statistical dependency* dan untuk menentukan jenis hubungan paralel; *single choice* XOR atau *parallel* AND. Setelah dimodifikasi, rumus barunya menjadi:

$$X =>_w Y = \left(\frac{|X \geqslant_w Y| - |Y \geqslant_w X|}{|X \geqslant_w Y| + |Y \geqslant_w X| + 2|X||wY| + 1}\right) \quad (2.5)$$

$$X =>_w (Y \wedge Z) = \left(\frac{|Y \geqslant_w Z| + |Z \geqslant_w Y| + 2|Y||wZ|}{|X \geqslant_w Y| + |X \geqslant_w Z| + 1}\right) \quad (2.6)$$

di mana notasi $|Y||wZ|$ adalah jumlah (per *case* aktivitas) aktivitas Y dan Z yang memiliki relasi paralel dalam sebuah *event log* E. (Andrea Burattin, Alessandro Sperduti, 2010) memperkenalkan aktivitas dalam *succession relation* jika aktivitas-aktivitas tersebut tumpang tindih di dalam *event log*, sehingga pada proses model yang ditemukan, hubungan aktivitas tersebut paralel. Dengan rumus baru, diperoleh "*backward compatibility*" dengan memodifikasi algoritma asli dari *Heuristics Miner*. Menggunakan persamaan 2.5, jika dua aktivitas X dan Y

tidak tumpang tindih, mereka tidak dalam relasi paralel, maka $|X||wY| = 0$. Rumus ini dapat digunakan jika aktivitas dinyatakan sebagai interval waktu dan bersifat *instantaneous*. Tanpa kehilangan manfaat dari algoritma asli *Heuristics Miner*, rumus ini juga dapat meningkatkan kinerja algoritma menjadi lebih baik.

2.13 Evaluasi Kualitas *Fitness* Suatu Algoritma

Dalam literatur proses *mining*, tidak ada *framework* umum yang digunakan untuk mengevaluasi proses model yang ditemukan. Hanya dinyatakan bahwa untuk membandingkan dan mengevaluasi proses model yang ditemukan dapat menggunakan hasil proses model yang ditemukan dari algoritma-algoritma berbeda dan kualitas *fitness* dari proses model yang ditemukan digunakan untuk mengevaluasi sebagai keputusan akhir jika salah satu metode lebih baik daripada metode yang lain. Beberapa kriteria untuk mengevaluasi kualitas proses model yang ditemukan adalah: (i) kebugaran, (ii) presisi, (iii) generalisasi, dan (iv) struktur. Sebuah proses model dengan *fitness* yang baik menunjukkan "kecocokan" dari proses model yang ditemukan dengan "realita". Sebuah proses model dikatakan memiliki *fitness* sempurna jika semua *trace* di *event log* dapat diwakili oleh proses model dari awal sampai akhir. Tetapi jika banyak *trace* di *event log* tidak dapat diwakili oleh proses model dari awal sampai akhir, maka proses model disebut memiliki *fitness* yang buruk. Presisi didefinisikan sebagai proses model yang *overfitted*, deskripsi dari generalisasi adalah proses model yang *underfitted*, sementara definisi struktur berkaitan dengan kualitas pragmatis proses model (Sofie De Cnudde, Jan Claes, Geert Poels, 2014).

Dalam algoritma *Heuristics Miner*, kualitas *fitness* dapat dihitung dengan menggunakan dua rumus: *parsing measure PM* dan *continuous parsing measure CPM*. Pertama, *parsing measure* (PMw) didefinisikan sebagai berikut:

$$\text{PMw} = \frac{c}{t} \tag{2.7}$$

Keterangan:
PMw : *parsing measure*
c : jumlah dari *traces* yang berhasil diuraikan dengan benar
t : total jumlah *traces* di dalam *event log*

Kedua, rumus untuk menghitung *continuous parsing measure* (CPMw) dengan rincian yang lebih tinggi (yaitu mengidentifikasi aktivitas daripada *trace*) yaitu sebagai berikut:

$$\text{CPMw} = \frac{1}{2}\frac{(e-m)}{e} + \frac{1}{2}\frac{(e-r)}{e} \quad (2.8)$$

Keterangan:
CPMw : *continuous parsing measure*
e : total jumlah dari aktivitas di dalam *event log*
m : jumlah dari *input* yang hilang (misalnya aktivitas yang tidak berhasil diurai di *trace*)
r : jumlah dari *output* yang tersisa (misalnya aktivitas yang mengikuti hingga akhir *trace*)

2.14 Evaluasi *Completeness* dari Proses Model

Completeness didefinisikan sebagai dependensi penting yang tidak ditampilkan dalam proses model yang ditemukan. Masalah *completeness* fokus pada perhitungan *Length Two Loop (L2) dependency measure*. Sedangkan di Bagian 4 (Sofie De Cnudde, Jan Claes, Geert Poels, 2014) membahas mengenai kesalahan dari *Length Two Loop (L2) dependency measure*. Dengan percobaan menggunakan algoritma *Modified Time-Based Heuristics Miner,* kita dapat menyimpulkan bahwa jika salah satu dari kondisi berikut terpenuhi, maka *Length Two Loop (L2) dependency measure* antara dua aktivitas X dan Y tidak dihitung. Tiga kondisi tersebut adalah:

- X adalah *Length One Loop* dan $|X >>_W Y|$ lebih tinggi daripada *positive observation threshold* (POT) atau,
- Y adalah *Length One Loop* dan $|Y >>_W X|$ lebih tinggi daripada *positive observation threshold* (POT) atau,
- semua kondisi di atas.

2.15 Evaluasi *Validity* dari Proses Model Semantik

Validity fokus pada proses model semantik yang dihasilkan oleh algoritma *Modified Time-Based Heuristics Miner*. *Validity* dari proses model semantik tidak dianggap valid karena tidak mengikuti aturan-aturan formal yang didefinisikan dalam (Sofie De Cnudde, Jan Claes, Geert Poels, 2014) (A.J.M.M. Weijters, J.T.S. Ribeiro, 2010). Valid berarti proses model dibuat dengan mendefinisikan perubahan dari *dependency graph* ke proses model semantik. Kita perlu mengetahui dua langkah untuk membuat proses model semantik. Pertama, membuat *dependency graph* menjadi *causal net* dan yang kedua, membuat *causal net* menjadi proses model semantik.

Untuk mengatasi masalah ini, pertama, pembuatan *dependency graph* menjadi *causal net* harus benar untuk menafsirkan hubungan paralel AND-, OR- dan XOR- dari *input* dan *output* dalam *dependency graph*. Selanjutnya, algoritma digunakan untuk membuat perbedaan hubungan *sequence* dan hubungan paralel (AND-, OR- atau XOR-) antar aktivitas. *Gateway* dipilih dan ditambahkan tergantung pada sifat hubungan relasi (*sequence* atau paralel). Akhirnya, *gateway* yang terpilih menghubungkan satu aktivitas ke aktivitas lainnya dengan benar.

2.16 Evaluasi Optimasi *Time* dan *Cost*

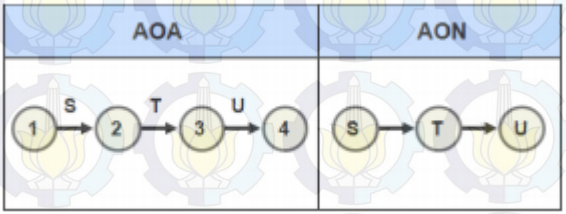
Dalam buku Tugas Akhir ini juga dikembangkan model *tradeoff time* dan *cost* untuk manajemen proyek dengan menggunakan *non-linear programming*. Sebuah model dasar dengan hubungan biaya kuadrat dikembangkan dan kemudian diperluas untuk mencakup pertimbangan operasional lainnya. Penggunaan fungsi non-linear untuk menggambarkan hubungan

time-cost dalam lingkungan penjadwalan proyek memungkinkan representasi yang lebih akurat dari pengaturan operasional daripada dengan menggunakan *linear programming* (Richard F. Deckro, John E. Hebert, William A. Verdini, Per Henning Grimsrud, Satya Venkateshwar, 1994). Langkah pengerjaan optimasi *time-cost* akan dijelaskan pada bab selanjutnya.

2.17 *Critical Path Method* (CPM)

Critical Path Method (CPM) adalah teknik untuk menganalisa jaringan kegiatan/aktivitas-aktivitas ketika menjalankan proyek dalam rangka memprediksi durasi total. *Critical path* sendiri dalam sebuah proyek adalah jalur terpanjang dalam *network diagram* (Prawira, 2014).

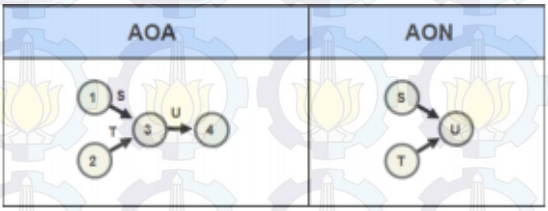
Critical path didapatkan dari sebuah diagram jaringan (*network diagram*) yang memperlihatakan hubungan dan urutan aktitivas-aktivitas dalam suatu proyek. Secara umum *network diagram* digambarkan menggunakan *activity on node* (AON) dan *activity on arrow* (AOA) (Larson & Gray, 2006). Pada AON, aktivitas proyek direpresentasikan dengan titik (*node*), sementara pada AOA, aktivitas kegiatan direpresentasikan dengan panah (*arrow*). Aktivitas proyek yang mendahului/menjadi syarat dillakukan aktivitas lainnya disebut *predesesor*. Gambar 2.6 sampai dengan Gambar 2.8, menunjukkan hubungan aktivitas dalam proyek menggunakan AOA dan AON.



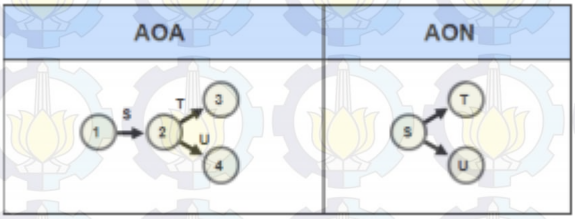
Gambar 2.6 Bentuk Serial

Pada Gambar 2.6 (bentuk serial), diperlihatkan tiga aktivitas proyek yaitu S, T, dan U. Pada gambar tersebut ditunjukkan bahwa aktivitas S merupakan *predesesor* bagi

aktivitas T, sementara aktivitas T menjadi *predesesor* bagi aktivitas U. Pada Gambar 2.7 (bentuk *join*), diperlihatkan bahwa aktivitas S dan T menjadi *predesesor* bagi aktivitas U, atau dengan kata lain dapat dikatakan bahwa aktivitas U bisa dilaksanakan jika aktivitas S dan T sudah dilaksanakan terlebih dahulu. Pada Gambar 2.8 (bentuk *split*) memperlihatkan aktivitas S menjadi *predesesor* bagi aktivitas T dan U. Hal ini menggambarkan bahwa aktivitas T dan U bisa dilaksanakan jika aktivitas S telah dilaksanakan terlebih dahulu.



Gambar 2.7 Bentuk *Join*



Gambar 2.8 Bentuk *Split*

Pada Tugas Akhir ini digunakan jenis AON untuk menentukan *critical path* pada jaringan atau proses bisnis yang digunakan yang berguna untuk menentukan durasi dari proses bisnis. Selain digunakan untuk menentukan durasi dari proses bisnis salah satu metode CPM yaitu *crashing project* digunakan sebagai dasar untuk optimasi durasi. Dasar yang digunakan yaitu perhitungan dari perhitungan *cost slope* dan *time slope* yang akan digunakan untuk menjadi konstanta dalan model matematika dari *non-linear programming*.

BAB III
METODE PEMECAHAN MASALAH

Pada bab ini akan dibahas mengenai metodologi pemecahan masalah yang digunakan sebagai dasar solusi dari pembuatan Tugas Akhir. Metodologi tersebut menerangkan langkah demi langkah proses hingga dapat menghasilkan proses model dari suatu *event log*. Untuk permasalahan men-*discover* proses model dengan *top-to-down* proses *mining* dibahas pada subbab 3.2 sampai dengan subbab 3.3. Sedangkan untuk permasalahan men-*discover* proses model dengan *modified time-based heuristics miner* hingga optimasi waktu, biaya dan *resource* akan dibahas pada subbab 3.4 sampai dengan subbab 3.13.

3.1 Cakupan Permasalahan

Permasalahan utama yang diangkat dalam pembuatan Tugas Akhir ini adalah menentukan *top level* dari proses model dengan *abstract transitions* dan *bottom level* dari proses model. Setelah *top level* dan *bottom level* dari proses model ditemukan, kemudian digabungkan dengan proses *refinement of petri nets* sehingga *abstract transitions* dari *top level* diganti menjadi *bottom level* proses model sehingga membentuk *workflow* standar dari *petri nets*. Selain itu, Tugas Akhir ini juga memodifikasi dan menggabungkan algoritma *heuristics miner for time intervals* dan *original heuristics miner* menjadi *modified time-based heuristics miner*. Algoritma *heuristics miner for time intervals* merupakan salah satu algoritma *process discovery* yang sulit untuk digunakan, hal ini dikarenakan seperti pada algoritma *heuristics miner* terdapat *threshold-threshold* yang harus ditentukan oleh pengguna. Sementara pengertian dari *threshold* itu sendiri tidak semua pengguna mengerti. Oleh karena itu, dalam Tugas Akhir ini yang dimaksud dengan memodifikasi *heuristics miner algorithm* adalah menentukan nilai dari *threshold* yang ada. Selain penentuan *threshold* yang dimaksud dalam memodifikasi algoritma *heuristics miner* disini adalah mengubah rumus untuk

mining parallel activity dan penambahan beberapa interval sehingga dapat men-*discover split* dan *join* OR. Sehingga dalam modifikasi algoritma *heuristics miner* terdapat dua tahap utama yaitu penentuan nilai *threshold* dan perubahan pada rumus *mining parallel activity.* Selain memodifikasi algoritma *heuristics miner*, Tugas Akhir ini juga menggunakan prinsip interval waktu dari *heuristics miner for time intervals.*

Permasalahan kedua adalah membandingkan kualitas *fitness* dari metode-metode yang digunakan, yaitu *modified time-based heuristics miner, heuristics miner for time intervals, original heuristics miner,* dan *process model discovery based on activity lifespan.* Permasalahan selanjutnya adalah mengevaluasi *completeness* dan *validity* dari proses model hasil dari metode yang diusulkan dan permasalahan terakhir adalah bagaimana mengoptimasi waktu *makespan* dari suatu proses bisnis dan biaya optimasi waktu yang diperlukan (total *crash cost*). Pengoptimasian waktu atau *makespan* dari proses bisnis dan biaya yang dibutuhkan (total *crash cost*). Bentuk dari *event log* yang digunakan adalah *double timestamp.* Langkah awal dalam memecahkan masalah ini adalah menentukan durasi untuk setiap aktivitas yang ada dalam proses bisnis. Selanjutnya untuk pengotimalan waktu atau *makespan* dan biaya yang dibutuhkan diperlukan *crash time* (waktu yang dikurangkan dari normal waktu aktivitas) dan *crash cost* (biaya yang diperlukan untuk pemampatan per aktivitas), maka langkah selanjutnya adalah penentuan durasi (*normal duration*), *crash duration,* dan *crash cost* per aktivitas, yang terakhir adalah pemodelan bentuk non-linear untuk mendapatkan waktu dan biaya paling optimal.

Proses pemodelan proses bisnis tersebut membutuhkan data atau *event log* proses bisnis yang sudah ada dan berjalan. Dari *event log* yang digunakan sebagai masukan dilakukan serangkaian proses hingga menghasilkan keluaran yang diharapkan. Gambar 3.1 merupakan alur proses pengerjaan Tugas Akhir secara sederhana.

Gambar 3.1 Alur Proses Pengerjaan Tugas Akhir

3.2 Algoritma *Top-to-Down Process Mining Based on Refinement of Petri Nets from Multi-source Event Log*

Dalam rangka untuk mendapatkan proses model kompleks yang lengkap dan benar untuk proses bisnis di kehidupan nyata dari *Multi Organizational Collaboration*, berikut adalah langkah-langkah yang akan dilakukan (Qingtian Zeng, Hua Duan, Cong Liu, 2015):

1. Men-*discover Top Level* dengan Proses *Mining*

Pendekatan proses *mining* untuk menemukan proses model dari *top level* yang terdiri dari dua komponen fungsional disajikan dalam Algoritma 1 dan Algoritma 2. Algoritma 1 digunakan untuk memperoleh hubungan *dependency* antar aktivitas dan algoritma 2 menggunakan hubungan ini sebagai *input* untuk membangun proses model akhir dari *top level*.

Algoritma 1. Untuk memperoleh AS_i.PreSet dan AS_i.PostSet dari masing-masing aktivitas AS_i di dalam *event log*

Input: RLogs– sekumpulan event log dari alur kerja sistem, yang terdiri dari beberapa *case* yang dieksekusi.

Output: (AS_i, AS_i.PreSet, AS_i.PostSet, AS_i.ReceivedMessage, AS_i.SentMessage).

1: For each $AS_i \in$ AssignmentSet(RCase) Do
  (1.1) ASi.PreSet ← ∅
  (1.2) ASi.PostSet ← ∅
End do

2: For each ASi ∈ AssignmentSet(RCase) Do
ASi.PostSet ← ASi.PostSet ∪ $\bigcap_{1 \le j \le |RLog|}$ PostSet(ASi,RCasej);

3: For each ASi ∈ AssignmentSet(RCase) Do
For each ASj ∈ AssignmentSet(RCase) Do
ASi.PreSet ← ASi.PreSet ∪ {ASj|ASi ∈ ASj.PostSet};
End do
End do

4: **return** (AS_i, AS_i.PreSet, AS_i.PostSet, AS_i.ReceivedMessage, AS_i.SentMessage).

Algoritma 2. Untuk memperoleh *top level* proses model $\sum TPM$.

Input: {(AS_i, AS_i.PreSet, AS_i.PostSet, AS_i.ReceivedMessage, AS_i.SentMessage)|$1 \leq i \leq$|RCase|, RCase ∈ RLog}.

Output: $\sum TPM$ = (P,T; F, M_0).

1: P ← ∅, PL ← ∅, PM ← ∅, T ← ∅, TA ← ∅, TPA ← ∅, F ← ∅, and M0 ← ∅
2: For each ASi ∈ AssignmentSet(RCase) Do
  If ASi ∈ ActivitySet(RCase) then
    TA ← TA ∪ {ASi}
  Else if ASi ∈ ProcedureSet(RCase)
    TPA ← TPA ∪ {ASi}
  End if
 End do
3: For each ASi.ASj ∈ T Do
  If ASj ∈ ASi.PostSet then
    (3.1) PL ← PL ∪ {pij};
    (3.2) F ← F ∪ {(ASi,pij),(pij,ASj)};
  End if
 End do
4: For each ASi ∈ T Do
  If ASi.ReceivedMessage ≠ ∅ then
    For each mi ∈ ASi.ReceivedMessage Do
      (4.1) PM ← PM ∪ {pmi};
      (4.2) F ← F ∪ {(pmi,ASi)};
  End if
 End do
End do
5: For each ASi ∈ T Do
  If ASi.SentMessage ≠ ∅ then
    For each mi ∈ ASi.SentMessage Do
      (4.1) PM ← PM ∪ {pmi};
      (4.2) F ← F ∪ {(ASi,pmi)};
  End if
 End do
End do
6: For each ASi ∈ T Do
  If ASi.PreSet == ∅ then

(6.1) PL ← PL ∪ {ps};
(6.2) F ← F ∪ {(ps, ASi)};
Else if ASi.PreSet == ∅ then
(6.3) PL ← PL ∪ {pe};
(6.4) F ← F ∪ {(ASi, pe)};
End if
End do
7: For each p ∈ P Do
If p == ps then
M0(p) ← 1;
Else
M0(p) ← 0;
End if
End do
8: **return** $\sum TPM$= (P, T; F, M0).

2. Men-*discover Bottom Level* dengan Proses *Mining*

Untuk memperbaiki *abstract procedures* dari proses model *top level*, maka diperlukan proses model dari *bottom level*. Maka Algoritma 3 diusulkan untuk me-*mining* proses model *bottom level* dari *event log* yang dieksekusi.

Algoritma 3. Untuk memperoleh *bottom level* proses model $\sum BPM$.

Input: {(AS_i, AS_i.PreSet, AS_i.PostSet, AS_i.ReceivedMessage, AS_i.SentMessage)|$1 \leq i \leq$|RCase|, RCase ∈ RLog}.

Output: $\sum BPM$ = (P,T; F, M_0).
1: P ← ∅, PL ← ∅, PM ← ∅, T ← AssignmentSet(RCase), F ← ∅, and M0 ← ∅.
2: For each ASi,ASj ∈ T Do
If ASj ∈ ASi.PostSet then
(2.1) PL ← PL ∪ {pij};
(2.2) F ← F ∪ {(ASi, pij), (pij, ASj)};
End if
End do
3: For each ASi ∈ T Do
If ASi.ReceivedMessage ≠ ∅ then
For each mi ∈ ASi.ReceivedMessage Do
(3.1) PM ← PM ∪ {pmi};

(3.2) F ← F ∪ {(pmi,ASi)};
End if
End do
End do
4: For each ASi ∈ T Do
If ASi.SentMessage ≠ ∅ then
For each mi ∈ ASi.SentMessage Do
(4.1) PM ← PM ∪ {pmi};
(4.2) F ← F ∪ {(ASi,pmi)};
End if
End do
End do
5: P ← PL ∪ PM.
6 : **return** $\sum BPM$= (P, T; F, M0).

3. *Refinement of Petri Nets* untuk Proses Integrasi

Teknologi proses *mining* digunakan secara terpisah untuk menemukan proses model *top level* dan proses model *bottom level*. Dengan proses *refinement of petri nets*, suatu *abstract transition* dari proses model *top level* dapat disempurnakan oleh proses model *bottom level* yang benar dan sesuai. Berikutnya, Algoritma 4 digunakan untuk melakukan proses *refinement of petri nets*.

Algoritma 4. Untuk memperbaiki *top-level* proses model $\sum TPM$ menggunakan *bottom level* proses model $\sum BPM$.

Input: $\sum TPM$= (P, T; F, M0) dan θ = {$\sum BPM\ i$= (Pi, Ti; Fi, M0i)| 1≤ i ≤|Tp|}.

Output: $\sum' TPM$ = (P', T'; F', M0').
1: P' ← P, T' ← T, F' ← F, and M0' ← M0.
2: For each $\sum BPM\ i \in \theta$ Do
(2.1) In ($\sum BPM\ i$) ← ∅;
(2.2) Out ($\sum BPM\ i$) ← ∅;
For each pi ∈ P Do
If PreSet pi == ∅ then
In ($\sum BPM\ i$) ← In ($\sum BPM\ i$) ∪ pi;
Else if PostSet pi == ∅ then

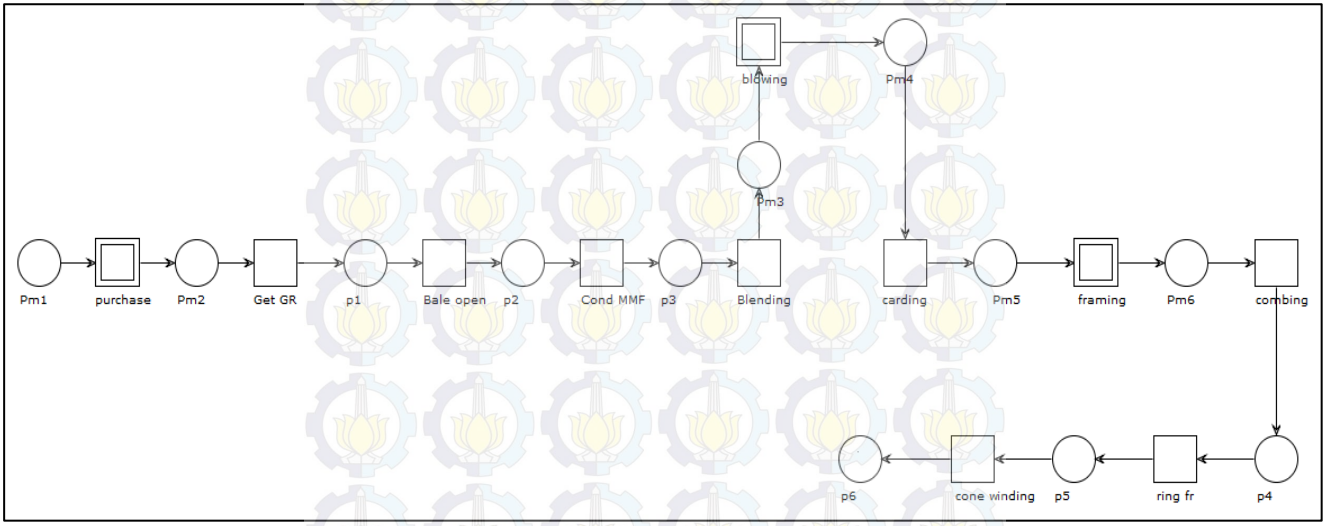
Out ($\sum BPM\ i$) ← In ($\sum BPM\ i$) ∪ pi;
End if
End do
3: For each t ∈ TPA Do
For each $\sum BPM\ i \in \theta$ Do
If (PreSet t == In ($\sum BPM\ i$)) ∧ (PostSet t == Out ($\sum BPM\ i$)) then
(3.1) P' ← P' ∪ Pi;
(3.2) T' ← (T' – t) ∪ Ti;
(3.3) F' ← (F' ∪ Fi) ∩ ((P' x T') ∪ (T' x P'));
End if
End do
End do
4: **return** $\sum' TPM$ = (P', T'; F', M0').

3.3 Contoh Penerapan Algoritma *Top-to-Down Process Mining Based on Refinement of Petri Nets from Multi-source Event Log*

Pada bagian ini akan dijelaskan contoh penerapan algoritma *top-to-down process mining based on refinement of petri nets from multi-source event log*. Data *event log* yang digunakan yaitu proses bisnis produksi benang. Sebuah proses bisnis produksi benang digunakan sebagai sebuah kasus untuk mengilustrasikan metode *top-to-down process mining*. Gambar 3.2 di bawah menunjukkan *top level* proses model dari proses bisnis produksi benang dan pada Tabel 3.1 menunjukkan keterangan pesan dari masing-masing aktivitas.

Tabel 3.1 Keterangan Pesan Setiap Aktivitas

Message	Meaning
Pm1	Accept Booking
Pm2	Goods Arrival notice
Pm3	Goods Preparation notice
Pm4	Deliver Goods
Pm5	Deliver Clumps of Cotton Fiber
Pm6	Processed Rolls Lap Notice
Pm7	Deliver Web Sliver
Pm8	Deliver Sliver Can
Pm9	Complete Lap Former for Spinning Process notice
Pm10	Deliver Bobbin Roving
Pm11	Bobbin Roving Verification notice



Gambar 3.2 *Top Level* Proses Model dari Proses Bisnis Produksi Benang

***Step* 1**. Tabel 3.2 menunjukkan bagian dari *event log* arsitektur *top level* yang melibatkan satu *case event log* yang dieksekusi. Sesuai data dari Tabel 3.2, *required messages* dan *sent messages* dari masing-masing aktivitas dapat diperoleh secara langsung. *Event log* Tabel 3.2 dijadikan sebagai *input* dan langkah selanjutnya adalah menerapkan Algoritma 1, *Pre-Set*, *Post-set*, *ReceivedMessage* dan *SentMessage* dari masing-masing aktivitas yang ditunjukkan oleh Tabel 3.3. Lalu dengan menerapkan Algoritma 2 yang menggunakan Tabel 3.3 sebagai input, *top level* proses model dari proses bisnis produksi benang ditunjukkan pada Gambar 3.3.

Tabel 3.2 Bagian dari *Event Log Top Level* Proses Model

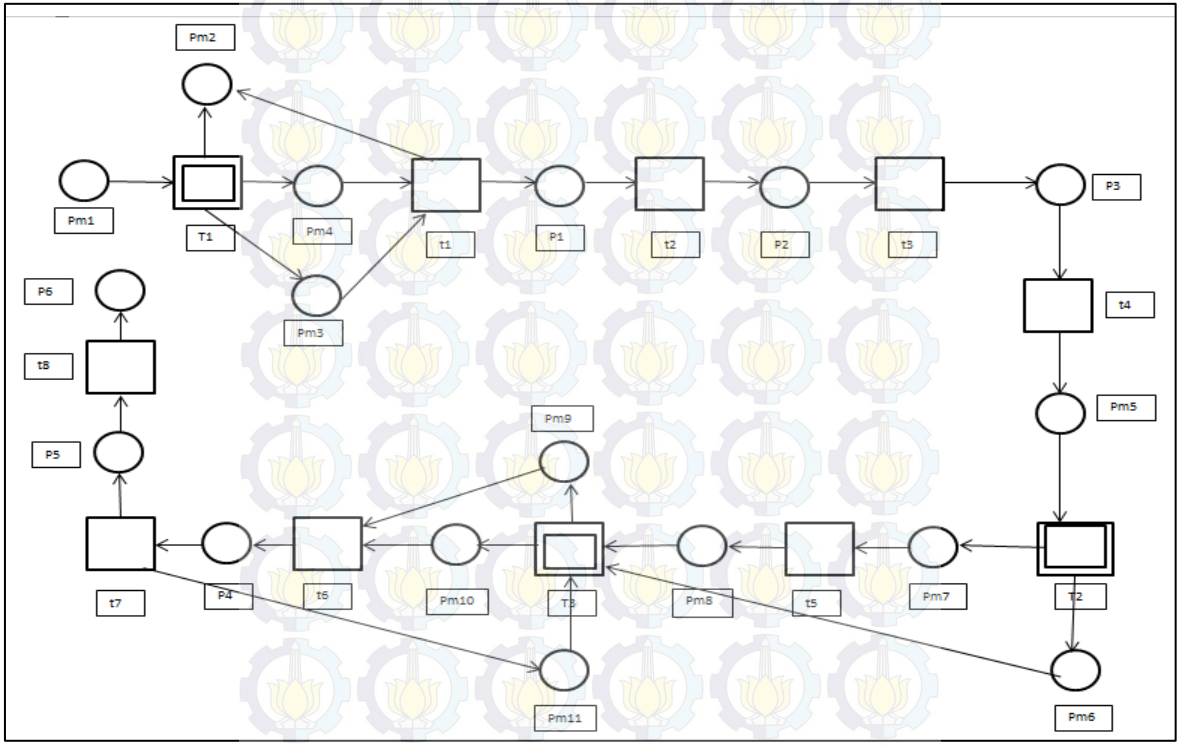
Case ID	Activity	Time Stamp	End Time	Organisator	Operator	RequiredMessage	SentMessage
PP1	T1	6/20/2014 8:32	6/20/2014 13:42	Purchase Department	Operator 2	{Pm1,Pm2}	{Pm3,Pm4}
PP1	t1	6/20/2014 13:42	6/20/2014 23:41	Spinning Department	Operator 1	{Pm3}	{Pm2}
PP1	t2	6/20/2014 23:41	6/21/2014 8:16	Spinning Department	Operator 1	{}	{}
PP1	t3	6/21/2014 8:16	6/21/2014 10:46	Spinning Department	Operator 1	{}	{}
PP1	t4	6/21/2014 10:46	6/21/2014 16:57	Spinning Department	Operator 1	{}	{Pm5}
PP1	T2	6/21/2014 16:57	6/21/2014 18:09	Blowing Department	Operator 3	{Pm5}	{Pm6,Pm7}
PP1	t5	6/21/2014 19:15	6/22/2014 10:51	Spinning Department	Operator 1	{Pm7}	{Pm8}
PP1	T3	6/22/2014 10:51	6/22/2014 23:42	Framing Department	Operator 4	{Pm8,Pm11}	{Pm9,Pm10}
PP1	t6	6/22/2014 23:42	6/23/2014 4:48	Spinning Department	Operator 1	{Pm10,Pm9}	{}
PP1	t7	6/23/2014 4:48	6/23/2014 16:44	Spinning Department	Operator 1	{}	{Pm11}
PP1	t8	6/23/2014 16:44	6/23/2014 23:26	Spinning Department	Operator 1	{}	{}

Tabel 3.3 ***Pre-Set*** **dan** ***Post-Set*** **dari Setiap Aktivitas pada** ***Top Level*** **Proses Model**

Assignment	Meaning	Pre-Set	Post-Set	RequiredMessage	SentMessage
T1	Purchase procedure	{}	{t1}	{Pm1,Pm2}	{Pm3,Pm4}
t1	Getting good receive	{T1}	{t2}	{Pm3}	{Pm2}
t2	Bale opening	{t1}	{t3}	{}	{}
t3	Conditioning of MMP Fiber	{t2}	{t4}	{}	{}
t4	Blending	{t3}	{T2}	{}	{Pm5}
T2	Blowing procedure	{t4}	{t5}	{Pm5}	{Pm6,Pm7}
t5	Carding	{T2}	{T3}	{Pm7}	{Pm8}
T3	Framing procedure	{t5}	{t6}	{Pm8,Pm11}	{Pm9,Pm10}
t6	Combing	{T3}	{t7}	{Pm10,Pm9}	{}
t7	Ring framing	{t6}	{t8}	{}	{Pm11}
t8	Cone winding	{t7}	{}	{}	{}

Hasil *mining* dari *top level* adalah sebuah proses model *top level* dengan *abstract transitions* yang direpresentasikan oleh transisi A_i (i = 1, 2,…,9) dan *abstract transitions* T_j (j = 1,2,3). Rincian proses dari tiga *abstract procedure* ini tidak dapat diperoleh pada tahap ini. Arti dari setiap pesan yang terdapat pada masing-masing aktivitas ditunjukkan pada Tabel 3.1.

***Step* 2**. Bagian *event log* dari *purchase department*, *blowing department* dan *framing department* ditunjukkan pada Tabel 3.4, Tabel 3.5, dan Tabel 3.6.



Gambar 3.3 Hasil *Mining Top Level* Proses Model untuk Proses Bisnis Produksi Benang

Pertama, Tabel 3.7, *Pre-Set*, *Post-Set*, *ReceivedMessage* dan *SentMessage* dari masing-masing aktivitas ditunjukkan pada Tabel 3.7 dengan menggunakan Algoritma 1. Kemudian, dengan mengeksekusi Algoritma 3, diperoleh *bottom level* proses model untuk *purchase department* (T1) seperti yang ditunjukkan oleh Gambar 3.4.

Kedua, Tabel 3.8, *Pre-Set*, *Post-Set*, *ReceivedMessage* dan *SentMessage* dari masing-masing aktivitas ditunjukkan pada Tabel 3.8 dengan menggunakan Algoritma 1. Kemudian, dengan mengeksekusi Algoritma 3, diperoleh *bottom level* proses model untuk *blowing department* (T2) seperti yang ditunjukkan oleh Gambar 3.5.

Ketiga, Tabel 3.9, *Pre-Set*, *Post-Set*, *ReceivedMessage* dan *SentMessage* dari masing-masing aktivitas ditunjukkan pada Tabel 3.9 dengan menggunakan Algoritma 1. Kemudian, dengan mengeksekusi Algoritma 3, diperoleh *bottom level* proses model untuk *framing department* (T3) seperti yang ditunjukkan oleh Gambar 3.6.

Tabel 3.4 Bagian dari *Event Log Purchase Department*

Case ID	Activity	Time Stamp	End Time	Operator	RequiredMessage	SentMessage
PP1	A1.1	6/20/2014 8:32	6/20/2014 8:45	Operator 2	{Pm1}	{}
PP1	A1.2	6/20/2014 8:45	6/20/2014 11:20	Operator 2	{}	{}
PP1	A1.3	6/20/2014 11:20	6/21/2014 9:09	Operator 2	{}	{Pm3}
PP1	A1.4	6/21/2014 9:09	6/21/2014 10:46	Operator 2	{}	{}
PP1	A1.5	6/21/2014 10:46	6/21/2014 11:20	Operator 2	{}	{}
PP1	A1.6	6/21/2014 11:20	6/20/2014 14:48	Operator 2	{}	{}
PP1	A1.7	6/20/2014 14:48	6/20/2014 16:02	Operator 2	{}	{}
PP1	A1.8	6/20/2014 16:02	6/20/2014 17:42	Operator 2	{Pm2}	{}
PP1	A1.9	6/20/2014 17:42	6/20/2014 19:42	Operator 2	{}	{Pm4}

Tabel 3.5 Bagian dari *Event Log Blowing Department*

Case ID	Activity	Time Stamp	End Time	Operator	RequiredMessage	SentMessage
PP1	A2.1	6/21/2014 10:46	6/21/2014 16:57	Operator 3	{Pm5}	{}
PP1	A2.2	6/21/2014 16:57	6/21/2014 18:09	Operator 3	{}	{}
PP1	A2.3	6/21/2014 18:09	6/21/2014 18:20	Operator 3	{}	{}
PP1	A2.4	6/21/2014 18:20	6/21/2014 19:15	Operator 3	{}	{}
PP1	A2.5	6/21/2014 19:15	6/22/2014 2:51	Operator 3	{}	{Pm6}
PP1	A2.6	6/22/2014 2:51	6/21/2014 19:15	Operator 3	{}	{Pm7}

Tabel 3.6 Bagian dari *Event Log Framing Department*

Case ID	Activity	Time Stamp	End Time	Operator	RequiredMessage	SentMessage
PP1	A3.1	6/22/2014 10:51	6/22/2014 15:08	Operator 4	{Pm8}	{}
PP1	A3.2	6/22/2014 15:08	6/22/2014 19:25	Operator 4	{}	{}
PP1	A3.3	6/22/2014 19:25	6/22/2014 23:42	Operator 4	{Pm6}	{Pm9}
PP1	A3.4	6/22/2014 23:42	6/23/2014 1:42	Operator 4	{}	{}
PP1	A3.5	6/23/2014 1:42	6/23/2014 2:16	Operator 4	{}	{}
PP1	A3.6	6/23/2014 2:16	6/23/2014 3:17	Operator 4	{Pm11}	{Pm10}

Tabel 3.7 *Pre-Set* dan *Post-Set* dari Setiap Aktivitas dari *Purchase Department*

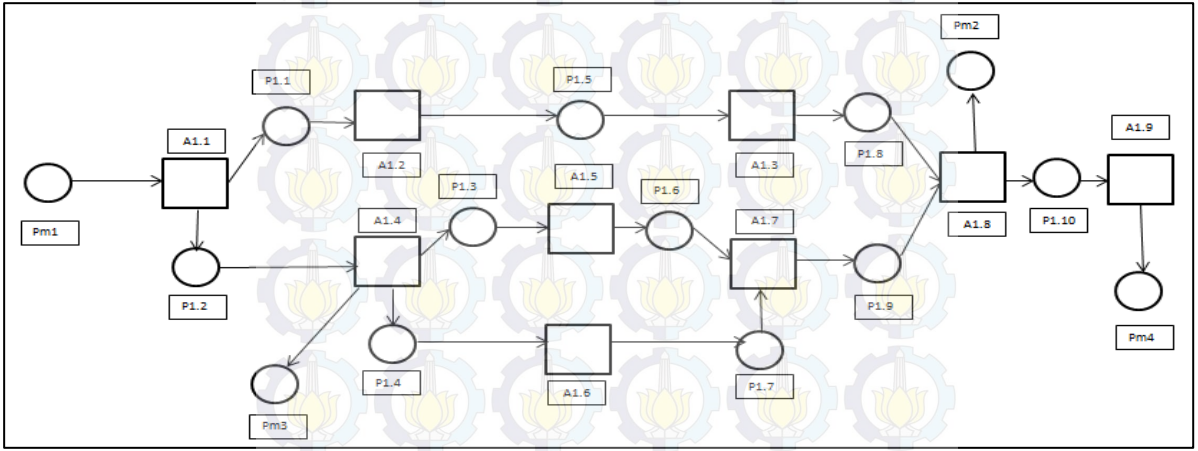
Assignment	Meaning	Pre-Set	Post-Set	RequiredMessage	SentMessage
A1.1	Sending PO Number	{}	{A1.2}	{Pm1}	{}
A1.2	Producing Good Orders Sup1	{A1.1}	{A1.3}	{}	{}
A1.3	Determining PPh and Giving Permission	{A1.2}	{A1.4}	{}	{Pm3}
A1.4	Paying PPh	{A1.3}	{A1.5}	{}	{}
A1.5	Producing Good Orders Sup2	{A1.4}	{A1.6}	{}	{}
A1.6	Packaging Good Orders	{A1.5}	{A1.7}	{}	{}
A1.7	Packaging Good Orders and Getting PPh Confirm	{A1.6}	{A1.8}	{}	{}
A1.8	Sending Good Orders	{A1.7}	{A1.9}	{Pm2}	{}
A1.9	Receiving Good Receive	{A1.8}	{}	{}	{Pm4}

Tabel 3.8 *Pre-Set* dan *Post-Set* dari Setiap Aktivitas dari *Blowing Department*

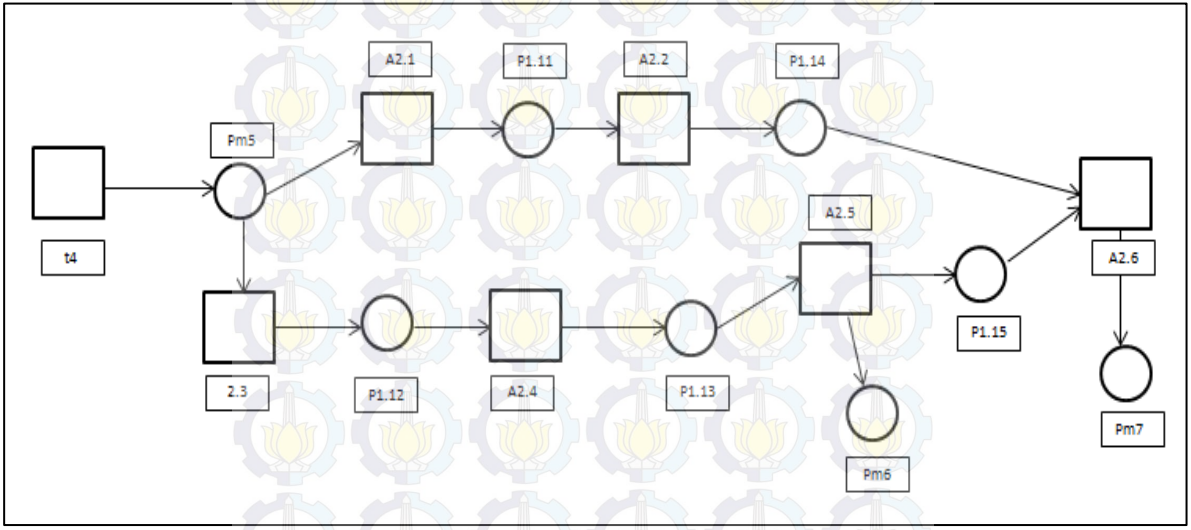
Assignment	Meaning	Pre-Set	Post-Set	RequiredMessage	SentMessage
A2.1	Opposing Spike	{}	{A2.2}	{Pm5}	{}
A2.2	Air Current Blowing	{A2.1}	{A2.3}	{}	{}
A2.3	Open Clumps of Paper	{A2.2}	{A2.4}	{}	{}
A2.4	Cleaning Fiber with Dirt	{A2.3}	{A2.5}	{}	{}
A2.5	Cleaning/ Separation Fibers with Dirt and Material to Make Rolls of Cloth	{A2.4}	{A2.6}	{}	{Pm6}
A2.6	Striking Cotton	{A2.5}	{}	{}	{Pm7}

Tabel 3.9 *Pre-Set* dan *Post-Set* dari Setiap Aktivitas dari *Framing Department*

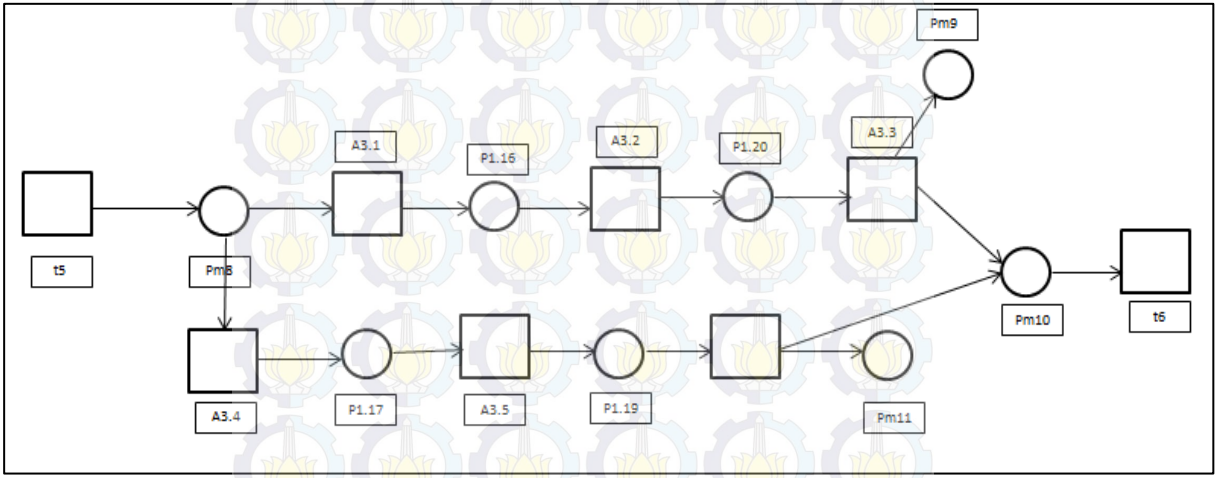
Assignment	Meaning	Pre-Set	Post-Set	RequiredMessage	SentMessage
A3.1	Drawing Breaker	{}	{A3.2}	{Pm8}	{}
A3.2	Drawing Finisher	{A3.1}	{A3.3}	{}	{}
A3.3	Lap Former	{A3.2}	{A3.4}	{Pm6}	{Pm9}
A3.4	Drafting	{A3.3}	{A3.5}	{}	{}
A3.5	Twisting	{A3.4}	{A3.6}	{}	{}
A3.6	Winding	{A3.5}	{}	{Pm11}	{Pm10}



Gambar 3.4 Hasil *Mining* Proses Model *Purchase Department*



Gambar 3.5 Hasil *Mining* Proses Model *Blowing Department*

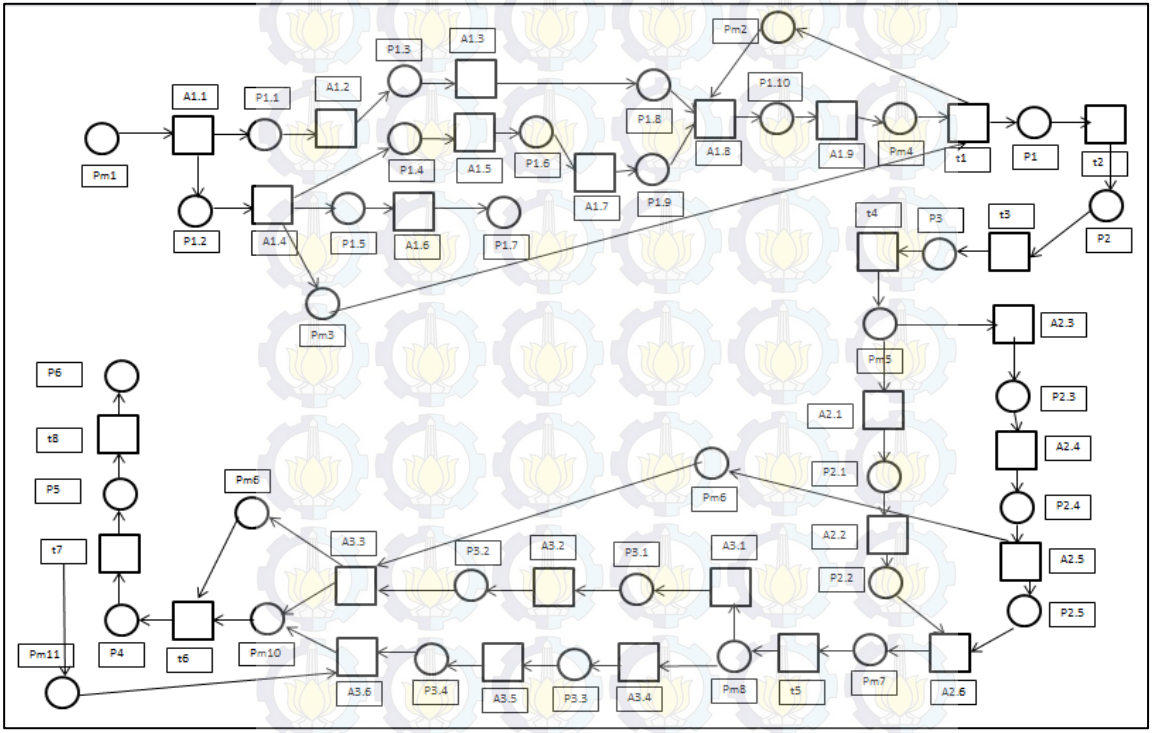


Gambar 3.6 Hasil *Mining* Proses Model *Framing Department*

***Step* 3**. Proses model *bottom level* yang ditunjukkan oleh Gambar 3.4, Gambar 3.5, dan Gambar 3.6 yang sesuai dengan tiga *abstract procedure* pada Gambar 3.3. Kemudian *abstract transitions* T1, T2 dan T3 dapat diperbaiki oleh proses model dari Gambar 3.4, Gambar 3.5, dan Gambar 3.6 dengan menggunakan Algoritma 4. Proses bisnis produksi benang yang telah di *refinement* ditunjukkan oleh Gambar 3.7 dan Tabel 3.10 menunjukkan nama setiap aktivitas proses bisnis produksi benang.

Tabel 3.10 Keterangan Nama Setiap Aktivitas Produksi Benang

Activity Code	Name of Activity
A	Sending PO Number
B	Producing Good Orders Sup1
C	Packaging Good Orders
D	Determining PPh and Giving Permission
E	Paying PPh
F	Producing Good Orders Sup2
G	Packaging Good Orders and Getting PPh Confirm
H	Sending Good Orders
I	Receiving Good Receive
J	Getting Good Receive
K	Bale Opening
L	Conditioning of MMP Fiber
M	Blending
N	Opposing Spike
O	Cleaning Fiber with Dirt
P	Air Current Blowing
Q	Open Clumps of Paper
R	Cleaning/ Separation Fibers with Dirt and Material to Make Rolls of Cloth
S	Striking cotton
T	Carding
U	Drawing Breaker
V	Drawing Finisher
W	Lap Former
X	Drafting
Y	Twisting
Z	Winding
AA	Combing
AB	Ring framing
AC	Cone winding



Gambar 3.7 Hasil Proses *Refinement* untuk Proses Bisnis Produksi Benang

3.4 Algoritma *Modified Time-Based Heuristics Miner*

Pada bagian ini dijelaskan tentang bagaimana algoritma *modified time-based heuristics miner* digunakan dalam Tugas Akhir ini dan bagaimana algoritma *original heuristics miner* dan *heuristics miner for time intervals* dimodifikasi dalam Tugas Akhir ini. Tahapan algoritma *modified time-based heuristics miner* yang telah dikembangkan memiliki kesamaan dengan algoritma *original heuristics miner* dan *heuristics miner for time intervals* yang telah dikembangkan sebelumnya hanya saja terdapat tambahan dan perubahan pada tahapan dan rumus yang ada pada tiga tahap utama.

3.4.1. *Mining Dependency Graph*

Sama seperti tahapan yang ada pada algoritma *heuristics miner* yang telah dikembangkan sebelumnya (Weijters, van der Aalst, & de Medeiros, 2006), pada algoritma *modified time-based heuristic miner* yang telah dimodifikasi memerlukan matriks frekuensi dari hubungan antar aktivitas dalam proses bisnis. Kemudian frekuensi dari matriks frekuensi akan dihitung menggunakan Persamaan 2.1. Modifikasi yang dilakukan pada bagian ini adalah pada penentuan *threshold* yang akan digunakan untuk memilih *dependency measure* pada dependency matriks yang akan digunakan dalam model. Penentuan dari nilai-nilai *threshold* ini dimaksudkan untuk mempermudah pengguna agar dapat men-*discover* model yang benar. Penentuan dari *threshold-threshold* tersebut adalah sebagai berikut:

a. *Relative-to-best threshold* (RBT)

Ada dua langkah untuk mengetahui nilai dari *relative-to-best threshold* ini, yaitu:

- Langkah pertama dalam menghitung *Relative-to-best threshold* (RBT) ini adalah menghitung rata-rata (Avg) dari *positive dependency measure* pada *matrix dependency* (PDM).
- Langkah kedua adalah menentukan nilai dari RBT menggunakan Persamaan 3.1.

$$RBT = Avg\ PDM - \left(\frac{SD\ PDM}{2}\right) \quad (3.1)$$

Keterangan:
RBT : *Relative-to-best threshold*
$Avg\ PDM$: rata-rata *positive dependency measure* pada *matrix dependency*
$SD\ PDM$: standar deviasi *positive dependency measure* pada *matrix dependency*

b. *Positive observations threshold* (POT)

Threshold ini mengontrol jumlah minimum berapa kali aktivitas memiliki ketergantungan hubungan dengan aktivitas lain. Persamaan 3.2 merupakan rumus untuk penentuan *threshold* ini.

$$POT = Min\ (f_{a=>b}) \quad (3.2)$$

c. *Dependency Threshold* (DT)

Threshold ini berguna untuk menentukan semua hubungan yang nilai *dependency measure* berada diatas nilai parameter ini. Jika nilai *dependency measure* berada diatas nilai parameter ini, maka hubungan antar aktivitas tersebut digunakan dalam penentuan proses model. Persamaan 3.3 merupakan rumus untuk penentuan *threshold* ini.

$$DT = Avg\ PDM - SD\ PDM \quad (3.3)$$

Hubungan antar aktivitas diambil jika memenuhi Persamaan 3.4.

$$DM_{a=>b} \geq DT \quad (3.4)$$

Keterangan:
DT : *dependency threshold*

3.4.2. *Mining Short Loop* (*Length One Loop* dan *Length Two Loop)*

Pada tahap ini tidak terdapat perubahan. Untuk menghitung *short loop heuristics miner* yang dimodifikasi tetap menggunakan Persamaan 2.2 dan Persamaan 2.3 (Weijters, van der Aalst, & de Medeiros, 2006).

3.4.3. *Mining Process Parallel*

Perhitungan *parallel measure* membutuhkan frekuensi aktivitas paralel di *event log*. Frekuensi diperoleh dengan menghitung hubungan *concurrent* masing-masing aktivitas paralel di *event log*. *Discovery* hubungan *concurrent* membutuhkan *double timestamp*. Namun, *event log* saat ini adalah *single time stamped* yang merupakan *event log* umum diperoleh dari sebuah organisasi. Oleh karena itu, kita perlu memperkirakan durasi setiap aktivitas untuk mengubah *single timestamp* menjadi *double timestamp*. Menentukan durasi setiap aktivitas di *event log* dilakukan dengan rata-rata masing-masing durasi aktivitas dalam setiap *case* yang terdapat di *event log*. Durasi aktivitas dalam setiap *case* diambil dari rata-rata durasi pelaksanaan setiap aktivitas di *event log*. Durasi eksekusi diperoleh dengan mengurangkan *output* dari suatu aktivitas dengan aktivitas lain (misalnya A memiliki aktivitas B sebagai *output* maka eksekusi durasi A adalah waktu eksekusi B mengurangkan dengan waktu pelaksanaan A).

$$if\ activity_{output} = tru\ then\ ed = et_{activity_{output}} - et_{activity} \quad (3.5)$$

Keterangan:

ed	:	durasi eksekusi aktivitas
$et_{activity_{output}}$	:	waktu eksekusi dari aktivitas output
$et_{activity}$	:	waktu eksekusi dari aktivitas

Setelah konversi *event log*, langkah selanjutnya adalah menghitung frekuensi kaitan antar aktivitas. Frekuensi dihitung

dengan *non-linear dependence*. Langkah-langkah dijelaskan sebagai berikut:

- Klasifikasikan relasi *sequence* (>) dan paralel (||) aktivitas dari setiap *trace* ($>_i$) di *event log* berdasarkan Definisi 3 dan Definisi 4.
- Hitung frekuensi dari relasi yang ditemukan pada *event log*.

Definisi 3. Diberikan *event log* (E) dan *trace* (σ) sehingga $\sigma \in L$. Relasi *causal* antara dua aktivitas $X(X_s, X_f)$ dan $Y(Y_s, Y_f)$, berdasarkan $X, Y \in L$ dapat dibedakan sebagai berikut:

$Before\ and\ meets, X > Y\ iff\ X_f \leq Y_s$

$Overlaps, X \square Y\ \ iff\ X_f > Y_s\ and\ X_f < Y_f$

$Contains, X@Y\ iff\ X_s < Y_s\ and\ Y_f > X_s\ and\ X_f > Y_f$

$Is\ finish\ by, X_f Y\ iff\ X_f = Y_f\ and\ X_s < Y_s\ and\ Y_s < X$

$Equals, X \diamondsuit Y\ iff\ X_s = Y_s\ and\ X_f = Y_f$

$Starts, X_p Y\ iff\ X_s = Y_s\ and\ X_f > Y_f$

Definisi 4. Diberikan *event log* (E) dan *trace* (σ) sehingga $\sigma \in L$. *Control flow* antara dua aktivitas $X(X_s, X_f)$ dan $Y(Y_s, Y_f)$, berdasarkan $X, Y \in L$ dapat dibedakan sebagai berikut:

$Sequence, X \rightarrow Y\ \ iff\ X > Y$

$Parallel, X \parallel Y\ \ iff\ X > Y\ and\ Y > X\ or$
$\{XY or\ X@Y\ or\ X_f Y\ or\ X \diamondsuit Y\ or\ X_p Y\}$

Pada algoritma *heuristics miner* yang telah dikembangkan sebelumnya hanya dapat men-*discover split* dan *join* AND dan XOR. Oleh karena itu, di dalam algoritma *heuristics miner* yang telah dimodifikasi ini akan membuat algoritma *heuristics miner* dapat men-*discover* semua jenis *split* dan *join* baik itu AND, XOR, maupun OR. Untuk dapat men-*discover* OR yang pertama dilakukan adalah sama dengan algoritma *heuristics miner* yaitu membentuk *causal matrix* dari hasil *dependency graph*, tetapi dalam memodifikasi ini dilakukan perubahan pada rumus yang digunakan untuk menghitung *parallel measure*. Pada persamaan

sebelumya pada Persamaan 2.4, Persamaan 2.5, dan Persamaan 2.6 hanya mementingkan *direct followed frequency* dari aktivitas pada cabang, padahal sebenarnya untuk *split* dan *join* AND sendiri memperbolehkan adanya *undirect followed frequency*.

Perubahan dilakukan dengan mengganti frekuensi yang hanya menghitung *direct followed frequency* pada aktivitas percabangan dengan juga menghitung semua *undirect followed frequency* karena sifat dari AND sendiri yang tidak hanya dapat dikakukan secara bersamaan atau *sequence* tapi aktivitas yang ada dalam percabangan dapat dilakukan secara tidak *sequence* yang penting semua aktivitas yang ada dalam percabangan dilakukan.

Untuk perhitungan OR dalam Tugas Akhir ini juga diperhitungkan frekuensi dari aktivitas yang tidak dijalankan secara bersamaan. Rumus untuk *parallel measure* adalah sebagai berikut:

$$X =>_w Y \wedge Z = \left(\frac{|Y \gg>_w Z| + |Z \gg>_w Y| + 2\,|Y\,||_w Z|}{|X >_w Y| + |X >_w Z| + |Y \gg> not_w Z| + |Z \gg> not_w Y| + 1} \right) \quad (3.6)$$

Keterangan:

$X =>_w Y \wedge Z$: *parallel measure* antara aktivitas pada percabangan dari X yaitu Y dan Z.

$|Y \gg>_w Z|$: *direct followed frequency* aktivitas Y dan Z

$|Z \gg>_w Y|$: *undirect and direct followed frequency* aktivitas Z dan Y

$2\,|Y\,||_w Z|$: jumlah aktivitas Y dan Z yang *overlap*

$|X >_w Y|$: frekuensi aktivitas X yang diikuti secara langsung oleh Y.

$|X >_w Z|$: frekuensi aktivitas X yang diikuti secara langsung oleh Z.

$|Y \gg> not_w Z|$: frekuensi eksekusi aktivitas Y yang tidak diikuti aktivitas Z.

$|Z \gg> not_w Y|$: frekuensi eksekusi aktivitas Z yang tidak diikuti aktivitas Y.

3.4.4. Pemodelan Relasi Paralel OR, XOR, dan AND

Untuk pemodelan XOR, OR, dan AND penentuannya menggunakan interval yang digunakan untuk menentukan letak dari rata-rata *parallel measure* dari aktivitas dengan *input split* yang sama ataupun *output join* yang sama. Yang pertama dilakukan adalah menentukan rata-rata dari *dependency measure* yang masuk dalam *dependency graph* dan rata-rata dari *parallel measure.*

$$Avg\ PDM = \frac{\sum_{i=1}^{n_e} e_i}{n_e} \quad (3.7)$$

$$Avg\ PM = \frac{\sum_{i=1}^{n_{PM}} PM_i}{n_{PM}} \quad (3.8)$$

Interval untuk XOR, OR, dan AND adalah sebagai berikut:

- XOR
  $If\ Avg\ PM \leq Minimum\ PDM\ then\ XOR$ (3.9)
- OR
  $If\ Minimum\ PDM \leq Avg\ PM \leq AVG\ PDM\ then\ OR$ (3.10)
- AND
  $If\ Avg\ PDM \leq Avg\ PM\ then\ AND$ (3.11)

Keterangan:

$Avg\ PDM$: rata-rata *positive dependency measure* yang lebih dari 0 dari *positive dependency measure*

e_i : *dependency measure* pada *dependency matrix*

n_e : jumlah *dependency measure*

PM : *Parallel measure* dari Persamaan 3.6

$Avg\ PM$: rata-rata dari *parallel measure* yang memiliki induk aktivitas yang sama.

n_{PM} : jumlah *parallel measure* yang memiliki *input* atau *output* yang sama.

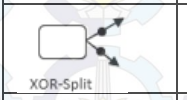
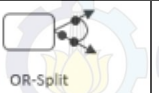
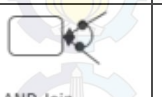
PDM : *positive dependency measure*

Minimum PDM : nilai minimum dari positive *dependency measure* pada *matrix dependency measure.*

3.4.5. Penggambaran OR, XOR, dan AND pada *Causal Net*

Karena bentuk dari pemodelan untuk OR, XOR, dan AND yang telah dilimiki oleh *Causal Net* kurang informatif, susah dibaca dan dimengerti oleh pengguna awam, maka dalam Tugas Akhir ini modelnya diubah menjadi sebagai berikut:

Tabel 3.11 *C-Net* dan *Proposed Parallel Model*

C-Net Model	Proposed Model
	
	
	
	
	
	

3.5 Contoh *Discovery* Menggunakan *Modified Time-Based Heuristics Miner*

Diberikan *event log* sebagai berikut:

$$E = \begin{bmatrix} (ABCDEFGHIJK), (ABCDEFGIJK), (ABDCEFHIJK), (ABDCEFHGIJK), \\ (ABCEFGHIJK), (ABCDEFGIJK), (ABCDEFGHIJK), (ABCDEFGIJ), \\ (ABDCEFHIJJK), (ABDCEFHGIJKJK), (ABCDEFGIJJK), (ABCDEFGIJKJK), \\ (ABCDEFGHIJJK), (ABDCEFHIJKJK), (ABDCEFHGIJJK), \\ (ABDCEFHGIJKJK), (ABCDEFGHIJKJK), (ABCDEFHIJKJK) \end{bmatrix}$$

Tabel 3.12 ***Event Log***

Case ID	Activity	Start Stamp	End Stamp				
PP1	A	6/20/2014 8:32	6/20/2014 13:42	PP4	A	6/28/2014 19:29	6/28/2014 23:28
	B	6/20/2014 13:42	6/20/2014 23:41		B	6/28/2014 23:28	6/29/2014 9:12
	C	6/20/2014 23:41	6/21/2014 9:30		D	6/29/2014 9:12	6/29/2014 18:41
	D	6/21/2014 8:16	6/21/2014 10:46		C	6/29/2014 15:27	6/29/2014 20:50
	E	6/21/2014 10:46	6/21/2014 16:57		E	6/29/2014 20:50	6/30/2014 2:55
	F	6/21/2014 16:57	6/21/2014 18:09		F	6/30/2014 2:55	6/30/2014 9:14
	G	6/21/2014 18:09	6/22/2014 4:14		H	6/30/2014 9:14	6/30/2014 14:37
	H	6/21/2014 19:15	6/22/2014 10:51		G	6/30/2014 11:25	6/30/2014 17:33
	I	6/22/2014 10:51	6/22/2014 16:28		I	6/30/2014 17:33	7/1/2014 6:48
	J	6/22/2014 16:28	6/22/2014 23:42		J	7/1/2014 6:48	7/1/2014 9:41
	K	6/22/2014 23:42	6/23/2014 4:48		K	7/1/2014 9:41	7/1/2014 11:15
PP2	A	6/23/2014 4:48	6/23/2014 16:44	PP5	A	7/1/2014 11:15	7/1/2014 15:01
	B	6/23/2014 16:44	6/23/2014 23:26		B	7/1/2014 15:01	7/1/2014 21:44
	C	6/23/2014 23:26	6/24/2014 5:40		C	7/1/2014 21:44	7/2/2014 4:30
	D	6/24/2014 4:19	6/24/2014 7:48		E	7/2/2014 4:30	7/2/2014 14:50
	E	6/24/2014 7:48	6/25/2014 1:08		F	7/2/2014 14:50	7/3/2014 3:06
	F	6/25/2014 1:08	6/25/2014 3:02		G	7/3/2014 3:06	7/3/2014 22:01
	G	6/25/2014 3:02	6/25/2014 5:11		H	7/3/2014 14:41	7/4/2014 1:40
	I	6/25/2014 5:11	6/25/2014 8:25		I	7/4/2014 1:40	7/4/2014 9:02
	J	6/25/2014 8:25	6/25/2014 12:45		J	7/4/2014 9:02	7/4/2014 17:12
	K	6/25/2014 12:45	6/26/2014 0:12		K	7/4/2014 17:12	7/4/2014 20:56
PP3	A	6/26/2014 0:12	6/26/2014 4:48	PP6	A	7/4/2014 20:56	7/5/2014 10:21
	B	6/26/2014 4:48	6/26/2014 15:16		B	7/5/2014 10:21	7/5/2014 13:22
	D	6/26/2014 15:16	6/27/2014 1:52		C	7/5/2014 13:22	7/5/2014 17:56
	C	6/26/2014 22:49	6/27/2014 5:19		D	7/5/2014 16:26	7/5/2014 19:35
	E	6/27/2014 5:19	6/27/2014 11:40		E	7/5/2014 19:35	7/6/2014 0:37
	F	6/27/2014 11:40	6/27/2014 20:00		F	7/6/2014 0:37	7/6/2014 10:59
	H	6/27/2014 20:00	6/28/2014 7:10		G	7/6/2014 10:59	7/6/2014 21:18
	I	6/28/2014 7:10	6/28/2014 9:10		I	7/6/2014 21:18	7/7/2014 2:43
	J	6/28/2014 9:10	6/28/2014 16:40		J	7/7/2014 2:43	7/7/2014 17:13
	K	6/28/2014 16:40	6/28/2014 19:29		K	7/7/2014 17:13	7/8/2014 0:31

3.5.1. *Mining Dependency Graph*

Sesuai dengan tahapan dari algoritma *heuristics miner* yang telah diterangkan pada Bagian 3.4 maka yang pertama dilakukan adalah *mining dependency graph*. Dalam *mining dependency graph* diperlukan matriks frekuensi. Matriks frekuensi dari *event log* E dapat dilihat pada Tabel 3.13.

Tabel 3.13 Matriks Frekuensi *Event Log* E

Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	18	0	0	0	0	0	0	0	0	0
B	0	0	10	8	0	0	0	0	0	0	0
C	0	0	0	9	9	0	0	0	0	0	0
D	0	0	8	0	9	0	0	0	0	0	0
E	0	0	0	0	0	18	0	0	0	0	0
F	0	0	0	0	0	0	10	8	0	0	0
G	0	0	0	0	0	0	0	5	9	0	0
H	0	0	0	0	0	0	4	0	9	0	0
I	0	0	0	0	0	0	0	0	0	18	0
J	0	0	0	0	0	0	0	0	0	0	23
K	0	0	0	0	0	0	0	0	0	6	0

Lalu membentuk matriks *dependency measure* dengan menggunakan perhitungan pada Persamaan 2.1 menghasilkan Tabel 3.14.

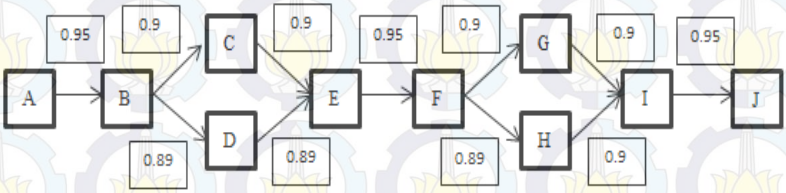
Tabel 3.14 Matriks *Dependency Measure*

Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	0.947	0	0	0	0	0	0	0	0	0
B	0	0	0.9	0.889	0	0	0	0	0	0	0
C	0	0	0	0	0.9	0	0	0	0	0	0
D	0	0	0	0	0.9	0	0	0	0	0	0
E	0	0	0	0	0	0.947	0	0	0	0	0
F	0	0	0	0	0	0	0.9	0.889	0	0	0
G	0	0	0	0	0	0	0	0	0.9	0	0
H	0	0	0	0	0	0	0	0	0.9	0	0
I	0	0	0	0	0	0	0	0	0	0.947	0
J	0	0	0	0	0	0	0	0	0	0	0.567
K	0	0	0	0	0	0	0	0	0	-0.567	0

Dengan menggunakan Persamaan 3.1, Persamaan 3.2, Persamaan 3.3, dan Persamaan 3.4 maka didapatkan nilai *threshold* sebagai berikut:

- *Average of positive dependency measure (PDM) = 0.884*
- *Standard deviation of PDM = 0.102*
- *Relative-to-best threshold = 0.833*
- *Positive observations threshold = 4*
- *Dependency threshold = 0.782*

Dengan menggunakan nilai dari *threshold* tersebut maka didapatkan *dependency graph* sebagai berikut:



Gambar 3.8 *Dependency Graph* dengan *Dependency Measure*

3.5.2. *Mining Short Loop (Length One Loop* dan *Length two Loop)*

Setelah mendapatkan *dependency graph*, selanjutnya dilakukan *mining* terhadap *short loop*. Dari matriks frekuensi pada Tabel 3.13 dan menggunakan Persamaan 2.2 didapatkan matriks *Length One Loop* sebagai berikut:

Tabel 3.15 Matriks Frekuensi *Length One Loop*

Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	4	0
K	0	0	0	0	0	0	0	0	0	0	0

Dari matriks *Length One Loop* pada Tabel 3.15 kemudian dengan menggunakan Persamaan 2.2 diketahui matriks *dependency Length One Loop* seperti Tabel 3.16:

Tabel 3.16 Matriks *Dependency Length One Loop*

Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0.8	0
K	0	0	0	0	0	0	0	0	0	0	0

Sedangkan untuk *Length Two Loop*, yang pertama harus dilakukan adalah menghitung frekuensi *Length Two Loop* sehingga mendapatkan matriks frekuensi *Length Two Loop* seperti Tabel 3.17:

Tabel 3.17 Matriks Frekuensi *Length Two Loop*

Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	6
K	0	0	0	0	0	0	0	0	0	6	0

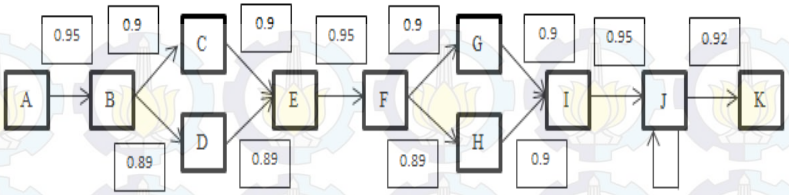
Kemudian menggunakan Persamaan 2.3 menghitung *Length Two Loop* dan membentuk matriks *Length Two Loop*. Hasil matriks *dependency Length Two Loop* seperti pada Tabel 3.18.

Tabel 3.18 Matriks *Dependency Length Two Loop*

Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0.923
K	0	0	0	0	0	0	0	0	0	0.923	0

Dari Gambar 3.18 diketahui bahwa proses model memiliki *Length Two Loop* JKJ dan KJK. Dari Tabel 3.15 and Tabel 3.17, diketahui bahwa *Length One Loop* 'JJ' dan *Length Two Loop* 'JKJ' termasuk di dalam proses model. Berdasarkan definisi dari *completeness*, *Length Two Loop dependency measure* antara dua aktivitas J dan K tidak dihitung karena mengandung salah satu dari tiga kondisi di bawah ini:
(1) J adalah *Length One Loop* dan $|J >>_W K|$ lebih tinggi dari *positive observation threshold* (POT) atau,
(2) K *adalah Length One Loop* dan $|K >>_W J|$ lebih tinggi dari *positive observation threshold* (POT) atau,
(3) semua kondisi di atas.

Jadi, berdasarkan kondisi di atas, untuk *event log* di Tabel 3.12, *Length Two Loop dependency* tidak dimasukkan ke dalam proses model. Tetapi, jika kita lihat pada *matrix dependency Length Two Loop*, hasil nilai dari JK dan KJ adalah 0.923. Nilai tersebut lebih tinggi daripada *Dependency Threshold* (0.782) sehingga relasi dari aktivitas J dan aktivitas K adalah *sequence*. Proses model yang dibentuk dari algoritma *modified time-based heuristics miner* adalah seperti Gambar 3.9.



Gambar 3.9 *Dependency Graph* dengan *Dependency Measure* dan *Length One Loop*

3.5.3. *Mining Parallel Activity*

Langkah terakhir dalam algoritma *modified time-based heuristics miner* adalah *mining parallel activity*. Yang pertama dilakukan dalam *mining parallel activity* adalah menentukan *causal matrix*. *Causal matrix* untuk hasil *dependency graph* pada *event log* pada Tabel 3.12 seperti pada Tabel 3.19.

Tabel 3.19 *Causal Matrix* dari Gambar 3.8

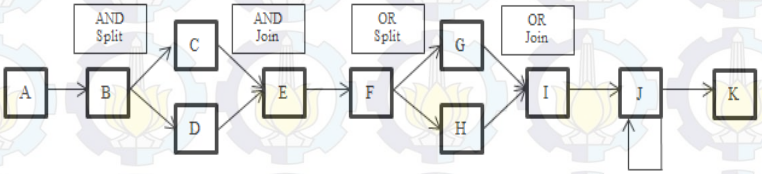
INPUT	ACTIVITY	OUTPUT
{}	A	B
A	B	C,D
B	C	E
B	D	E
C,D	E	F
E	F	G,H
F	G	I
F	H	I
G.H	I	J
I	J	K
J	K	{}

Dari *causal matrix* pada Tabel 3.19 didapatkan dua *split* dan *join* yaitu *split* dengan *input* B dan *input F*, *join* dengan *output C* dan *output I* dengan aktivitas pada percabangan adalah C-D dan G-H. Dengan Persamaan 3.6 maka didapatkan *parallel measure:*

$B \rightarrow w^{C \wedge D} = 1.378$ $E \rightarrow w^{C \wedge D} = 1.378$

$F \rightarrow w^{G \wedge H} = 0.729$ $I \rightarrow w^{G \wedge H} = 0.729$

Lalu untuk nilai Avg PM *input* B dan *output E* adalah 1.378 dan *Avg PM input F* dan *output* I adalah 0.729. Selain *Avg PM*, yang dibutuhkan selanjutnya adalah *Avg PDM* dengan Persamaan 3.7 adalah sebesar 0.884 dan *Minimum PDM* sebesar 0.567. Dengan menggunakan interval dari Persamaan 3.9, Persamaan 3.10, dan Persamaan 3.11, maka *split* dan *join* yang digunakan model yang di-*discovery* keduanya adalah AND dan OR. Sehingga modelnya menjadi sebagai berikut:



Gambar 3.10 Proses Model Akhir dengan algoritma *Modified Time-Based Heuristics Miner*

3.6 *Mining* dengan algoritma *original Heuristics Miner*

Berdasarkan paper *Process Mining with the Heuristics Miner Algorithm* (A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros), *event log* di Tabel 3.12 di-*discover* dengan menggunakan *original Heuristics Miner* dengan matriks frekuensi *direct successor* seperti Tabel 3.13. Pada algoritma ini, digunakan formula dasar dari *mining dependency measure* algoritma *Heuristics Miner* (A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros) (Rakesh Agrawal, Dimitrios Gunopulos, Frank Leymann). Lalu terbentuklah matriks *dependency measure* seperti pada Tabel 3.20.

Dengan menggunakan Persamaan 3.1, Persamaan 3.2, Persamaan 3.3, dan Persamaan 3.4 maka didapatkan nilai *threshold* sebagai berikut:

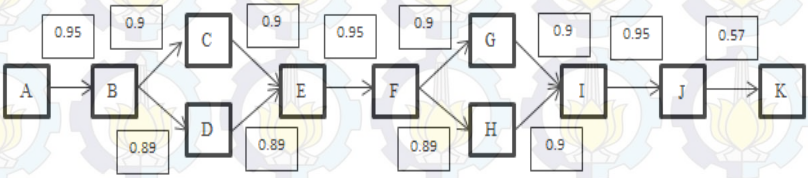
- *Average of positive dependency measure (PDM) = 0.772*
- *Standard deviation of PDM = 0.299*
- *Relative-to-best threshold = 0.622*
- *Positive observations threshold = 4*

- *Dependency threshold = 0.472*

Tabel 3.20 Matriks *Dependency Measure*

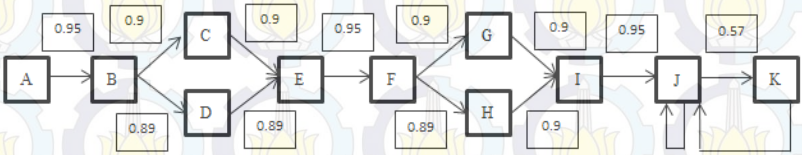
Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	0.947	0	0	0	0	0	0	0	0	0
B	0	0	0.9	0.889	0	0	0	0	0	0	0
C	0	0	0	0.1	0.9	0	0	0	0	0	0
D	0	0	-0.1	0	0.9	0	0	0	0	0	0
E	0	0	0	0	0	0.947	0	0	0	0	0
F	0	0	0	0	0	0	0.9	0.889	0	0	0
G	0	0	0	0	0	0	0	0.1	0.9	0	0
H	0	0	0	0	0	0	-0.1	0	0.9	0	0
I	0	0	0	0	0	0	0	0	0	0.947	0
J	0	0	0	0	0	0	0	0	0	0	0.567
K	0	0	0	0	0	0	0	0	0	-0.567	0

Dengan menggunakan nilai dari *threshold* tersebut maka didapatkan *dependency graph* sebagi berikut:



Gambar 3.11 *Dependency Graph* dengan *Dependency Measure*

Untuk *mining short loop* (*length one loop* dan *length two loop*) hasilnya sama dengan Tabel 3.15 hingga Tabel 3.18. Perbedaannya adalah tidak ada kondisi pengambilan *Length One Loop* dan *Length Two Loop* sehingga proses modelnya menjadi:



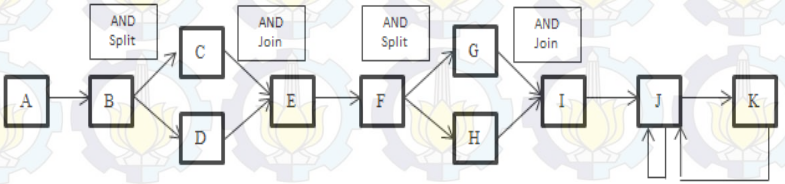
Gambar 3.12 *Dependency Graph* dengan *Dependency Measure*, *Length One Loop* dan *Length Two Loop*

Dengan menggunakan *causal matrix* yang sama seperti Tabel 3.19, tetapi perhitungan *parallel measure* yang berbeda. *Original Heuristics Miner* menggunakan Persamaan 2.5 untuk menghitung *parallel measure*.

$B \rightarrow w^{C \wedge D} = 0.894$ $\quad E \rightarrow w^{C \wedge D} = 0.894$

$F \rightarrow w^{G \wedge H} = 0.474$ $\quad I \rightarrow w^{G \wedge H} = 0.474$

Dengan menggunakan *rule* dari *original Heuristics Miner*, jika *parallel measure* lebih dari 0.1 maka *split* dan *join* yang digunakan adalah AND. Oleh karena itu, proses model yang di-*discover* dengan menggunakan *original Heuristics Miner* menjadi:



Gambar 3.13 Proses Model Akhir dengan Algoritma *Original Heuristics Miner*

3.7 *Mining* dengan algoritma *Process Model Discovery Based on Activity Lifespan*

Algoritma yang diperkenalkan oleh Rizka et. al. (Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Minsoo Kim, 2014) digunakan untuk menemukan proses model dari *event log* di Tabel 3.12. Relasi-relasi yang ditemukan (*discovered relations)* dari *trace* ke-1 hingga *trace* ke-18 akan dijabarkan sebagai berikut:

- *Discovered relations* di *trace* ke-1:
  Step 1. *Input* (T1) dan *output* (T0).
  I_1 = A, O_1 = K.
  Step 2. Relasi *Sequence*
  $>_1$ = {{} $\rightarrow$ A, A $\rightarrow$ B, B $\rightarrow$C, B $\rightarrow$ D, C $\rightarrow$ E, D $\rightarrow$ E, E $\rightarrow$ F, F $\rightarrow$ G, F $\rightarrow$ H, G $\rightarrow$ I, H $\rightarrow$ I, I $\rightarrow$ J, J $\rightarrow$ K, K $\rightarrow$ {}}.
  Step 3. Relasi *Parallel*

$\|_1 = \{C\|D, G\|H\}$.

- *Discovered relations* di *trace* ke-2:

  Step 1. *Input* (T1) dan *output* (T0).

  $I_1 = A, O_1 = K$.

  Step 2. Relasi *Sequence*

  $>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, G \rightarrow I, I \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}$.

  Step 3. Relasi *Parallel*

  $\|_1 = \{C\|D\}$.

- *Discovered relations* di *trace* ke-3:

  Step 1. *Input* (T1) dan *output* (T0).

  $I_1 = A, O_1 = K$.

  Step 2. Relasi *Sequence*

  $>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow H, H \rightarrow I, I \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}$.

  Step 3. Relasi *Parallel*

  $\|_1 = \{C\|D\}$.

- *Discovered relations* di *trace* ke-4:

  Step 1. *Input* (T1) dan *output* (T0).

  $I_1 = A, O_1 = K$.

  Step 2. Relasi *Sequence*

  $>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}$.

  Step 3. Relasi *Parallel*

  $\|_1 = \{C\|D, G\|H\}$.

- *Discovered relations* di *trace* ke-5:

  Step 1. *Input* (T1) dan *output* (T0).

  $I_1 = A, O_1 = K$.

  Step 2. Relasi *Sequence*

  $>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, C \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}$.

  Step 3. Relasi *Parallel*

  $\|_1 = \{G\|H\}$.

- *Discovered relations* di *trace* ke-6:

  Step 1. *Input* (T1) dan *output* (T0).

  $I_1 = A, O_1 = K$.

  Step 2. Relasi *Sequence*

$>_1$ = {{} → A, A → B, B →C, B → D, C → E, D → E, E → F, F → G, G → I, I → J, J → K, K → {}}.

Step 3. Relasi *Parallel*

$\|_1$ = {C‖D}.

- *Discovered relations* di *trace* ke-7:

Step 1. *Input* (T1) dan *output* (T0).

I_1 = A, O_1 = K.

Step 2. Relasi *Sequence*

$>_1$ = {{} → A, A → B, B →C, B → D, C → E, D → E, E → F, F → G, F → H, G → I, H → I, I → J, J → K, K → {}}.

Step 3. Relasi *Parallel*

$\|_1$ = {C‖D, G‖H}.

- *Discovered relations* di *trace* ke-8:

Step 1. *Input* (T1) dan *output* (T0).

I_1 = A, O_1 = K.

Step 2. Relasi *Sequence*

$>_1$ = {{} → A, A → B, B →C, B → D, C → E, D → E, E → F, F → G, G → I, I → J, J → K, K → {}}.

Step 3. Relasi *Parallel*

$\|_1$ = {C‖D}.

- *Discovered relations* di *trace* ke-9:

Step 1. *Input* (T1) dan *output* (T0).

I_1 = A, O_1 = K.

Step 2. Relasi *Sequence*

$>_1$ = {{} → A, A → B, B →C, B → D, C → E, D → E, E → F, F → H, H → I, I → J, J → J, J → K, K → {}}.

Step 3. Relasi *Parallel*

$\|_1$ = {C‖D}.

- *Discovered relations* di *trace* ke-10:

Step 1. *Input* (T1) dan *output* (T0).

I_1 = A, O_1 = K.

Step 2. Relasi *Sequence*

$>_1$ = {{} → A, A → B, B →C, B → D, C → E, D → E, E → F, F → G, F → H, G → I, H → I, I → J, J → K, K → J, J → K, K → {}}.

Step 3. Relasi *Parallel*

$\|_1$ = {C‖D, G‖H}.

- *Discovered relations* di *trace* ke-11:

Step 1. *Input* (T1) dan *output* (T0).

$I_1 = A, O_1 = K.$

Step 2. Relasi *Sequence*

$>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, C \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}.$

Step 3. Relasi *Parallel*

$\|_1 = \{G \| H\}.$

- *Discovered relations* di *trace* ke-12:

Step 1. *Input* (T1) dan *output* (T0).

$I_1 = A, O_1 = K.$

Step 2. Relasi *Sequence*

$>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, G \rightarrow I, I \rightarrow J, J \rightarrow K, K \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}.$

Step 3. Relasi *Parallel*

$\|_1 = \{C \| D\}.$

- *Discovered relations* di *trace* ke-13:

Step 1. *Input* (T1) dan *output* (T0).

$I_1 = A, O_1 = K.$

Step 2. Relasi *Sequence*

$>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}.$

Step 3. Relasi *Parallel*

$\|_1 = \{C \| D, G \| H\}.$

- *Discovered relations* di *trace* ke-14:

Step 1. *Input* (T1) dan *output* (T0).

$I_1 = A, O_1 = K.$

Step 2. Relasi *Sequence*

$>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow H, H \rightarrow I, I \rightarrow J, J \rightarrow K, K \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}.$

Step 3. Relasi *Parallel*

$\|_1 = \{C \| D\}.$

- *Discovered relations* di *trace* ke-15:

Step 1. *Input* (T1) dan *output* (T0).

$I_1 = A, O_1 = K.$

Step 2. Relasi *Sequence*

$>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}.$

Step 3. Relasi *Parallel*

$\|_1 = \{C\|D, G\|H\}$.

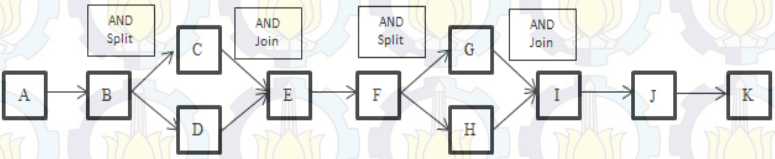
- *Discovered relations* di *trace* ke-16:
  Step 1. *Input* (T1) dan *output* (T0).
  $I_1 = A, O_1 = K$.
  Step 2. Relasi *Sequence*
  $>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow K, K \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}$.
  Step 3. Relasi *Parallel*
  $\|_1 = \{C\|D, G\|H\}$.
- *Discovered relations* di *trace* ke-17:
  Step 1. *Input* (T1) dan *output* (T0).
  $I_1 = A, O_1 = J$.
  Step 2. Relasi *Sequence*
  $>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}$.
  Step 3. Relasi *Parallel*
  $\|_1 = \{C\|D, G\|H\}$.
- *Discovered relations* di *trace* ke-18:
  Step 1. *Input* (T1) dan *output* (T0).
  $I_1 = A, O_1 = K$.
  Step 2. Relasi *Sequence*
  $>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow K, K \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}$.
  Step 3. Relasi *Parallel*
  $\|_1 = \{C\|D, G\|H\}$.

Discovered relations di setiap *trace* akan digabungkan dan diklasifikasikan seperti berikut:

- *Step* 1. *Input* (T1) dan *output* (T0) dari setiap *trace*.
  $I = A, O = K$.
- *Step* 2. Semua relasi *sequence*
  $> = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow E, E \rightarrow F, F \rightarrow G, F \rightarrow H, G \rightarrow I, H \rightarrow I, I \rightarrow J, J \rightarrow J, J \rightarrow K, K \rightarrow J, J \rightarrow K, K \rightarrow \{\}\}$.
- *Step* 3. Semua relasi *parallel*

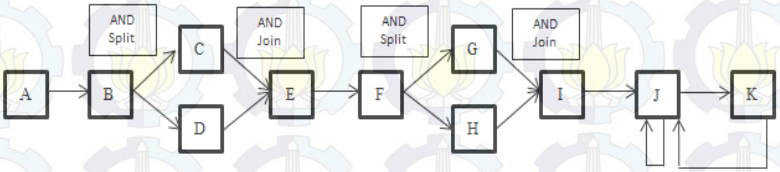
|| = {C||D, G||H}

- *Step* 4. Klasifikasikan relasi *parallel* menjadi *parallel* AND dan *conditional XOR*.
  ⊕ (*Parallel* AND) = {C⊕D, G⊕H}
  ⊗ (*Conditional XOR*) = { }
- *Step* 5. Relasi *parallel AND*
  ⊕ = {B ⊕ (C, D)}
  ⊕ = {F ⊕ (G, H)}
- *Step* 6. Graf yang tersusun dari relasi paralel XOR atau AND
  G = {⊕,⊕}



Gambar 3.14 Proses Model dengan relasi paralel AND

- *Step* 7. Tambahkan relasi *sequence* dan *input/output* pada graf final.
  $G \leftarrow G \cup (>, I, O)$



Gambar 3.15 Proses Model Akhir dengan algoritma *Process Model Discovery Based on Activity Lifespan*

3.8 *Mining* dengan algoritma *Heuristics Miner for Time Intervals*

Algoritma ini diperkenalkan oleh Andrea Burattin untuk menemukan proses model dari *event log* menggunakan aktivitas sebagai interval waktu (Andrea Burattin, Alessandro Sperduti,

2010). *Event log* di Tabel 3.12 di-*mining* menggunakan *Heuristics Miner for Time Intervals* dengan matriks frekuensi *direct successor* sebagai berikut:

Tabel 3.21 Matriks Frekuensi *Direct Successor*

Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	18	0	0	0	0	0	0	0	0	0
B	0	0	10	8	0	0	0	0	0	0	0
C	0	0	0	9	9	0	0	0	0	0	0
D	0	0	8	0	9	0	0	0	0	0	0
E	0	0	0	0	0	18	0	0	0	0	0
F	0	0	0	0	0	0	10	8	0	0	0
G	0	0	0	0	0	0	0	5	9	0	0
H	0	0	0	0	0	0	4	0	9	0	0
I	0	0	0	0	0	0	0	0	0	18	0
J	0	0	0	0	0	0	0	0	0	0	23
K	0	0	0	0	0	0	0	0	0	6	0

Langkah selanjutnya yaitu menghitung matriks *dependency measure*. Formula di Persamaan 2.1 dapat menghasilkan matriks *dependency measure* seperti Tabel 3.22:

Tabel 3.22 Matriks *Dependency Measure*

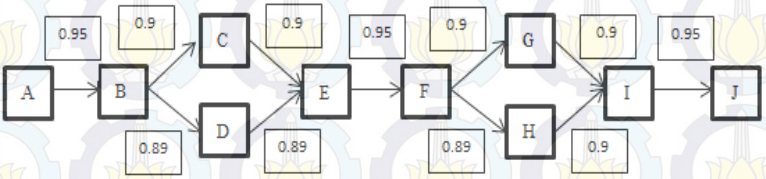
Activity	A	B	C	D	E	F	G	H	I	J	K
A	0	0.947	0	0	0	0	0	0	0	0	0
B	0	0	0.9	0.889	0	0	0	0	0	0	0
C	0	0	0	0	0.9	0	0	0	0	0	0
D	0	0	0	0	0.9	0	0	0	0	0	0
E	0	0	0	0	0	0.947	0	0	0	0	0
F	0	0	0	0	0	0	0.9	0.889	0	0	0
G	0	0	0	0	0	0	0	0	0.9	0	0
H	0	0	0	0	0	0	0	0	0.9	0	0
I	0	0	0	0	0	0	0	0	0	0.947	0
J	0	0	0	0	0	0	0	0	0	0	0.567
K	0	0	0	0	0	0	0	0	0	-0.567	0

Dengan menggunakan Persamaan 3.1, Persamaan 3.2, Persamaan 3.3, dan Persamaan 3.4 maka didapatkan nilai *threshold* sebagai berikut:

- *Average of positive dependency measure (PDM) = 0.884*
- *Standard deviation of PDM = 0.102*

- *Relative-to-best threshold = 0.833*
- *Positive observations threshold = 4*
- *Dependency threshold = 0.781*

Dengan menggunakan nilai dari *threshold* tersebut maka didapatkan *dependency graph* sebagi berikut:



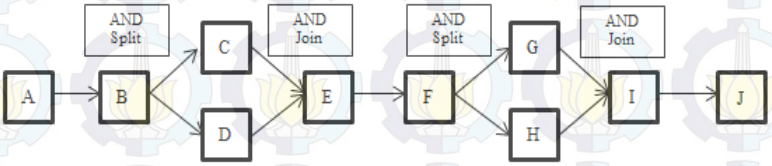
Gambar 3.16 *Dependency Graph* dengan *Dependency Measure*

Dengan menggunakan *causal matrix* yang sama seperti Tabel 3.19, tetapi perhitungan *parallel measure* yang berbeda. *Heuristics Miner for Time Intervals* menggunakan Persamaan 2.5 untuk menghitung *parallel measure*.

$B \rightarrow w^{C \wedge D} = 2.684$ $\quad E \rightarrow w^{C \wedge D} = 2.684$

$F \rightarrow w^{G \wedge H} = 1.421$ $\quad I \rightarrow w^{G \wedge H} = 1.421$

Dengan menggunakan *rule* dari *Heuristics Miner for Time Intervals*, jika *parallel measure* lebih dari 0.1 maka *split* dan *join* yang digunakan adalah AND. Oleh karena itu, proses model yang di-*discover* dengan menggunakan *Heuristics Miner for Time Intervals* menjadi:



Gambar 3.17 Proses Model Akhir dengan Algoritma *Heuristics Miner for Time Intervals*

3.9 Perbandingan Kualitas *Fitness*

Berdasarkan pengertian dari kualitas *fitness* yang telah dijelaskan pada Bagian 2.13, kita dapat membandingkan dan mengevaluasi kualitas *fitness* dari proses model yang ditemukan dari masing-masing algoritma *discovery*. Menggunakan *parsing measure* PM (Persamaan 2.7) dan *continuous parsing measure CPM* (Persamaan 2.8), nilai *fitness* untuk masing-masing algoritma seperti pada Tabel 3.23 dan Tabel 3.24:

Tabel 3.23 Nilai *fitness* dari *Parsing Measure PM*

The Parsing Measure PM			
PM	c	T	Method
0.9408867	191	203	Proposed Method
0.81773399	166	203	Original Heuristics Miner
0.886699507	180	203	Time-Based Discovery
0.852216749	173	203	Heuristics Miner for Time Intervals

Tabel 3.24 Nilai *fitness Continuous Parsing Measure CPM*

The Continuous Parsing Measure CPM				
CPM	e	m	r	Method
0.954545455	198	10	8	Proposed Method
0.954545455	198	10	8	Original Heuristics Miner
0.952020202	198	11	8	Time-Based Discovery
0.909090909	198	20	16	Heuristics Miner for Time Intervals

Nilai-nilai *fitness* dan variabel yang terlibat pada *event log* di Tabel 3.12 yang di-*mining* menggunakan empat algoritma berbeda ditunjukkan pada Tabel 3.23 dan Tabel 3.24. Jumlah *trace* yang dapat diurai dengan benar c telah meningkat dan jumlah yang dari aktivitas yang hilang dan yang tersisa (m dan r) mengalami penurunan, yang berarti bahwa proses model yang ditemukan sesuai dengan realitas apabila di-*mining* menggunakan *Modified Time-based Heuristics Miner*. Algoritma *Modified Time-Based Heuristics Miner* lebih baik dari proses model yang ditemukan oleh algoritma *original*

Heuristics Miner, Process Model Discovery Based on Activity Lifespan dan *Heuristics Miner for Time Intervals*.

3.10 Evaluasi *Completeness* dari Proses Model

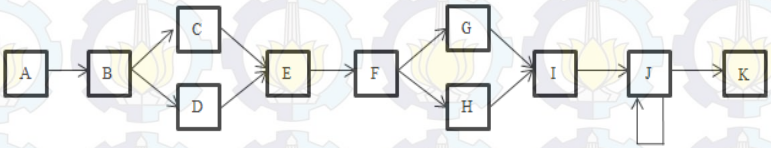
Berdasarkan definisi *completeness*, *Length Two Loop dependency measure* antara dua aktivitas J dan K tidak dihitung karena memenuhi salah satu dari tiga kondisi berikut:

(1) J adalah aktivitas *Length One loop* dan $|J >>_W K|$ lebih tinggi dari *positive observation threshold* (POT) atau,
= J adalah aktivitas *Length One loop* dan $|J >>_W K| = 6$ lebih tinggi dari POT = 4.

(2) K adalah aktivitas *Length One loop* dan $|K >>_W J|$ lebih tinggi dari *positive observation threshold* (POT) atau,

(3) Semua kondisi di atas.

Sehingga, berdasarkan kondisi di atas, untuk *event log* di Tabel 3.12, *Length Two Loop dependency* tidak dimasukkan ke dalam proses model akhir. Tetapi berdasarkan perhitungan matriks *dependency Length Two Loop*, hasil dari JK dan KJ adalah 0.923, lebih tinggi daripada *Dependency Threshold* (0.782) sehingga relasi antara aktivitas J dan aktivitas K adalah *sequence*. Proses model akhir dari metode yang diusulkan seperti pada Gambar 3.18:



Gambar 3.18 Proses Model Akhir dengan Algoritma *Modified Time-Based Heuristics Miner*

3.11 Evaluasi *Validity* dari Proses Model

Berdasarkan definisi dari *validity*, valid berarti proses model yang ditemukan dapat menghasilkan dan mendefinisikan perubahan dari *dependency graph* ke proses model semantik. Kita perlu mengetahui dua *step* untuk membuat proses model

semantik. Pertama, membuat *dependency graph* menjadi *causal net* dan kedua, memodelkan kembali *causal net* menjadi proses model semantik.

Dari proses model yang ditemukan pada Gambar 3.9 menggunakan algoritma *Modified Time-Based Heuristics Miner*, proses model semantik dapat dihasilkan sebagai berikut:

Tabel 3.25 *Dependency Graph* dari *Event Log*

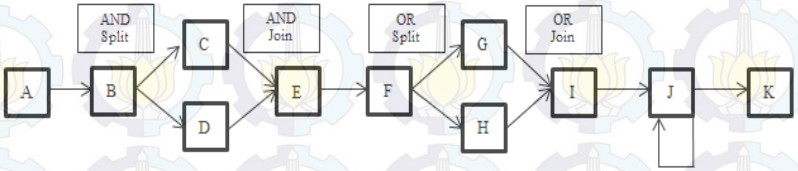
INPUT SET	ACTIVITY	OUTPUT SET
{Ø}	A	{B}
{A}	B	{C,D}
{B}	C	{E}
{B}	D	{E}
{C,D}	E	{F}
{E}	F	{G,H}
{F}	G	{I}
{F}	H	{I}
{G.H}	I	{J}
{I}	J	{K}
{J}	K	{Ø}

Tabel 3.26 *Causal Net* dari *Event Log*

INPUT SET	ACTIVITY	OUTPUT SET
{Ø}	A	{{B}}
{{A}}	B	{{C},{D}}
{{B}}	C	{{E}}
{{B}}	D	{{E}}
{{C},{D}}	E	{{F}}
{{E}}	F	{{G},{H}}
{{F}}	G	{{I}}
{{F}}	H	{{I}}
{{G}.{H}}	I	{{J}}
{{I}}	J	{{K}}
{{J}}	K	{Ø}

Algoritma *Modified Time-Based Heuristics Miner*, dapat menemukan *length one loop, length two loop, parallel AND,*

conditional OR, dan *single choice XOR* yang sudah ditunjukkan pada Bagian 3.4 dan Bagian 3.5. Proses model akhir dari algoritma yang diusulkan pada Tugas Akhir ini dapat dilihat pada Gambar 3.19.



Gambar 3.19 Proses Model Akhir dari *Modified Time-Based Heuristics Miner*

3.12 Optimasi Waktu dan Biaya Proses Bisnis dalam Manajemen Proyek

Optimasi yang dilakukan dalam Tugas Akhir ini adalah optimasi antara tambahan biaya yang harus dibayarkan untuk memampatkan waktu eksekusi (*makespan*) dari proses bisnis. Kasus ini juga dapat disebut dengan *Time-cost Tradeoff Prolem.* Tugas akhir ini menggunakan metode *Crasing Project* yang ada pada CPM. *Crashing Project* sendiri merupakan suatu metode untuk mempersingkat lamanya waktu proyek dengan mengurangi waktu dari satu atau lebih aktivitas proyek yang penting menjadi kurang dari waktu normal aktivitas (Elmabrouk, 2011).

Terdapat dua nilai waktu yang ditunjukkan tiap aktivitas dalam suatu jaringan kerja saat terjadi c*rashing project* yaitu:

1. *Normal Duration*
   Waktu yang dibutuhkan untuk menyelesaikan suatu aktivitas atau kegiatan dengan sumber daya normal yang ada tanpa adanya biaya tambahan lain dalam sebuah proyek.
2. *Crash Duration*
   Waktu yang dibutuhkan suatu proyek dalam usahanya mempersingkat waktu yang durasinya lebih pendek dari *normal duration.*

Untuk membuat model matematika yang digunakan untuk optimasi diperlukan *time*. *Time slope* sendiri merupakan percepatan maksimum yang dapat dilakukan untuk *crashing project.*

$$X_i = T_{ni} - T_{ci} \quad (3.12)$$

Keterangan:
X_i : *Time slope* aktivitas ke – i dimana i ∈ {Aktivitas}
T_{ni} : durasi normal aktivitas ke – i dimana i ∈ {Aktivitas}
T_{ci} : *crash* duration aktivitas ke – i dimana i ∈ {Aktivitas}

Crashing Project juga menyebabkan perubahan pada elemen biaya yaitu:

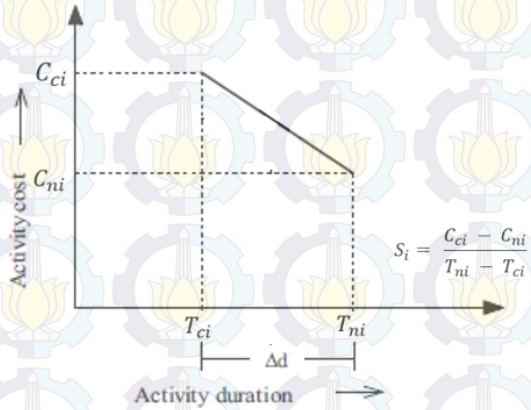
1. *Normal Cost*
   Biaya yang dikeluarkan dengan penyelesaian proyek dalam waktu normal. Perkiraan biaya ini adalah pada saat perencanaan dan penjadwalan bersamaan dengan penentuan waktu normal.
2. *Crash Cost*
   Biaya yang dikeluarkan dengan penyelesaian proyek dalam jangka waktu sebesar durasi *crash*-nya. Biaya setelah di *crashing* akan menjadi lebih besar dari biaya normal.

Perhitungan *Time-Cost TradeOff* diutamakan pada kegiatan-kegiatan yang memiliki nilai *cost slope* terendah. *Cost slope* merupakan perbandingan antara pertambahan biaya dengan percepatan waktu penyelesaian proyek. Persamaan 3.13 adalah sebagai berikut:

$$S_i = \frac{C_{ci} - C_{ni}}{T_{ni} - T_{ci}} \quad (3.13)$$

Keterangan:
S_i : *cost slope* aktivitas ke – i dimana i ∈ {Aktivitas}
C_{ni} : *cost* normal aktivitas ke – i dimana i ∈ {Aktivitas}

C_{ci} : *cost crash* aktivitas ke – i dimana i ∈ {Aktivitas}
T_{ni} : durasi normal aktivitas ke – i dimana i ∈ {Aktivitas}
T_{ci} : *crash duration* aktivitas ke- i dimana i ∈ {Aktivitas}



Gambar 3.20 *Cost Slope*

3.12.1. Penentuan Durasi Normal dan *Cost* Normal

Penentuan durasi dengan rata-rata sudah dikembangkan oleh Van Der Aalst, tetapi yang yang dicari merupakan durasi dari eksekusi satu *trace* dalam *event log* (van der Aalst, Schonenberg, & Song, Time Prediction Based on Process Mining, 2011). Oleh karena itu dalam Tugas Akhir ini dikembangkan metode untuk menentukan durasi normal dan biaya normal dari setiap aktivitas dengan merata-rata setiap durasi dan biaya dari aktivitas pada setiap *case* yang ada pada *event log*. Durasi eksekusi diperoleh dengan mengurangkan *output* dari aktivitas dengan aktivitas (aktivitas yaitu A memiliki aktivitas B sebagai *output* maka eksekusi durasi A adalah waktu eksekusi B mengurangkan dengan waktu pelaksanaan A).

$$if\ activity_{output} = true\ then$$
$$ed = et_{activity_{output}} - et_{activity} \quad (3.14)$$

Keterangan:
ed : durasi eksekusi per *case*
$et_{activity_{output}}$: waktu eksekusi *output* dari aktivitas
$et_{activity}$: waktu eksekusi aktivitas

Setelah mendapatkan semua eksekusi durasi setiap aktivitas pada semua kasus di *event log* maka rata-rata pelaksanaan durasi per aktivitas dihitung untuk menentukan durasi normal.

$$T_{ni} = \frac{\sum_{i=1}^{h} ed_i}{h} \quad (3.15)$$

Setelah menghitung durasi dari menghitung biaya normal setiap aktivitas. *Event log* berisi biaya pelaksanaan setiap aktivitas dalam setiap *case*, oleh karena itu untuk mendapatkan biaya normal setiap aktivitas digunakan perhitungan biaya rata-rata aktivitas dalam setiap kasus.

$$C_{ni} = \frac{\sum_{i=1}^{h} cost_i}{h} \quad (3.16)$$

Keterangan:
T_{ni} : durasi normal setiap aktivitas ke-i dimana i ∈ {Aktivitas}
ed_q : waktu eksekusi aktivitas pada *case* ke-q dimana q = 1, 2, 3, … w
C_{ni} : *cost* normal setiap aktivitas ke-i dimana i ∈ {Aktivitas}
$cost_q$: biaya aktivitas pada *case* ke-i ke-q dimana q = 1, 2, … w
h : banyaknya aktivitas dieksekusi dalam *event log* (banyaknya *case* yang mengandung aktivitas)

3.12.2. Penentuan *Crash Time* dan *Crash Cost*

Dalam menentukan *crash time* dan *crash cost* dari setiap aktivitas diambil dengan mengurangi normal durasi dengan standar deviasi waktu eksekusi masing-masing aktivitas untuk *crash time* dan menambahkan biaya normal dengan standar deviasi biaya dari masih-masing aktivitas untuk *crash cost*.

3.12.3. Pemodelan Fungsi Non-Linear untuk Optimasi

Dalam pemodelan fungsi matematika untuk model optimasi ini dengan menggunakan *upper bound* dan *lower bound serta* fungsi batasan.

Upper Bound dan *Lower Bound*
Upper bounds dan *lower bounds* dari durasi setiap aktivitas perlu untuk diketahui. Durasi aktivitas akan lebih terbagi dengan baik dengan menggunakan *range* dari *lower bound* dan *upper bound* (Richard F. Deckro, John E. Hebert, William A. Verdini, Per Henning Grimsrud, Satya Venkateshwar, 1994).

$$lower\ bound \leq duration\ of\ activity\ (i,j) \leq upper\ bound \quad for\ all\ (i,j) \in Activity \tag{3.17}$$

Constraint:
$lower\ bound$: *lower bound* durasi aktivitas (i,j);
lower bound ≥ 0, *lower bound* $\leq$ *upper bound*
$upper\ bound$: *upper bound* durasi aktivitas (i,j);
upper bound ≥ 0

Langkah-langkah optimasi *Time* dan *Cost* dengan *Non-Linear Programming:*

1. Hitung normal durasi dan minimum durasi dari proses bisnis.
2. Temukan *critical path* dari proses bisnis menggunakan *Critical Path Method* (CPM).

3. *Update additional cost* (min Z) dengan pemodelan non-linear.

$$Min\ Z = \sum(Normal\ Cost\ activity\ i + (Cost\ Slope\ activity\ i * (Normal\ Duration\ activity\ i - Crash\ Duration\ activity\ i)^{2})) \quad (3.18)$$

3.13 Contoh Optimasi Waktu dan Biaya

Optimasi waktu dan biaya menggunakan proses model seperti pada Tabel 3.27.

Tabel 3.27 Input Output Graf

INPUT	Relasi	OUTPUT
A	Sequence	B
B	AND-Split	C,D
C,D	AND-Join	E
E	Sequence	F
F	OR-Split	G,H
G,H	OR-Join	I
I	Sequence	J
J	Sequence	K

Graf di Tabel 3.27 memiliki *event log* seperti pada Tabel 3.28.

Tabel 3.28 *Event Log* untuk optimasi *Time* dan *Cost*

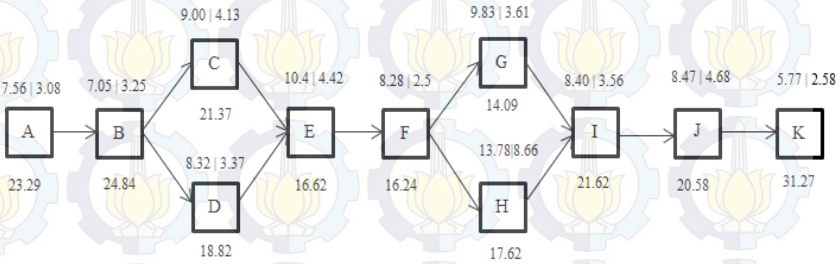
Case ID	Activity	Start Stamp	End Stamp	Cost	Resource
PP1	A	6/20/2014 8:32	6/20/2014 13:42	800	1
	B	6/20/2014 13:42	6/20/2014 23:41	600	1
	C	6/20/2014 23:41	6/21/2014 9:30	800	1
	D	6/21/2014 8:16	6/21/2014 10:46	500	1
	E	6/21/2014 10:46	6/21/2014 16:57	650	1
	F	6/21/2014 16:57	6/21/2014 18:09	700	1
	G	6/21/2014 18:09	6/22/2014 4:14	750	1
	H	6/21/2014 19:15	6/22/2014 10:51	600	1
	I	6/22/2014 10:51	6/22/2014 16:28	800	1
	J	6/22/2014 16:28	6/22/2014 23:42	600	1
	K	6/22/2014 23:42	6/23/2014 4:48	500	1
PP2	A	6/23/2014 4:48	6/23/2014 16:44	650	1
	B	6/23/2014 16:44	6/23/2014 23:26	700	1
	C	6/23/2014 23:26	6/24/2014 5:40	750	1
	D	6/24/2014 4:19	6/24/2014 7:48	600	1
	E	6/24/2014 7:48	6/25/2014 1:08	800	1
	F	6/25/2014 1:08	6/25/2014 3:02	500	1
	G	6/25/2014 3:02	6/25/2014 5:11	650	1
	I	6/25/2014 5:11	6/25/2014 8:25	700	1
	J	6/25/2014 8:25	6/25/2014 12:45	750	1
	K	6/25/2014 12:45	6/26/2014 0:12	600	1
PP3	A	6/26/2014 0:12	6/26/2014 4:48	800	1
	B	6/26/2014 4:48	6/26/2014 15:16	600	1
	D	6/26/2014 15:16	6/27/2014 1:52	600	1
	C	6/26/2014 22:49	6/27/2014 5:19	500	1
	E	6/27/2014 5:19	6/27/2014 11:40	650	1
	F	6/27/2014 11:40	6/27/2014 20:00	700	1
	H	6/27/2014 20:00	6/28/2014 7:10	750	1
	I	6/28/2014 7:10	6/28/2014 9:10	600	1
	J	6/28/2014 9:10	6/28/2014 16:40	800	1
	K	6/28/2014 16:40	6/28/2014 19:29	800	1
PP4	A	6/28/2014 19:29	6/28/2014 23:28	500	1
	B	6/28/2014 23:28	6/29/2014 9:12	650	1
	D	6/29/2014 9:12	6/29/2014 18:41	700	1
	C	6/29/2014 15:27	6/29/2014 20:50	750	1
	E	6/29/2014 20:50	6/30/2014 2:55	600	1
	F	6/30/2014 2:55	6/30/2014 9:14	800	1
	H	6/30/2014 9:14	6/30/2014 14:37	600	1
	G	6/30/2014 11:25	6/30/2014 17:33	600	1
	I	6/30/2014 17:33	7/1/2014 6:48	500	1
	J	7/1/2014 6:48	7/1/2014 9:41	750	1
	K	7/1/2014 9:41	7/1/2014 11:15	600	1
PP5	A	7/1/2014 11:15	7/1/2014 15:01	800	1
	B	7/1/2014 15:01	7/1/2014 21:44	600	1
	C	7/1/2014 21:44	7/2/2014 4:30	500	1
	E	7/2/2014 4:30	7/2/2014 14:50	650	1
	F	7/2/2014 14:50	7/3/2014 3:06	700	1
	G	7/3/2014 3:06	7/3/2014 22:01	750	1
	H	7/3/2014 14:41	7/4/2014 1:40	600	1
	I	7/4/2014 1:40	7/4/2014 9:02	800	1
	J	7/4/2014 9:02	7/4/2014 17:12	800	1
	K	7/4/2014 17:12	7/4/2014 20:56	500	1
PP6	A	7/4/2014 20:56	7/5/2014 10:21	650	1
	B	7/5/2014 10:21	7/5/2014 13:22	700	1
	C	7/5/2014 13:22	7/5/2014 17:56	750	1
	D	7/5/2014 16:26	7/5/2014 19:35	600	1
	E	7/5/2014 19:35	7/6/2014 0:37	800	1
	F	7/6/2014 0:37	7/6/2014 10:59	600	1
	G	7/6/2014 10:59	7/6/2014 21:18	600	1
	I	7/6/2014 21:18	7/7/2014 2:43	500	1
	J	7/7/2014 2:43	7/7/2014 17:13	650	1
	K	7/7/2014 17:13	7/8/2014 0:31	800	1

Dari Tabel 3.28 didapatkan hasil perhitungan normal durasi, *crash* durasi, normal *cost*, *crash cost*, *time slope* dan *cost slope* dengan cara yang telah disebutkan pada Bagian 3.12. Hasil perhitungan ditunjukkan oleh Tabel 3.29.

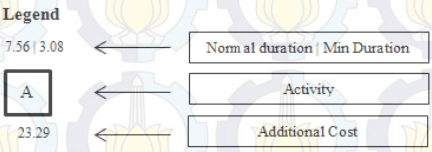
Tabel 3.29 Hasil Perhitungan

Activity	Normal Duration	Crash Duration	Normal Cost	Crash Cost	Time Slope	Cost Slope
A	7.56	3.08	656.67	760.99	4.48	23.29
B	7.05	3.25	673.33	767.61	3.80	24.84
C	9.00	4.13	610.00	714.04	4.87	21.37
D	8.32	3.37	675.00	768.15	4.95	18.82
E	10.40	4.42	650.00	749.34	5.98	16.62
F	8.28	2.50	660.00	753.76	5.77	16.24
G	9.83	3.61	750.00	837.71	6.22	14.09
H	13.78	8.66	670.00	760.23	5.12	17.62
I	8.40	3.56	680.00	784.67	4.84	21.62
J	8.47	4.68	668.33	746.36	3.79	20.58
K	5.77	2.58	642.86	742.42	3.18	31.27

Setelah menerapkan prediksi waktu ke proses model, proses model siap untuk dioptimasi seperti pada Gambar 3.21.



Gambar 3.21 Proses Model untuk Optimasi



Langkah-langkah optimasi proses bisnis sebagai berikut:

1. Hitung normal durasi dan minimum durasi dari proses model. Normal durasi dan minimum durasi dijelaskan pada Gambar 3.21.
2. *Critical path* yang ditemukan dengan menggunakan CPM dijelaskan pada Gambar 3.21. *Critical path* terdiri dari 11 aktivitas, yaitu A, B, C, D, E, F, G, H, I, J dan K. Total normal durasi adalah 96.85. Total minimum durasi adalah 43.84.
3. *Update additional cost* (Min Z) dengan menggunakan *non-linear modeling* menggunakan Persamaan 3.18.

$$Min\ Z = \sum(Normal\ Cost\ activity\ i + (Cost\ Slope\ activity\ i * (Normal\ Duration\ activity\ i - Crash\ Duration\ activity\ i)^{2}))$$

$Min\ Z$ = (656.67+(23.29*(7.56-3.08)^2)) + (673.33+(24.84*(7.05-3.25)^2)) + (610+(21.37*(9.00-4.13)^2)) + (675+(18.82*(8.32-3.37)^2)) + (650+(16.62*(10.40-4.42)^2)) + (660+(16.24*(8.28-2.50)^2)) + (750+(14.09*(9.83-3.61)^2)) + (670+(17.62*(13.78-8.66)^2)) + (680+(21.62*(8.40-3.56)^2)) + (668.33+(20.58*(8.47-4.68)^2)) + (642.86+(31.27*(5.77-2.58)^2))

$Min\ Z$ = (1.124,1096) + (1.032,0196) + (1.116,8302) + (1.136,1371) + (1.244,3378) + (1.202,5524) + (1.295,1196) + (1.131,8977) + (1.186,4615) + (963.9432) + (961.0666)

$Min\ Z$ = 12.394,4753

Sehingga diperoleh, hasil optimasi waktu adalah 43.84 dengan penambahan biaya minimum sebesar 12.394,4753.

Aktivitas	**Sebelum Optimasi**	**Setelah Optimasi**
Biaya Tambahan		12.394,475
***Durasi* (jam)**	96.85	43.84

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan Tugas Akhir. Analisis permasalahan membahas permasalahan yang yang diangkat dalam pengerjaan Tugas Akhir. Solusi yang ditawarkan oleh penulis juga dicantumkan pada tahap permasalahan analisis ini. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat. Perancangan direpresentasikan dengan diagram UML (*Unified Modelling Language*).

4.1 Analisis

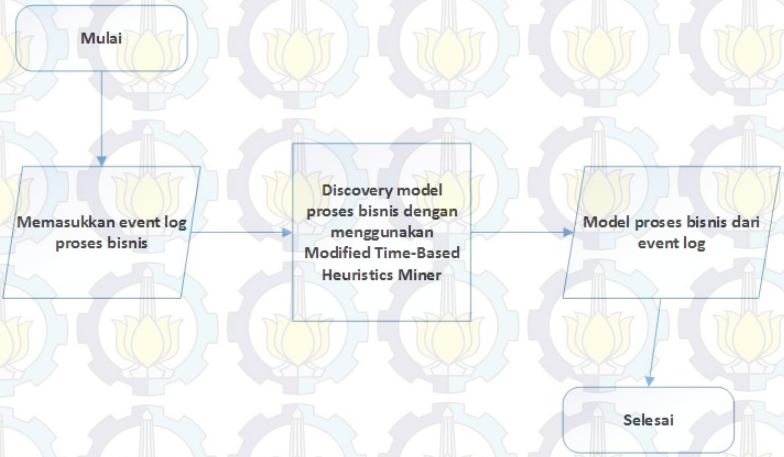
Tahap analisis dibagi menjadi beberapa bagian antara lain cakupan permasalahan, deskripsi umum sistem, kasus penggunaan sistem, dan kebutuhan perangkat lunak.

4.2 Deskripsi Umum Sistem

Perangkat lunak yang akan dibangun dapat menghasilkan model dari proses bisnis menggunakan metode yang diusulkan, yaitu *Modified Time-Based Heuristics Miner*. Proses pemodelan proses bisnis tersebut membutuhkan data atau *event log* proses bisnis yang sudah ada dan berjalan. Dari *event log* yang digunakan sebagai masukan dilakukan serangkaian proses hingga menghasilkan keluaran yang diharapkan. Gambar 4.1 merupakan alur pemrosesan dan bentuk arsitektur perangkat secara sederhana.

Pada fase *discovery*, pengguna akan memilih *file event log* yang akan di-*discover* bentuk proses modelnya. Setelah pengguna memilih *file event log* yang di-*discover* bentuk proses modelnya, hal yang paling awal dilakukan adalah melakukan proses pada *file event log* yang dipilih dengan menuliskan nama *file* Excel dengan benar pada halaman kerja Python dengan bantuan *library numpy* (Python Package Index – numpy 1.10.4,

https://pypi.python.org/pypi/numpy, 5 Desember, 2015), *xlrd* dan *xlwt* (Working with Excel Files in Python – xlrd and xlwt, http://www.python-excel.org/, 5 Desember, 2015). Setiap elemen-elemen *case ID* pada *file* Excel *event log* diidentifikasi. *Trace-trace* yang terdapat pada *event log* dan juga frekuensi *trace* yang ada pada event log juga diidentifikasi. Setelah mendapatkan *trace* dan masing-masing frekuensinya, kemudian menggunakan modifikasi dari *heuristics miner* dicari hubungan dari masing-masing aktivitas dari *event log*, sehingga dapat menghasilkan proses model dari *event log* yang dipilih oleh pengguna.



Gambar 4.1 Alur Proses Perangkat Lunak

4.3 Spesifikasi Kebutuhan Perangkat Lunak

Bagian ini berisi semua kebutuhan perangkat lunak yang diuraikan secara rinci dalam bentuk diagram kasus, diagram urutan, dan diagram aktivitas. Masing-masing diagram menjelaskan perilaku atau sifat dari sistem ini. Kebutuhan perangkat lunak dalam sistem ini mencakup kebutuhan fungsional saja. Pada bab ini juga dijelaskan tentang spesifikasi terperinci pada masing-masing kebutuhan fungsional. Rincian spesifikasi dari kasus penggunaan disajikan dalam bentuk tabel.

4.3.1. Kebutuhan Fungsional

Kebutuhan fungsional berisi kebutuhan utama yang harus dipenuhi oleh sistem agar dapat bekerja dengan baik. Kebutuhan fungsional mendefinisikan layanan yang harus disediakan oleh sistem, bagaimana reaksi terhadap masukan, dan apa yang harus dilakukan sistem pada situasi khusus. Daftar kebutuhan fungsional dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar Kebutuhan Fungsional Perangkat Lunak

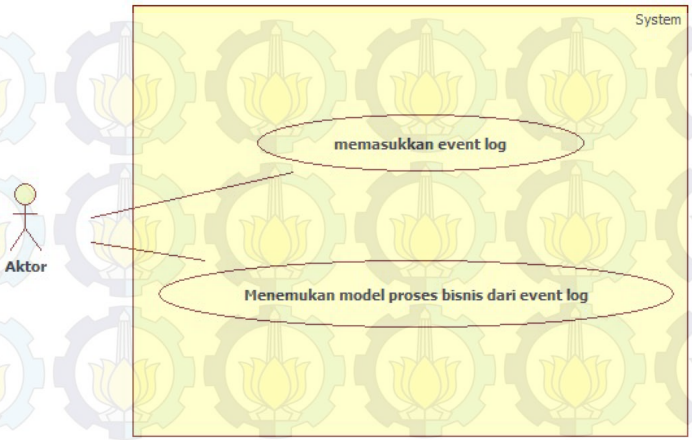
Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
TA-F0001	Memasukkan event log	Pengguna dapat memasukkan *event log* dalam bentuk Excel
TA-F0002	Menemukan bisnis proses model	Pengguna dapat menemukan bisnis proses model sehingga mengetahui bentuk *graph* bisnis proses dan hubungan antar aktivitas.

4.3.2. Aktor

Aktor mendefinisikan entitas-entitas yang terlibat dan berinteraksi langsung dengan sistem. Entitas ini bisa berupa manusia maupun sistem atau perangkat lunak yang lain. Penulis mendefinisikan aktor untuk sistem ini yaitu pengguna dari aplikasi yang dibangung oleh penulis.

4.4 Kasus Penggunaan

Kasus-kasus penggunaan dalam sistem ini akan dijelaskan secara rinci pada subbab ini. Kasus penggunaan secara umum akan digambarkan oleh salah satu model UML, yaitu diagram kasus penggunaan. Rincian kasus penggunaan berisi spesifikasi kasus penggunaan, diagram aktivitas, dan diagram urutan untuk masing-masing kasus penggunaan. Diagram kasus penggunaan dapat dilihat pada Gambar 4.2. Daftar kode diagram kasus penggunaan sistem dapat dilihat pada Tabel 4.2.



Gambar 4.2 Diagram Kasus Penggunaan

Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan

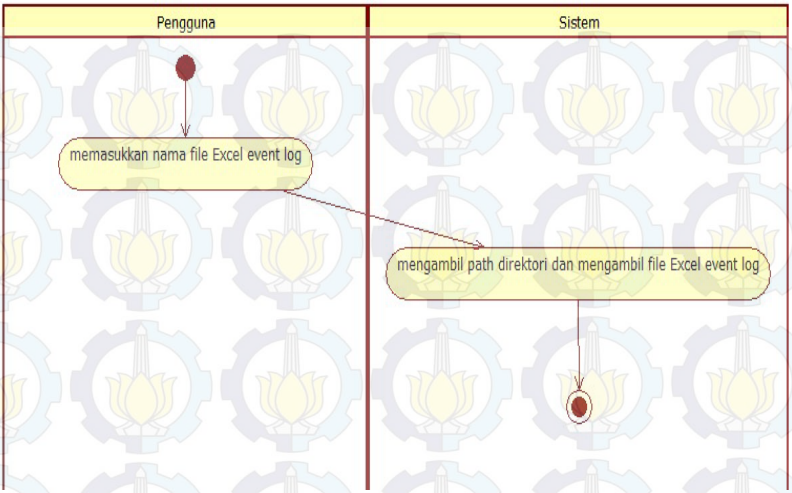
Kode Kasus Penggunaan	Nama
TA-UC0001	Memasukkan data *event log*
TA-UC0002	Menemukan model proses bisnis

4.4.1. Memasukkan Data *Event Log*

Pada kasus penggunaan ini, pengguna akan memasukkan data yang berupa *event log* proses bisnis dalam notasi Excel. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 4.3 dan diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 4.3.

Tabel 4.3 Spesifikasi Kasus Penggunaan Memasukkan *Data Event Log*

Nama	Memasukkan data *event log*
Kode	TA-UC0001
Deskripsi	Pengguna memasukkan nama *file event log* yang akan diolah oleh sistem
Tipe	Fungsional
Pemicu	Pengguna memasukkan nama *file* Excel *event log* yang akan diolah oleh sistem
Aktor	Pengguna
Kondisi Awal	*Event log* yang akan diproses belum ada
Aliran:	
-Kejadian Normal	1. Pengguna memasukkan nama *file* Excel *event log* ke dalam program 2. Sistem mengambil *path* direktori dan mengambil *file* Excel *event log* ke dalam sistem
-Kejadian Alternatif	Tidak ada
Kondisi Akhir	Sistem berhasil menyimpan nama *file* Excel *event log*
Kebutuhan Khusus	Tidak ada



Gambar 4.3 Diagram Aktivitas Memasukkan Data *Event Log*

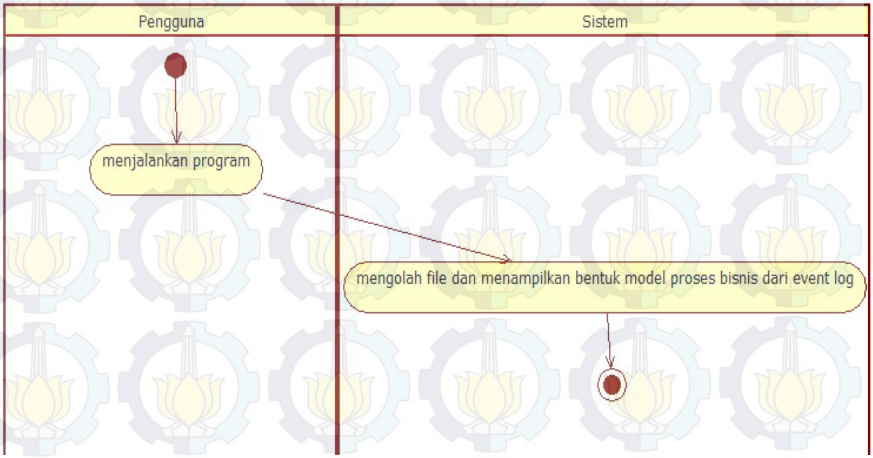
4.4.2. Menemukan Model Proses Bisnis

Pada kasus penggunaan ini, sistem akan menerima perintah dari pengguna untuk melakukan proses *discovery* pada *event log* yang telah dimasukkan oleh pengguna. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 4.4 dan diagram aktivitas dan diagram urutan dari kasus penggunaan ini bisa dilihat pada Gambar 4.4.

Tabel 4.4 Spesifikasi Kasus Penggunaan Menemukan Model Proses Bisnis

Nama	Menemukan model proses bisnis
Kode	TA-UC0002
Deskripsi	Pengguna dapat menemukan bisnis proses model sehingga mengetahui bentuk *graph* bisnis proses dan hubungan antar aktivitas.
Tipe	Fungsional
Pemicu	Pengguna menjalankan program
Aktor	Pengguna

Kondisi Awal	Nama *file* Excel *event log* dalam sudah terisi dengan benar
Aliran:	
-Kejadian Normal	1. Pengguna menjalankan program 2. Sistem mengolah *file event log* dan menampilkan bentuk model proses bisnis dari *event log*.
-Kejadian Alternatif	-
Kondisi Akhir	Sistem menampilkan *graph* model proses bisnis dengan benar dan lengkap
Kebutuhan Khusus	Tidak ada

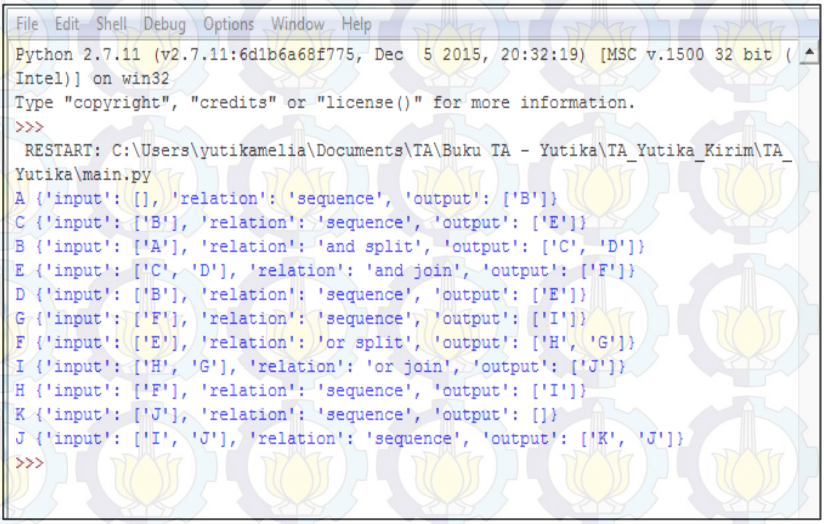


Gambar 4.4 Diagram Aktivitas Menemukan Model Proses Bisnis

4.5 Perancangan Sistem

Penjelasan tahap perancangan perangkat lunak yaitu perancangan proses analisis sehingga menghasilkan *output* model proses bisnis dari *event log*.

4.5.1. Halaman Hasil *Process Discovery*



Gambar 4.5 Halaman Hasil *Process Discovery*

Halaman ini merupakan halaman yang muncul ketika aplikasi dijalankan. Setelah *file* terbuka maka sistem akan melakukan proses *discovery* dan menampilkan hasil berupa *input*, *output*, dan relasi dari proses model.

BAB V
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari modul *processing* dan *main program*. Bahasa pemrograman yang digunakan adalah bahasa pemrograman Python.

5.1 Lingkungan Implementasi

Lingkungan implementasi merupakan lingkungan dimana sistem ini dibangun, dimana akan dijelaskan mengenai kebutuhan perangkat yang diperlukan untuk membangun sistem ini. Lingkungan implementasi dibagi menjadi dua, yaitu lingkungan implementasi terhadap perangkat keras dan lingkungan implementasi terhadap perangkat lunak.

5.1.1. Perangkat Keras

Implementasi dilakukan pada sebuah Laptop dengan spesifikasi sebagai berikut:

- Merk : Toshiba
- Seri : Satellite L840
- Processor : Processor Intel Core i5-2310M CPU @ 2.10GHz
- RAM : 4 GB

5.1.2. Perangkat Lunak

Suatu sistem atau perangkat lunak belum tentu bisa berdiri sendiri, dan sistem ini membutuhkan perangkat lunak lain yang dapat mendukung fungsionalitasnya, yang antara lain:

1. Sistem Operasi Windows
   Sistem operasi yang digunakan adalah Microsoft Windows 7 Professional 32-bit.
2. Python-2.7.11

Kakas bantu yang digunakan dalam mengembangkan sistem ini adalah python-2.7.11 dengan bahasa pemrograman Python.

3. *Library Python* yaitu *numpy, xlrd, dan xlwt*
   Library Python tersebut digunakan untuk menunjang proses pengerjaan, salah satunya adalah untuk membaca file Excel.

5.2 Penjelasan Implementasi

Pada subbab ini dijelaskan implementasi setiap metode yang dijelaskan pada bab metode pemecahan masalah, sehingga terbentuk suatu perangkat lunak yang mengimplementasi metode-metode pada bab metode pemecahan masalah.

5.2.1. Implementasi Algoritma *Modified Time-Based Heuristics Miner*

Pada bagian ini mengimplementasi algoritma *modified time-based heuristics miner* agar dapat melakukan proses *discovery* terhadap *event log* dan mendapatkan model proses bisnis yang benar. Proses dalam implementasi ini terdapat dua jenis yaitu:

- Pertama proses pembacaan *file event log* dan sekaligus melakukan pendefinisian *timestamp* yang terdapat dalam *event log*. Hal ini dapat dilihat pada Kode Sumber 5.1.

```
def dateToString(self,data):
        dates = ''
        for x in range (0,len(data)):
            if x == len(data)-1:
                break
            if int(data[x]) < 10 :
                dates = dates+'0'+str(data[x])
            else :
                dates = dates+''+str(data[x])
        return dates

def readData(self, file):
        wb = xlrd.open_workbook(filename=file)
        data = wb.sheet_by_index(0)
```

```
        list = dict()
        cur = ''
        for x in range(1,data.nrows):
            content = data.row(x)
            if content[0].value != '':
                cur = content[0].value
                list[str(content[0].value)] = []
            tmp = []
            for idx,cell_obj in enumerate(content):
                if idx == 0:
                    continue
                if idx == 2 or idx ==  3:
                    times=
self.dateToString(xlrd.xldate_as_tuple(cell_obj.value,
wb.datemode))
                    tmp.append(times)
                    continue
                tmp.append(str(cell_obj.value))
            list[cur].append(tmp)
        return list
```

Kode Sumber 5.1 Fungsi `Modified Time-Based HeuristicsMiner` bagian Membaca Data dan Mendefiniskan *Time Stamp*

- Yang kedua adalah bagian mendapatkan aktivitas-aktivitas yang waktu eksekusinya *overlap*. Pengimplementasian dapat dilihat pada Kode Sumber 5.2

```
def checkOverLap(self,data1,data2):
        if (data1[0]>data2[0] and data1[0]<data2[1]) or
(data1[1]>data2[0] and data1[1]<data2[1]):
            return True
        return False

def getOverlap(self,rawData):
        listTask = self.getTask(rawData)
        matrix = [[1 for x in range(len(listTask))] for x in
range(len(listTask))]
        matrixCountOverlap = [[0 for x in
range(len(listTask))] for x in range(len(listTask))]
        tmpMatrixOverlap = [[0 for x in
range(len(listTask))] for x in range(len(listTask))]

        for x in rawData:
            for index in range (0,len(rawData[x])-1):
                data1 = []
```

```
                data2 = []
                data1.append(rawData[x][index][1])
                data1.append(rawData[x][index][2])
                data2.append(rawData[x][index+1][1])
                data2.append(rawData[x][index+1][2])
                if self.checkOverLap(data1,data2) == True:
                    i =
self.getIndexList(listTask,rawData[x][index][0])
                    j =
self.getIndexList(listTask,rawData[x][index+1][0])
                    tmpMatrixOverlap[i][j] = 1
                    tmpMatrixOverlap[j][i] = 1
                    matrix[i][j] = 0
                    matrix[j][i] = 0
            matrixCountOverlap =
np.add(tmpMatrixOverlap,matrixCountOverlap)
            for y in range(0,len(tmpMatrixOverlap)):
                for x in range(0,len(tmpMatrixOverlap[0])):
                    tmpMatrixOverlap[y][x] = 0
        return matrix,matrixCountOverlap
```

Kode Sumber 5.2 **Fungsi `Modified Time-Based HeuristicsMiner` bagian Mendapatkan Aktivitas yang *overlap***

- Yang ketiga adalah bagian *mining dependency graph* hingga mendapatkan *threshold RBT*, *POT* dan *DT*. Pengimplementasian dapat dilihat pada Kode Sumber 5.3 dan Kode Sumber 5.4

```
def getMatrixDependencyMeasure(self,matrixDependency):
    result = copy.deepcopy(matrixDependency)
    for y in range(0,len(matrixDependency)):
        for x in range(0,len(matrixDependency)):
            if matrixDependency[y][x] == 0:
                continue
            res = (float(matrixDependency[y][x])-
                   float(matrixDependency[x][y]))/
                  (float(matrixDependency[y][x])+
                   float(matrixDependency[x][y])+1)
            result[y][x] = res
    return result

def getMatrixDependecy(self, rawData):
    listTask = self.getTask(rawData)
    matrix = [[0 for x in range(len(listTask))]
for x in range(len(listTask))]
for x in rawData :
```

```
    for index in range (0,len(rawData[x])-1):
    i =self.getIndexList(listTask,rawData[x][index][0])
      j =
self.getIndexList(listTask,rawData[x][index+1][0])
       matrix[i][j]+=1
 return matrix
```

Kode Sumber 5.3 Fungsi `Modified Time-Based HeuristicsMiner` bagian *mining dependency graph*

```
def getRBTPOTDT(self,dataDependency, dataMeasureOverlap):
    dataArray = np.array(copy.copy(dataMeasureOverlap))
    resMeasure = np.where(dataArray>0.0)
    listsMeasure = []
    for x in range(0,len(resMeasure[0])):
        coor_y,coor_x = resMeasure[0][x],resMeasure[1][x]

listsMeasure.append(dataMeasureOverlap[coor_y][coor_x])
    avg = np.average(listsMeasure)
    std = np.std(listsMeasure)

    dataArrayDependency =
np.array(copy.copy(dataDependency))
    resDependency = np.where(dataArrayDependency>0.0)
    listsDependency = []
    for x in range(0,len(resDependency[0])):
        coor_y,coor_x =
resDependency[0][x],resDependency[1][x]

listsDependency.append(dataDependency[coor_y][coor_x])
    return float(avg)-(float(std)/float(2)),
     np.min(listsDependency),float(avg)-float(std)
```

Kode Sumber 5.4 Fungsi `Modified Time-Based HeuristicsMiner` bagian Mendapatkan *threshold RBT, POT* dan *DT*

- Yang keempat adalah bagian *mining short loop.* Pengimplementasian dapat dilihat pada Kode Sumber 5.5

```
def checkOneLoop(self,data1,data2):
    if data1 == data2:
        return True
    return False

def checkTwoLoop(self,data):
    if data[0] == data[2] and data[1] == data[3]:
        return True
```

```
    return False

def getLoop(self,rawData):
  listTask = self.getTask(rawData)
  matrix1Loop = [[0 for x in range(len(listTask))]
    for x in range(len(listTask))]
  matrix2Loop = [[0 for x in range(len(listTask))]
    for x in range(len(listTask))]

  tmpMatrix1Loop = [[0 for x in range(len(listTask))]
    for x in range(len(listTask))]
  tmpMatrix2Loop = [[0 for x in range(len(listTask))]
    for x in range(len(listTask))]

  for x in rawData:
      for index in range (0,len(rawData[x])-1):
         if self.checkOneLoop(rawData[x][index][0],
           rawData[x][index+1][0]) == True :
         i =
self.getIndexList(listTask,rawData[x][index][0])
         j =
self.getIndexList(listTask,rawData[x][index+1][0])
         tmpMatrix1Loop[i][j] = 1
      elif index+3 <= len(rawData[x])-1:
         data = []
         for indexes in range(index,index+4):
           data.append(rawData[x][indexes][0])
         if self.checkTwoLoop(data) == True:
         i =
self.getIndexList(listTask,rawData[x][index][0])
         j =
self.getIndexList(listTask,rawData[x][index+1][0])
         tmpMatrix2Loop[i][j]=1
         tmpMatrix2Loop[j][i]=1
      matrix1Loop = np.add(matrix1Loop,tmpMatrix1Loop)
      matrix2Loop = np.add(matrix2Loop,tmpMatrix2Loop)
      for y in range(0,len(tmpMatrix1Loop)):
         for x in range(0,len(tmpMatrix1Loop[0])):
            tmpMatrix1Loop[y][x] = 0
            tmpMatrix2Loop[y][x] = 0
 return matrix1Loop,matrix2Loop
```

Kode Sumber 5.5 Fungsi `Modified Time-Based HeuristicsMiner` bagian *mining short loop*

- Yang kelima adalah bagian *mining parallel activity*. Pengimplementasian dapat dilihat pada Kode Sumber 5.6

```
def getFirstNode(self,graph):
    lists = []
    for x in graph:
        if len(graph[x]['input']) == 0:
            lists.append(x)
    return lists

def checkVisit(self,now,list_visit, graph):
    inp = graph[now]['input']
        for x in inp:
            if list_visit[x] == False:
                return False
        return True

def findCouple(self,rawData,start,graph):
    listTask = self.getTask(rawData)
    res = ''
    queue = []
    is_visit = dict()
    for x in listTask:
        is_visit[x] = False
    is_visit[start] = True
    counter = 2
    for x in graph[start]['output']:
        queue.append(x)
    while counter!=0 and len(queue):
         cur = queue[0]
         res = cur
         queue.pop(0)
         is_visit[cur] = True
         if self.checkVisit(cur, is_visit, graph) == False:
            queue.append(cur)
            continue
         if len(graph[cur]['input']) > 1:
            counter-=len(graph[cur]['input'])
         elif len(graph[cur]['output']) > 1:
             counter+=graph[cur]['output']
             for x in graph[cur]['output']:
                 queue.append(x)
         else :
             for x in graph[cur]['output']:
                 queue.append(x)
    return res
```

Kode Sumber 5.6 Fungsi `Modified Time-Based HeuristicsMiner` bagian *mining parallel activity*

- Yang keenam adalah bagian penentuan *input*, *output*, dan relasi dari masing-masing aktivitas. Pengimplementasian dapat dilihat pada Kode Sumber 5.7

```
def getAvgPDMMinPDM(self,matrixMeasure):
   dataArray = np.array(copy.copy(matrixMeasure))
   resMeasure = np.where(dataArray>0.0)
   listsMeasure = []
   for x in range(0,len(resMeasure[0])):
     coor_y,coor_x = resMeasure[0][x],resMeasure[1][x]
    listsMeasure.append(matrixMeasure[coor_y][coor_x])
    return np.average(np.array(listsMeasure)),
           np.min(np.array(listsMeasure))

def getRelation(self,rawData,graph,matrix,
    matrixOverLap,matrixMeasure):
   listTask = self.getTask(rawData)
   avgPDM,minPDM = self.getAvgPDMMinPDM(matrixMeasure)
   for x in graph:
       graph[x]['relation'] = 'sequence'
   for x in graph:
       if len(graph[x]['output']) > 1:
         to1 =
self.getIndexList(listTask,graph[x]['output'][0])
         to2 =
self.getIndexList(listTask,graph[x]['output'][1])
         fromFrom = self.getIndexList(listTask,x)
         arrMatrix = np.array(matrix)
        PM1 = (
float(matrix[to1][to2])+float(matrix[to2][to1])+
                float(2*matrixOverLap[to1][to2]) ) /
              ( float(matrix[fromFrom][to1]) +
                float(matrix[fromFrom][to2]) +
                float(sum(arrMatrix[to1,:])-matrix[to1][to2]) +
                float(sum(arrMatrix[to2,:])-matrix[to2][to1]) +
                float(1))
       couple = self.findCouple(rawData,x,graph)

   from1 =
self.getIndexList(listTask,graph[couple]['input'][0])
   from2 =
self.getIndexList(listTask,graph[couple]['input'][1])
   toTo = self.getIndexList(listTask,couple)

       PM2 = ( float(matrix[from1][from2])+
                float(matrix[from2][from1])+
                float(2*matrixOverLap[from1][from2]) ) /
             ( float(matrix[from1][toTo]) +
```

```
                float(matrix[from2][toTo]) +
            float(sum(arrMatrix[from1,:])-matrix[from1][from2])
+
            float(sum(arrMatrix[from2,:])
             -matrix[from2][from1]) + float(1))
    res = (PM1+PM2)/2

    if res <= minPDM:
       graph[x]['relation'] = 'xor split'
       graph[couple]['relation'] = 'xor join'
    elif minPDM <= res and res <= avgPDM:
       graph[x]['relation'] = 'or split'
       graph[couple]['relation'] = 'or join'
    elif avgPDM <= res :
       graph[x]['relation'] = 'and split'
       graph[couple]['relation'] = 'and join'
 return graph
```

Kode Sumber 5.7 Fungsi `Modified Time-Based HeuristicsMiner` bagian penentuan input, output dan relasi masing-masing aktivitas

- Dan yang terakhir adalah *main program* untuk menampilkan proses model. Pengimplementasian dapat dilihat pada Kode Sumber 5.8.

```
if __name__ == '__main__':
   file = "EventLogTA-ModifiedTimeBasedHeuristicMiner.xlsx"
   excel = pr.excel()
   heuristic = pr.heuristic()

rawData = excel.readData(file)
dataDependency =
heuristic.getMatrixDependecy(copy.deepcopy(rawData))

dataMeasure =
heuristic.getMatrixDependencyMeasure
(copy.deepcopy(dataDependency))

dataOverLap, dataCountOverLap =
heuristic.getOverlap(copy.deepcopy(rawData))

dataMeasureOverlap =
heuristic.multipleList2D(copy.deepcopy(dataOverLap),
copy.deepcopy(dataMeasure))

RBT,POT,DT =
```

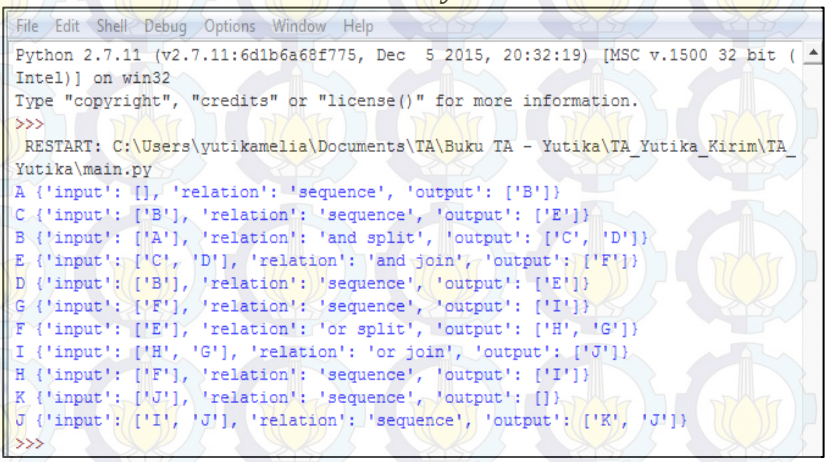
```
heuristic.getRBTPOTDT(copy.deepcopy(dataDependency),
copy.deepcopy(dataMeasureOverlap))
data1Loop,data2Loop =
heuristic.getLoop(copy.deepcopy(rawData))

graph = heuristic.getInitGraph(copy.deepcopy(rawData))
graph = heuristic.makeGraph(rawData,graph,dataMeasure,DT)
graph = heuristic.getRelation(rawData,graph,dataDependency,
dataCountOverLap,dataMeasureOverlap)
graph = heuristic.makeGraph(rawData,graph,data1Loop,1)
graph =
heuristic.makeGraphLoop(rawData,graph,data1Loop,data2Loop,P
OT)

 for x in graph:
    print x,graph[x]
```

Kode Sumber 5.8 Fungsi `Modified Time-Based HeuristicsMiner` bagian *main program*

5.3 Implementasi Antar Muka

Implementasi tampilan antarmuka pengguna pada Python. Berikut ini akan dijelaskan mengenai implementasi tampilan antarmuka pengguna yang terdapat pada perangkat lunak.

5.3.1. Halaman Proses *Discovery*



Gambar 5.1 Halaman Hasil *Process Discovery*

Halaman ini merupakan halaman yang muncul ketika aplikasi dijalankan. Setelah *file* terbuka maka sistem akan melakukan proses *discovery* dan menampilkan hasil berupa *input*, *output* dan relasi dari proses model.

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini membahas hasil dan pembahasan pada aplikasi yang dikembangkan. Pada bab ini akan dijelaskan tentang data yang digunakan, hasil yang didapatkan dari penggunaan perangkat lunak dan uji coba yang dilakukan pada perangkat lunak yang telah dikerjakan untuk menguji apakah fungsionalitas aplikasi telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya.

6.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi pembuatan sistem pada tugas akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Perangkat keras
   a. Prosesor: Intel® Core™ i5 CPU @ 2.10GHz
   b. Memori (RAM): 4 GB
   c. Tipe sistem: 32-bit sistem operasi
2. Perangkat lunak
   a. Sistem operasi: Windows 7 Professional
   b. Perangkat pengembang: Python-2.7.11

6.2 Tahapan Uji Coba

Dalam uji coba yang dilakukan dalam tugas akhir ini memiliki beberapa tahapan yang dijelaskan pada subbab ini.

6.2.1. Memasukkan Data *Event Log*

Yang pertama kali dilakukan untuk tahapan uji coba adalah memasukkan *event log* pada program. *Event log* yang digunakan harus mengandung informasi sebagai berikut:

- *Case ID*

  Case merupakan suatu kasus tertentu yang ada pada *event log*. Kasus tertentu tersebut dapat berupa suatu kasus

dalam memproduksi suatu barang tertentu, karena *event log* dapat terdiri dari catatan dari proses eksekusi pembuatan banyak barang atau proses eksekusi dari banyak kasus proses.

- Aktivitas-aktivitas yang dijalankan pada proses bisnis.
- Waktu mulai dijalankannya aktivitas dalam suatu *case* dalam proses bisnis (*start stamp*)
- Waktu selesai dijalankannya aktivitas dalam suatu *case* dalam proses bisnis (*end stamp*)

Data masukan yang digunakan dalam program ini memiliki ekstensi Excel. Contoh bentuk data masukan yang digunakan terdapat pada. Dalam perangkat lunak yang dikembangkan bagian ini dilakukan dengan memasukkan nama *file* Excel ke dalam program.

Tabel 6.1 Contoh Format Data Masukan

Case ID	Activity	Start Stamp	End Stamp
PP1	A	6/20/2014 8:32	6/20/2014 13:42
	B	6/20/2014 13:42	6/20/2014 23:41
	C	6/20/2014 23:41	6/21/2014 9:30
	D	6/21/2014 8:16	6/21/2014 10:46
	E	6/21/2014 10:46	6/21/2014 16:57
	F	6/21/2014 16:57	6/21/2014 18:09
	G	6/21/2014 18:09	6/22/2014 4:14
	H	6/21/2014 19:15	6/22/2014 10:51
	I	6/22/2014 10:51	6/22/2014 16:28
	J	6/22/2014 16:28	6/22/2014 23:42
	K	6/22/2014 23:42	6/23/2014 4:48
PP2	A	6/23/2014 4:48	6/23/2014 16:44
	B	6/23/2014 16:44	6/23/2014 23:26
	C	6/23/2014 23:26	6/24/2014 5:40
	D	6/24/2014 4:19	6/24/2014 7:48
	E	6/24/2014 7:48	6/25/2014 1:08
	F	6/25/2014 1:08	6/25/2014 3:02
	G	6/25/2014 3:02	6/25/2014 5:11
	I	6/25/2014 5:11	6/25/2014 8:25
	J	6/25/2014 8:25	6/25/2014 12:45
	K	6/25/2014 12:45	6/26/2014 0:12
PP3	A	6/26/2014 0:12	6/26/2014 4:48
	B	6/26/2014 4:48	6/26/2014 15:16
	D	6/26/2014 15:16	6/27/2014 1:52
	C	6/26/2014 22:49	6/27/2014 5:19
	E	6/27/2014 5:19	6/27/2014 11:40
	F	6/27/2014 11:40	6/27/2014 20:00
	H	6/27/2014 20:00	6/28/2014 7:10
	I	6/28/2014 7:10	6/28/2014 9:10
	J	6/28/2014 9:10	6/28/2014 16:40
	K	6/28/2014 16:40	6/28/2014 19:29

6.2.2. Melakukan Proses *Discovery*

Setelah memasukkan *event log*, proses selanjutnya yang dilakukan pada *event log* adalah melakukan proses *discovery*. Proses *discovery* dilakukan dengan algoritma *modified time-based heuristics miner* yang telah dijelaskan pada subbab 3.4. Proses *discovery* dimaksudkan untuk mengetahui hubungan antar aktivitas. Hubungan antar aktivitas diketahui setelah program dijalankan.

6.3 Data Studi Kasus

Data uji yang digunakan dalam Tugas Akhir terdapat empat jenis data uji. *Event log* 1 terdiri dari 11 aktivitas, 18 *case, start time, end time,* dan mengandung *length one loop* dan *length two loop. Event log* 2 memiliki 6 *case,* 5 aktivitas, *start time, end time, originator* dan *cost. Event log 3* memiliki 10 *case,* 10 aktivitas, *start time, end time, originator* dan *cost.* E*vent log 4* memiliki 13 *case,* 11 aktivitas, *start time, end time, originator* dan *cost. Event log 5* memiliki 10 *case,* 10 aktivitas, *start time, end time, originator* dan *cost.* Sedangkan *event log 6* memiliki 13 *case,* 11 aktivitas, *start time, end time, originator* dan *cost.*

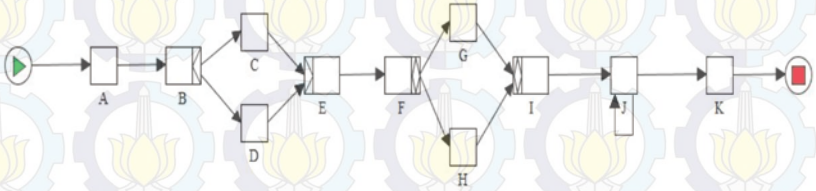
6.3.1. Data *Event Log*

Data *event log* pada Tugas Akhir ini terdiri dari aktivitas, *case* dan *trace*. *Event log* yang digunakan mengandung informasi *Case ID*, aktivitas, s*tart time, end time, cost, originator* dan *short loop*. Bentuk *event log* dari data ini diperlihatkan pada Tabel 6.2.

Tabel 6.2 Data *Event Log*

Case ID	Activity	Start Stamp	End Stamp	Cost	Resource
PP1	A	6/20/2014 8:32	6/20/2014 13:42	800	1
	B	6/20/2014 13:42	6/20/2014 23:41	600	1
	C	6/20/2014 23:41	6/21/2014 9:30	800	1
	D	6/21/2014 8:16	6/21/2014 10:46	500	1
	E	6/21/2014 10:46	6/21/2014 16:57	650	1
	F	6/21/2014 16:57	6/21/2014 18:09	700	1
	G	6/21/2014 18:09	6/22/2014 4:14	750	1
	H	6/21/2014 19:15	6/22/2014 10:51	600	1
	I	6/22/2014 10:51	6/22/2014 16:28	800	1
	J	6/22/2014 16:28	6/22/2014 23:42	600	1
	K	6/22/2014 23:42	6/23/2014 4:48	500	1
PP2	A	6/23/2014 4:48	6/23/2014 16:44	650	1
	B	6/23/2014 16:44	6/23/2014 23:26	700	1
	C	6/23/2014 23:26	6/24/2014 5:40	750	1
	D	6/24/2014 4:19	6/24/2014 7:48	600	1
	E	6/24/2014 7:48	6/25/2014 1:08	800	1
	F	6/25/2014 1:08	6/25/2014 3:02	500	1
	G	6/25/2014 3:02	6/25/2014 5:11	650	1
	I	6/25/2014 5:11	6/25/2014 8:25	700	1
	J	6/25/2014 8:25	6/25/2014 12:45	750	1
	K	6/25/2014 12:45	6/26/2014 0:12	600	1
PP3	A	6/26/2014 0:12	6/26/2014 4:48	800	1
	B	6/26/2014 4:48	6/26/2014 15:16	600	1
	D	6/26/2014 15:16	6/27/2014 1:52	600	1
	C	6/26/2014 22:49	6/27/2014 5:19	500	1
	E	6/27/2014 5:19	6/27/2014 11:40	650	1
	F	6/27/2014 11:40	6/27/2014 20:00	700	1
	H	6/27/2014 20:00	6/28/2014 7:10	750	1
	I	6/28/2014 7:10	6/28/2014 9:10	600	1
	J	6/28/2014 9:10	6/28/2014 16:40	800	1
	K	6/28/2014 16:40	6/28/2014 19:29	800	1
PP4	A	6/28/2014 19:29	6/28/2014 23:28	500	1
	B	6/28/2014 23:28	6/29/2014 9:12	650	1
	D	6/29/2014 9:12	6/29/2014 18:41	700	1
	C	6/29/2014 15:27	6/29/2014 20:50	750	1
	E	6/29/2014 20:50	6/30/2014 2:55	600	1
	F	6/30/2014 2:55	6/30/2014 9:14	800	1
	H	6/30/2014 9:14	6/30/2014 14:37	600	1
	G	6/30/2014 11:25	6/30/2014 17:33	600	1
	I	6/30/2014 17:33	7/1/2014 6:48	500	1
	J	7/1/2014 6:48	7/1/2014 9:41	750	1
	K	7/1/2014 9:41	7/1/2014 11:15	600	1
PP5	A	7/1/2014 11:15	7/1/2014 15:01	800	1
	B	7/1/2014 15:01	7/1/2014 21:44	600	1
	C	7/1/2014 21:44	7/2/2014 4:30	500	1
	E	7/2/2014 4:30	7/2/2014 14:50	650	1
	F	7/2/2014 14:50	7/3/2014 3:06	700	1
	G	7/3/2014 3:06	7/3/2014 22:01	750	1
	H	7/3/2014 14:41	7/4/2014 1:40	600	1
	I	7/4/2014 1:40	7/4/2014 9:02	800	1
	J	7/4/2014 9:02	7/4/2014 17:12	800	1
	K	7/4/2014 17:12	7/4/2014 20:56	500	1
PP6	A	7/4/2014 20:56	7/5/2014 10:21	650	1
	B	7/5/2014 10:21	7/5/2014 13:22	700	1
	C	7/5/2014 13:22	7/5/2014 17:56	750	1
	D	7/5/2014 16:26	7/5/2014 19:35	600	1
	E	7/5/2014 19:35	7/6/2014 0:37	800	1
	F	7/6/2014 0:37	7/6/2014 10:59	600	1
	G	7/6/2014 10:59	7/6/2014 21:18	600	1
	I	7/6/2014 21:18	7/7/2014 2:43	500	1
	J	7/7/2014 2:43	7/7/2014 17:13	650	1
	K	7/7/2014 17:13	7/8/2014 0:31	800	1

Tabel 6.2 menunjukan *case* dari *event log* data sebuah proses bisnis. Model dari *event log* proses bisnis diatas dapat dilihat pada Gambar 6.1. Hubungan antar aktivitas dari model data *event log* Tabel 6.2 dapat dilihat pada Tabel 6.3.



Gambar 6.1 Model Data *Event Log*

Tabel 6.3 Hubungan Antar Aktivitas dari Gambar 6.1

Input	Split / Join	Output
{Start}		A
A		B
B	AND Split	C, D
C		E
D		E
C, D	AND Join	E

Input	Split / Join	Output
E		F
F	OR Split	G, H
G		I
H		I
G, H	OR Join	I
J		J
J		K
K		{Finish}

6.4 Uji Kebenaran dan Hasil Uji Coba

Uji coba pada sistem ini mengacu pada pengujian *Blackbox* untuk menguji apakah fungsionalitas sistem telah berjalan sebagaimana mestinya. Pengujian mengacu pada setiap fitur yang telah diimplementasikan.

6.4.1. Pengujian Fungsionalitas

Pengujian fungsionalitas sistem dilakukan secara mandiri dengan menyiapkan sejumlah skenario sebagai tolak ukur keberhasilan pengujian. Pengujian fungsionalitas dilakukan dengan mengacu pada kasus penggunaan yang telah dijelaskan pada subbab 4.4. Pengujian pada kebutuhan fungsionalitas dapat dijabarkan pada subbab berikut.

6.4.1.1 Pengujian Fitur Memasukkan Event Log

Pengujian fitur memasukkan *file event log* dilakukan dengan melakukan pendefinisian nama *file* Excel *event log* pada program. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.4.

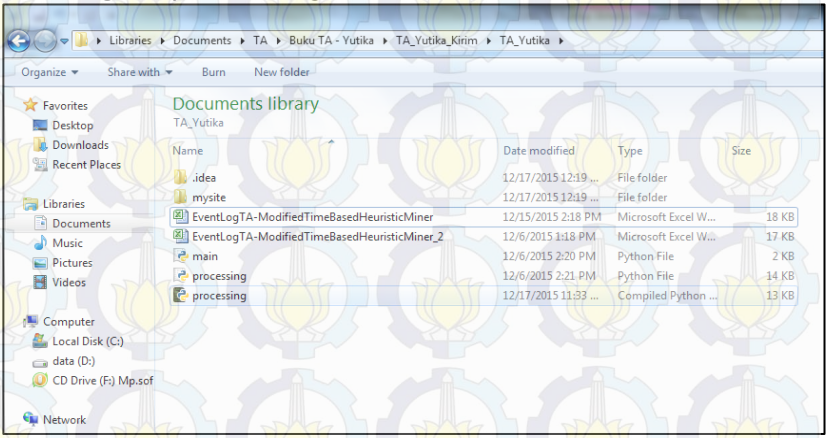
Tabel 6.4 Pengujian Fitur Memasukkan Data *Event Log*

ID	**TA-UJ.UC0001**
Referensi Kasus Penggunaan	TA-UC0001

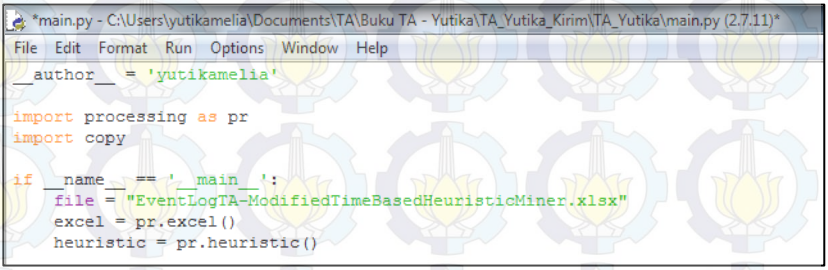
Nama	Pengujian fitur memasukkan *file event log*
Tujuan Pengujian	Menguji fitur untuk memasukkan *file* Excel *event log*
Skenario 1	Pengguna menuliskan nama *file* Excel *event log* di dalam program sesuai dengan nama *file* Excel *event log* yang tersimpan di direktori dan sistem akan mengambil *file* tersebut
Kondisi Awal	Sistem sudah dijalankan dan sistem menampilkan halaman hasil Proses *Discovery*
Data Uji	Data uji menggunakan *file* Excel *event log* sebuah proses produksi benang dan berada direktori.
Langkah Pengujian	Pengguna menuliskan nama *file* Excel *event log* yang sesuai pada halaman program
Hasil Yang Diharapkan	Sistem mampu mengenali *file* Excel *event log* dengan benar
Hasil Yang Didapat	*File* Excel *event log* yang sudah dikenali, digunakan untuk proses selanjutnya
Hasil Pengujian	100% Berhasil
Kondisi Akhir	*File event log* telah dikenali oleh sistem

Dalam uji ini digunakan *file* EventLogTA-ModifiedTimeBasedHeuristicMiner.xlsx seperti yang terlihat pada Gambar 6.2. Kemudian pengguna akan menuliskan nama *file event log* tersebut di halaman kerja program. Lalu saat sistem dijalankan, sistem berhasil menampilkan halaman hasil Proses *Discovery*. Proses tersebut dapat dilihat pada Gambar 6.3.

Terisinya *textbox* dengan *input*, *output*, dan relasi masing-masing aktivitas dapat disimpulkan bahwa sistem telah mampu mengenali dan mengolah *file event log.*



Gambar 6.2 *File* EventLogTA-ModifiedTimeBasedHeuristicMiner.xlsx pada Direktori



Gambar 6.3 Pengguna Menuliskan nama *File Event Log* di halaman kerja program

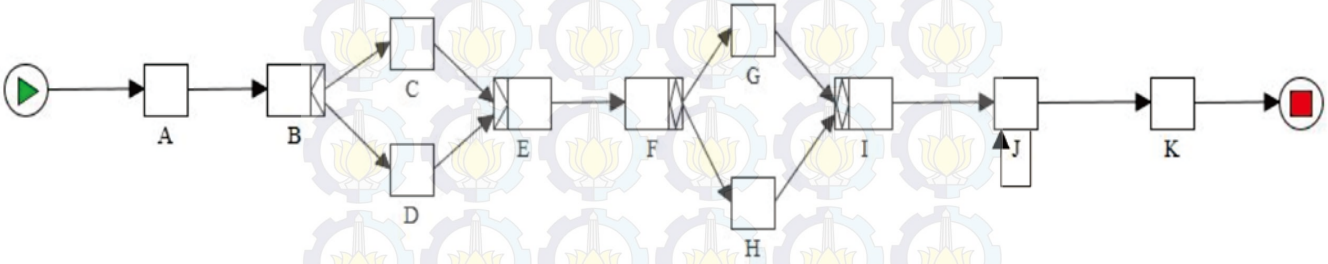
6.4.1.2 Pengujian Fitur Men-*discover* Proses Bisnis

Pengujian fitur *discover* proses bisnis dilakukan dengan melakukan pengisian atribut-atribut yang tersedia pada halaman konfigurasi. Rincian pengujian fitur ini dapat dilihat pada Tabel 6.5.

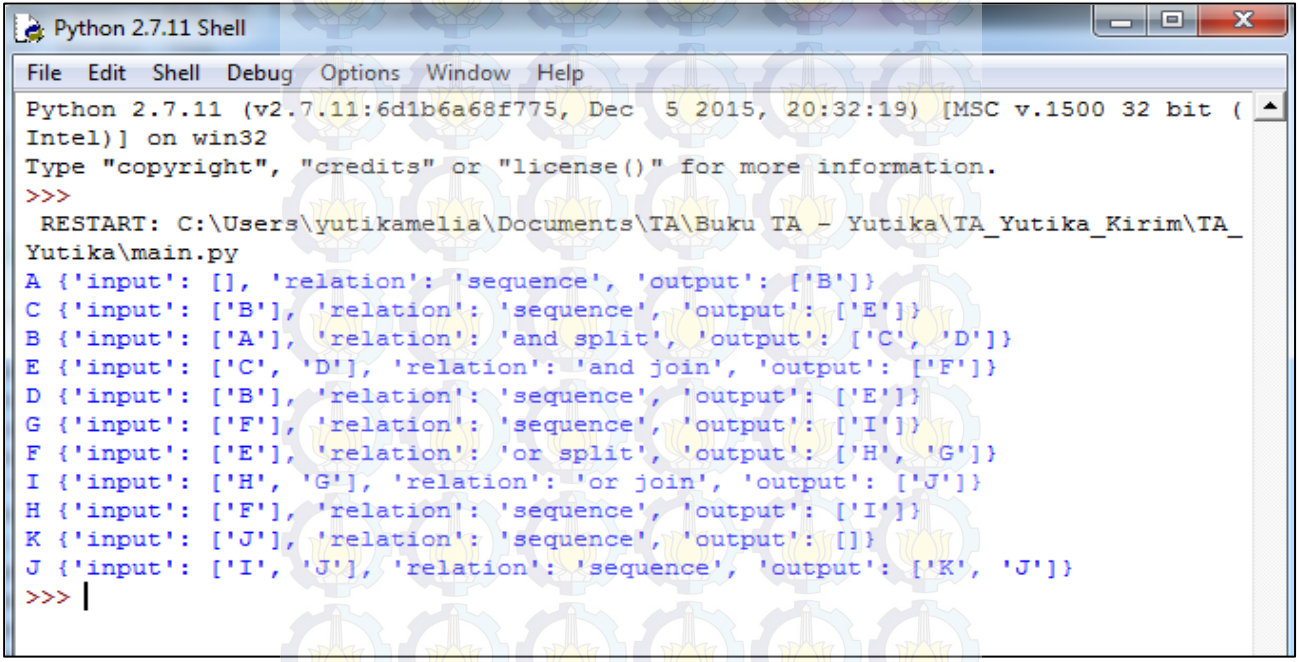
Tabel 6.5 Pengujian Fitur Konfigurasi BPMN

ID	**TA-UJ.UC0002**
Referensi Kasus Penggunaan	TA-UC0002
Nama	Pengujian fitur men-*discover* proses bisnis
Tujuan Pengujian	Menguji fitur untuk melakukan *discover* proses bisnis dari *event log* yang dimasukkan
Skenario	Pengguna menjalankan program
Kondisi Awal	Sistem sudah dijalankan dan sistem menampilkan halaman hasil proses *discovery*
Data Uji	Data uji file *event log* yang telah dimasukkan
Langkah Pengujian	Setelah memasukkan nama *file* event log dengan benar, pengguna menjalankan program untuk menampilkan halaman hasil proses *discovery*
Hasil Yang Diharapkan	Sistem mampu men-*discover event log*
Hasil Yang Didapat	*Graph* model proses bisnis dari *file* Excel *event log* berisi *input, output* dan relasi dari setiap aktivitas.
Hasil Pengujian	100% Berhasil
Kondisi Akhir	*Graph* model proses bisnis

Dalam pengujian ini digunakan *file* Exel yang sudah dituliskan pada halaman kerja program. Program dijalankan akan menampilkan halaman hasil proses discovery. Kemudian untuk *file* hasil *discovery* dapat dilihat pada Gambar 6.5 dan hasil *discovery* dalam bentuk *graph* dapat dilihat pada Gambar 6.4.



Gambar 6.4 Graf dari Hasil *Process Discovery*



Gambar 6.5 Hasil *Discovery* EventLogTA-ModifiedTimeBasedHeuristicMiner.xlsx

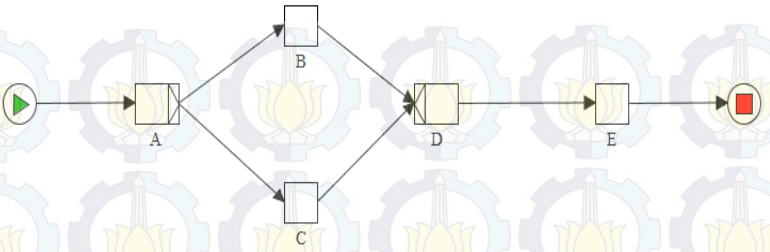
Berikut adalah tabel hubungan antar aktivitas pada Gambar 6.4 dan Gambar 6.5:

Tabel 6.6 Hubungan Antar Aktivitas dari Gambar 6.4 dan Gambar 6.5

Input	Relasi	Output
{Start}	Sequence	A
A	Sequence	B
B	AND Split	C, D
C	Sequence	E
D	Sequence	E
C, D	AND Join	E
E	Sequence	F
F	OR Split	G, H
G	Sequence	I
H	Sequence	I
G, H	OR Join	I
J	Sequence	J
J	Sequence	K
K	Sequence	{Finish}

6.4.2. Pengujian Validitas Hasil

Pada bagian ini akan dijelaskan pengujian validitas hasil sistem. Uji coba dilakukan dengan bantuan kakas YAWL dan Excel. Dalam pengujian ini yang pertama dilakukan adalah membentuk proses model dengan menggunakan YAWL, proses model dapat dilihat pada Gambar 6.6. Setelah itu dibangkitkan *event log* untuk model tersebut, *event log* yang dibangkitkan dari model YAWL kemudian diubah dalam bentuk Excel dan ditambahkan atribut pendukung yang tidak terdapat dalam *event log* hasil dari YAWL seperti *cost* tetapi banyak *trace*, aktivitas, dan *timestamp* tetap sama sesuai dengan *event log* yang dibangkitkan dari YAWL. *Event log* dari YAWL dapat dilihat pada Gambar 6.7 dan *event log* yang sudah diubah dalam bentuk Excel dapat dilihat pada Tabel 6.7.



Gambar 6.6 Model YAWL

```
<trace>
    <string key="concept:name" value="300.PO.IMP.201408.0021"/>
    <event>
        <string key="Operation" value="A"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="A"/>
        <date key="time:timestamp" value="2014-08-27T10:30:42+07:00"/>
    </event>
    <event>
        <string key="Operation" value="B"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="B"/>
        <date key="time:timestamp" value="2014-08-27T10:30:53+07:00"/>
    </event>
    <event>
        <string key="Operation" value="D"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="D"/>
        <date key="time:timestamp" value="2014-08-27T11:31:34+07:00"/>
    </event>
    <event>
        <string key="Operation" value="E"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="E"/>
        <date key="time:timestamp" value="2014-08-27T12:31:34+07:00"/>
    </event>
    <event>
        <string key="Operation" value="C"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:name" value="C"/>
        <date key="time:timestamp" value="2014-08-27T14:31:34+07:00"/>
    </event>
```

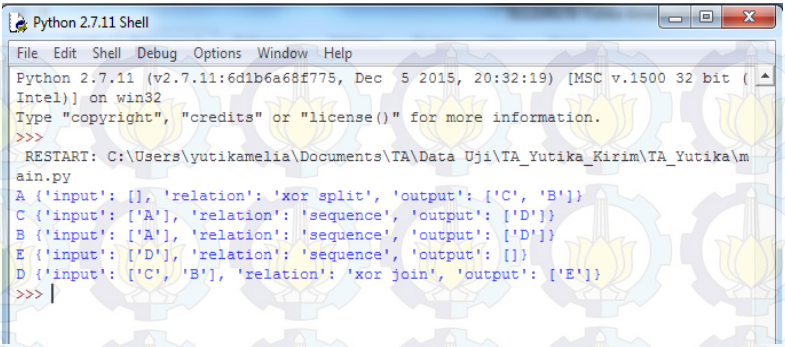
Gambar 6.7 Bentuk *Event Log* YAWL

Tabel 6.7 Bentuk *Event Log* YAWL yang Diubah ke Excel

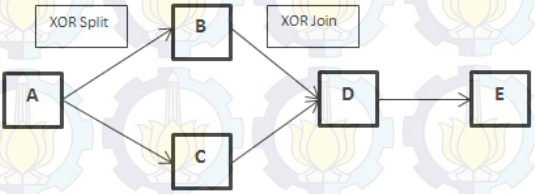
Case ID	Activity	Start Stamp	End Stamp	Originator	Cost
PP1	A	6/20/2014 8:32	6/20/2014 13:42	AN	400
	B	6/20/2014 13:42	6/21/2014 9:30	AN	1500
	C	6/21/2014 4:30	6/21/2014 9:30	AN	1000
	D	6/21/2014 9:30	6/21/2014 10:46	AN	2000
	E	6/21/2014 10:46	6/21/2014 16:57	AN	1500
PP2	A	6/21/2014 16:57	6/21/2014 18:09	AN	1000
	C	6/21/2014 18:09	6/22/2014 10:51	AN	2000
	B	6/21/2014 19:15	6/22/2014 10:51	AN	1000
	D	6/22/2014 10:51	6/22/2014 16:28	AN	400
	E	6/22/2014 16:28	6/22/2014 23:42	AN	500
PP3	A	6/22/2014 23:42	6/23/2014 4:48	AN	500
	B	6/23/2014 4:48	6/23/2014 16:44	AN	1000
	D	6/23/2014 16:44	6/23/2014 23:26	AN	1500
	E	6/23/2014 23:26	6/24/2014 5:40	AN	1500
PP4	A	6/24/2014 5:40	6/24/2014 7:48	AN	1000
	C	6/24/2014 7:48	6/25/2014 1:08	AN	1000
	D	6/25/2014 1:08	6/25/2014 3:02	AN	1500
	E	6/25/2014 3:02	6/25/2014 5:11	AN	400
PP5	A	6/25/2014 5:11	6/25/2014 8:25	AN	1500
	B	6/25/2014 8:25	6/26/2014 0:12	AN	1000
	C	6/25/2014 12:45	6/26/2014 0:12	AN	2000
	D	6/26/2014 0:12	6/26/2014 4:48	AN	1500
	E	6/26/2014 4:48	6/26/2014 15:16	AN	1000
PP6	A	6/26/2014 15:16	6/27/2014 1:52	AN	2000
	C	6/27/2014 1:52	6/27/2014 11:40	AN	1000
	B	6/27/2014 5:19	6/27/2014 11:40	AN	400
	D	6/27/2014 11:40	6/27/2014 20:00	AN	500
	E	6/27/2014 20:00	6/28/2014 7:10	AN	500

6.4.2.1 Pengujian Validitas Hasil *Discovery*

Pada bagian ini akan dijelaskan pengujian hasil *discovery event log* menjadi model proses bisnis. Pengecekan dengan membandingkan model awal yang dibuat pada YAWL pada Gambar 6.6 harus sama dengan hasil proses *discovery* pada Gambar 6.8 dan Gambar 6.7. Perbandingan dilakukan dengan membandingkan hubungan antar aktivitas dan percabangan yang digunakan. Hubungan antar aktivitas pada model YAWL dapat dilihat pada Tabel 6.8. Hubungan antar aktivitas pada model hasil proses *discovery* dapat dilihat pada Tabel 6.9.



Gambar 6.8 Hasil *Discovery* dengan Program



Gambar 6.9 Bentuk Model Hasil *Discovery*

Tabel 6.8 Hubungan Aktivitas Model YAWL Gambar 6.6

Input	Split atau Join	Output
A	XOR Split	B,C
D		E
B,C	XOR Join	D

Tabel 6.9 Hubungan Aktivitas Model Hasil *Discovery* Gambar 6.9

Input	Split atau Join	Output
A	XOR Split	B,C
D		E
B,C	XOR Join	D

Dari Tabel 6.8 dan Tabel 6.9 dapat dilihat bahwa kedua tabel tersebut sama, hal ini menandakan bahwa proses model hasil *discovery* benar.

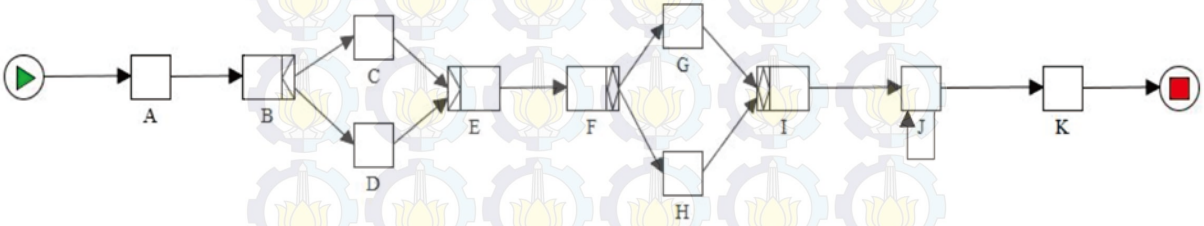
6.4.3. Hasil Uji Terhadap Data Studi Kasus

Pada bagian ini akan diperlihatkan hasil *running* sistem dengan masukan data studi kasus seperti yang ada pada subbab 6.2.

1.4.3.1 Hasil Data Studi Kasus *Event Log* 1

Data studi kasus ini telah dijelaskan pada subbab 6.3.1. Dalam subbab ini akan diperlihatkan hasil dari *running* sistem menggunakan studi kasus tersebut. Pada Gambar 6.10 menunjukkan hasil proses *discovery* dari *event log* 1. Kesamaan antara model dan hasil *discovery* juga ditunjukkan oleh Tabel 6.10. Gambar 6.11 menunjukkan hasil *discovery* dengan menggunakan sistem.

Hasil proses *discovery* pada sistem yang ditunjukkan Gambar 6.10 sama dengan proses model awal *event log* pada Gambar 6.1.

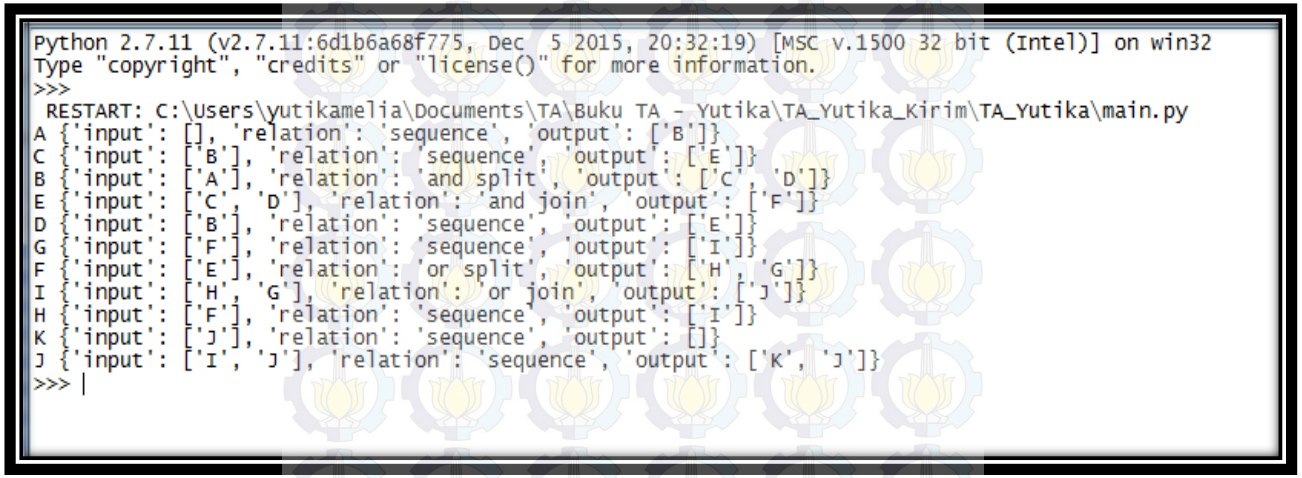


Gambar 6.10 Hasil Model *Proses Discovery Event Log* 1

Tabel 6.10 Perbandingan Hubungan Aktivitas Model dan Hasil *Discovery Event Log* 1

Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
{Start}		A	{Start}		A	Sama
A		B	A		B	Sama
B	AND Split	C, D	B	AND Split	C, D	Sama
C		E	C		E	Sama
D		E	D		E	
C, D	AND Join	E	C, D	AND Join	E	Sama

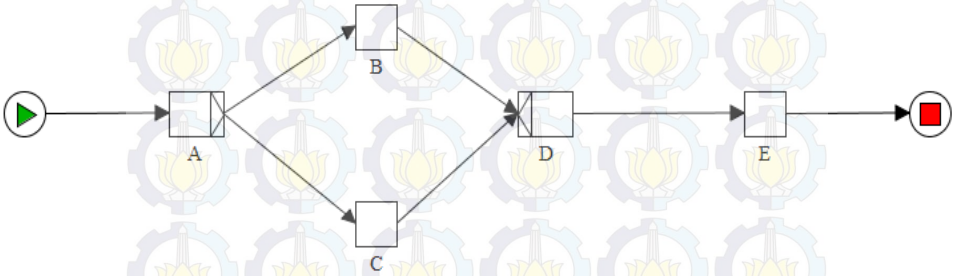
Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
E		F	E		F	Sama
F	OR Split	G, H	F	OR Split	G, H	Sama
G		I	G		I	Sama
H		I	H		I	Sama
G, H	OR Join	I	G, H	OR Join	I	Sama
I		J	I		J	Sama
J		J	J		J	Sama
J		K	J		K	Sama
K		{end}	K		{end}	Sama



Gambar 6.11 Hasil *Discovery Event Log* 1

1.4.3.2 Hasil Data Studi Kasus *Event Log* 2

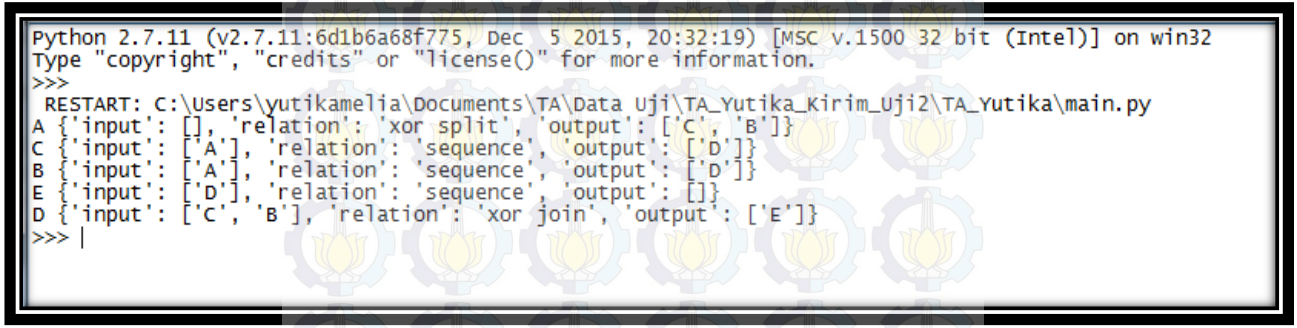
Data studi kasus ini telah dijelaskan pada subbab 6.4.2. Dalam subbab ini akan diperlihatkan hasil dari *running* sistem menggunakan studi kasus tersebut. Gambar 6.12 menunjukkan hasil proses *discovery* dari *event log* 2. Kesamaan antara model dan hasil *discovery* juga ditunjukkan oleh Tabel 6.11. Gambar 6.13 menunjukkan hasil *discovery* dengan menggunakan sistem. Hasil proses *discovery* pada sistem yang ditunjukkan Gambar 6.12 sama dengan proses model awal *event log* pada Gambar 6.6.



Gambar 6.12 Hasil Model *Proses Discovery Event Log* 2

Tabel 6.11 Perbandingan Hubungan Aktivitas Model dan Hasil *Discovery Event Log* 2

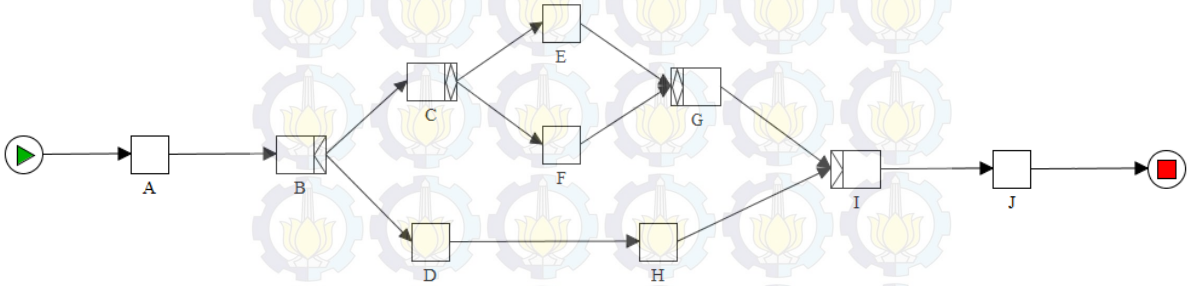
Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
{Start}		A	{Start}		A	Sama
A	XOR Split	B, C	A	XOR Split	B, C	Sama
B		D	B		D	Sama
C		D	C		D	Sama
B, C	XOR Join	D	B, C	XOR Join	D	Sama
D		E	D		E	Sama
E		{end}	E		{end}	Sama



Gambar 6.13 Hasil *Discovery Event Log* 2

1.4.3.3 Hasil Data Studi Kasus *Event Log* 3

Data studi kasus ini telah dijelaskan pada subbab 6.3.1. Dalam subbab ini akan diperlihatkan hasil dari *running* sistem menggunakan studi kasus tersebut. Pada Gambar 6.14 menunjukkan hasil proses *discovery* dari *event log* 3. Kesamaan antara model dan hasil *discovery* juga ditunjukkan oleh Tabel 6.12. Gambar 6.15 menunjukkan hasil *discovery* dengan menggunakan sistem.



Gambar 6.14 Hasil Model *Proses Discovery Event Log* 3

Tabel 6.12 Perbandingan Hubungan Aktivitas Model dan Hasil *Discovery Event Log* 3

Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
{Start}		A	{Start}		A	Sama
A		B	A		B	Sama
B	AND Split	C, D	B	AND Split	C, D	Sama
C	OR Split	E. F	C	OR Split	E. F	Sama
D		H	D		H	Sama

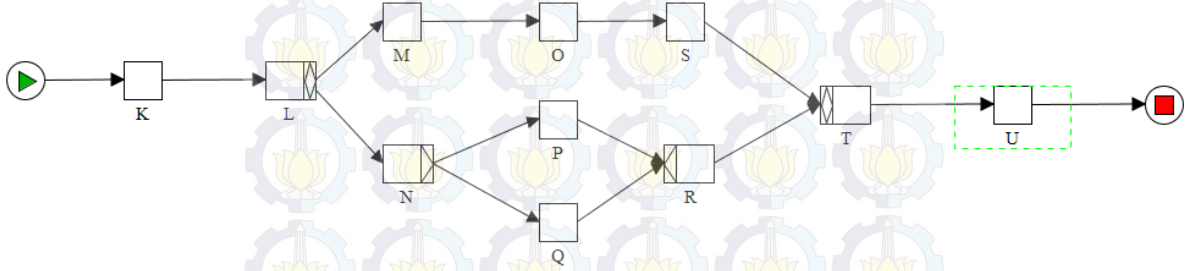
Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
E, F	OR Join	G	E, F	OR Join	G	Sama
G, H	AND Join	I	G, H	AND Join	I	Sama
I		J	I		J	Sama
J		{end}	J		{end}	Sama

```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\yutikamelia\Documents\TA\Buku TA - Yutika\TA_Yutika_Kirim\TA_Yutika\main.py
A {'input': [], 'relation': 'sequence', 'output': ['B']}
B {'input': ['A'], 'relation': 'and split', 'output': ['C', 'D']}
C {'input': ['B'], 'relation': 'or split', 'output': ['E', 'F']}
D {'input': ['B'], 'relation': 'sequence', 'output': ['H']}
E {'input': ['C'], 'relation': 'sequence', 'output': ['G']}
F {'input': ['C'], 'relation': 'sequence', 'output': ['G']}
G {'input': ['E', 'F'], 'relation': 'or join', 'output': ['I']}
H {'input': ['D'], 'relation': 'sequence', 'output': ['I']}
I {'input': ['G', 'H'], 'relation': 'and join', 'output': ['J']}
J {'input': ['I'], 'relation': 'sequence', 'output': []}
>>>
```

Gambar 6.15 Hasil *Discovery Event Log* 3

1.4.3.4 Hasil Data Studi Kasus *Event Log* 4

Data studi kasus ini telah dijelaskan pada subbab 6.3.1. Dalam subbab ini akan diperlihatkan hasil dari *running* sistem menggunakan studi kasus tersebut. Pada Gambar 6.16 menunjukkan hasil proses *discovery* dari *event log* 4. Kesamaan antara model dan hasil *discovery* juga ditunjukkan oleh Tabel 6.13. Gambar 6.17 menunjukkan hasil *discovery* dengan menggunakan sistem.

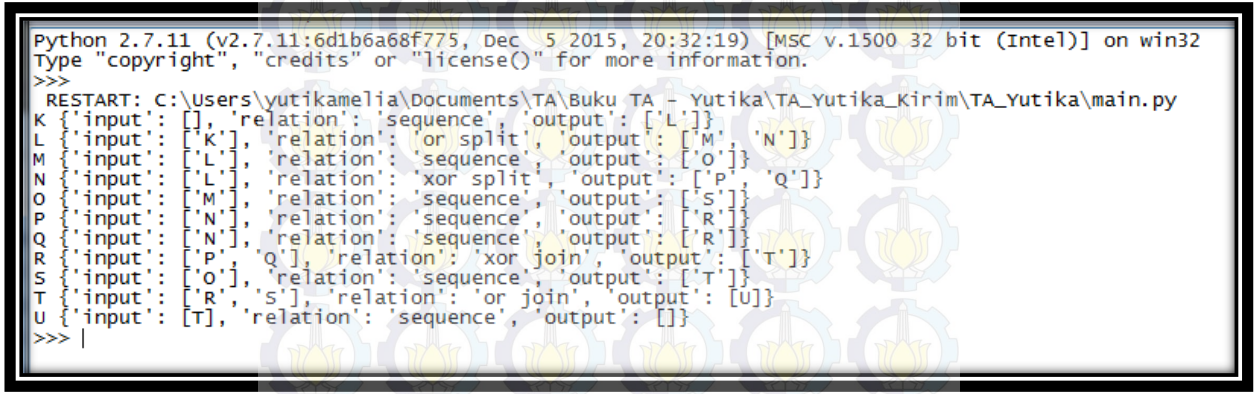


Gambar 6.16 Hasil Model *Proses Discovery Event Log* 4

Tabel 6.13 Perbandingan Hubungan Aktivitas Model dan Hasil *Discovery Event Log* 4

Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
{Start}		K	{Start}		K	Sama
K		L	K		L	Sama

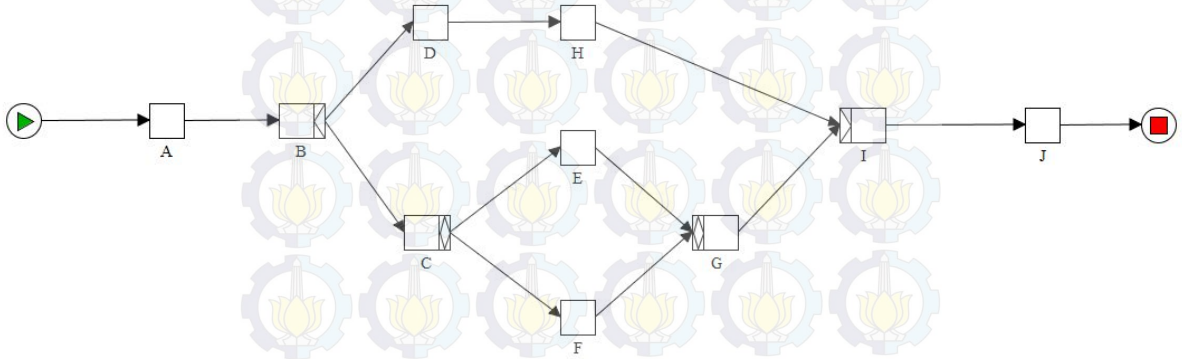
Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
L	OR Split	M, N	L	OR Split	M, N	Sama
N	XOR Split	P. Q	N	XOR Split	P. Q	Sama
M		O	M		O	Sama
P, Q	XOR Join	R	P, Q	XOR Join	R	Sama
S, R	OR Join	T	S, R	OR Join	T	Sama
T		U	T		U	Sama
U		{end}	U		{end}	Sama



Gambar 6.17 Hasil *Discovery Event Log* 4

1.4.3.5 Hasil Data Studi Kasus *Event Log* 5

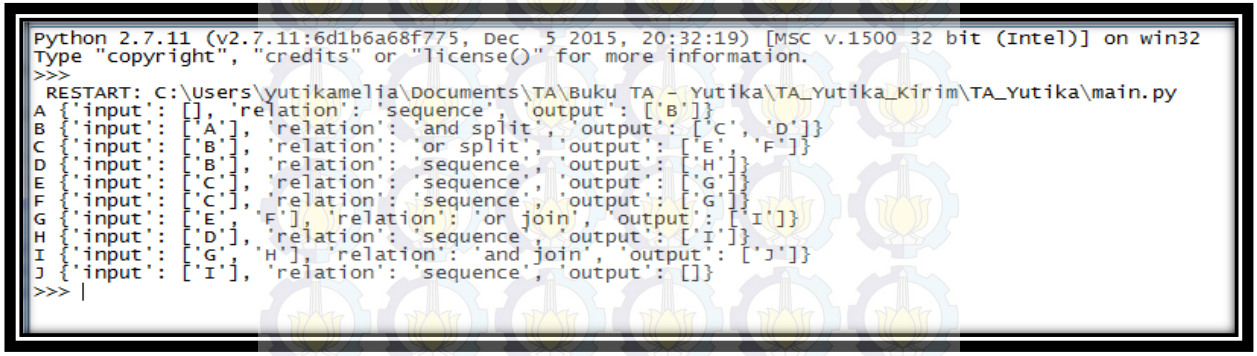
Data studi kasus ini telah dijelaskan pada subbab 6.3.1. Dalam subbab ini akan diperlihatkan hasil dari *running* sistem menggunakan studi kasus tersebut. Pada Gambar 6.18 menunjukkan hasil proses *discovery* dari *event log* 3. Kesamaan antara model dan hasil *discovery* juga ditunjukkan oleh Tabel 6.14. Gambar 6.19 menunjukkan hasil *discovery* dengan menggunakan sistem.



Gambar 6.18 Hasil Model *Proses Discovery Event Log* 5

Tabel 6.14 Perbandingan Hubungan Aktivitas Model dan Hasil *Discovery* *Event Log* 5

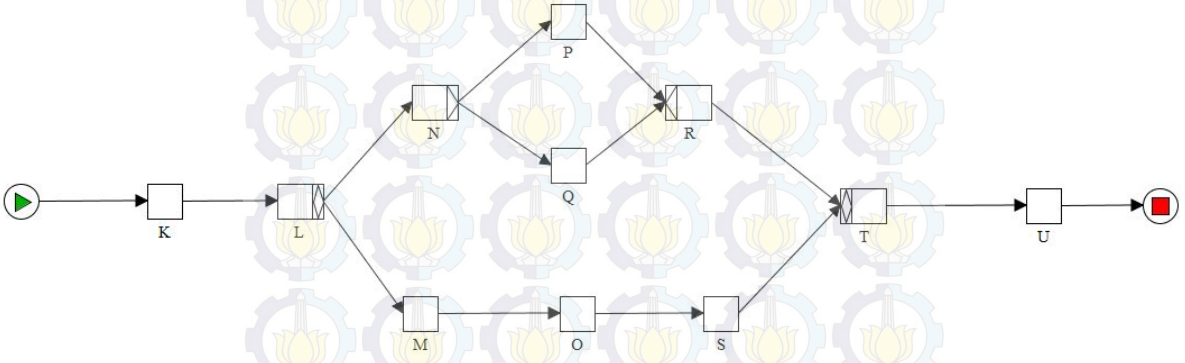
Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
{Start}		A	{Start}		A	Sama
A		B	A		B	Sama
B	AND Split	C, D	B	AND Split	C, D	Sama
C	OR Split	E. F	C	OR Split	E. F	Sama
D		H	D		H	Sama
E, F	OR Join	G	E, F	OR Join	G	Sama
G, H	AND Join	I	G, H	AND Join	I	Sama
I		J	I		J	Sama
J		{end}	J		{end}	Sama



Gambar 6.19 Hasil *Discovery Event Log* 5

1.4.3.6 Hasil Data Studi Kasus *Event Log* 6

Data studi kasus ini telah dijelaskan pada subbab 6.3.1. Dalam subbab ini akan diperlihatkan hasil dari *running* sistem menggunakan studi kasus tersebut. Pada Gambar 6.20 menunjukkan hasil proses *discovery* dari *event log* 4. Kesamaan antara model dan hasil *discovery* juga ditunjukkan oleh Tabel 6.15. Gambar 6.21 menunjukkan hasil *discovery* dengan menggunakan sistem.

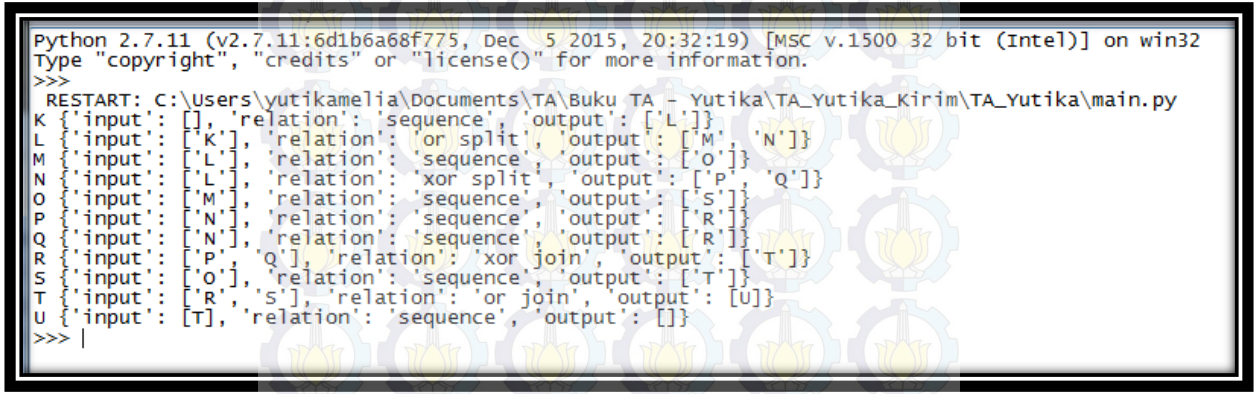


Gambar 6.20 Hasil Model *Proses Discovery Event Log* 6

Tabel 6.15 Perbandingan Hubungan Aktivitas Model dan Hasil *Discovery Event Log* 6

Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
{Start}		K	{Start}		K	Sama
K		L	K		L	Sama
L	OR Split	M, N	L	OR Split	M, N	Sama
N	XOR	P. Q	N	XOR	P. Q	Sama

Hubungan Aktivitas Model			Hubungan Aktivitas Hasil *Discovery*			Kesamaan
Input	Split / Join	Output	Input	Split / Join	Output	
	Split			Split		
M		O	M		O	Sama
P, Q	XOR Join	R	P, Q	XOR Join	R	Sama
S, R	OR Join	T	S, R	OR Join	T	Sama
T		U	T		U	Sama
U		{end}	U		{end}	Sama



Gambar 6.21 Hasil *Discovery Event Log* 6

6.5 Evaluasi Sistem dengan Sistem Lain

Evaluasi dilakukan dengan membandingkan kinerja sistem yang dikembangkan dengan sistem lain. Perbandingan dilakukan dengan menggunakan hasil yang didapatkan pada hasil uji studi kasus.

6.5.1. Evaluasi terhadap Data *Event Log* 1

Rincian evaluasi ditulis pada Tabel 6.16. Pada Tabel 6.16 merincikan evaluasi *process discovery* antarsistem. Pada evaluasi ini terdapat 2 sistem yaitu sistem A dan sistem B. Sistem A adalah sistem yang dikembangkan. Sistem B adalah sistem yang dikembangkan pada buku Tugas Akhir dengan judul "Dekomposisi Proses Bisnis untuk Optimasi Proses *Mining*".

Tabel 6.16 Evaluasi Sistem dengan Sistem Lain terhadap *Process Discovery* Proses Bisnis *Event Log* 1

Aktivitas	Sistem A		Sistem B	
	Relasi Masuk	**Relasi Keluar**	**Relasi Masuk**	**Relasi Keluar**
A	Mulai	*Sequence*	Mulai	*Sequence*
B	*Sequence*	*AND-Split*	*Sequence*	*AND-Split*
C	*AND-Split*	*AND-Join*	*AND-Split*	*AND-Join*
D	*AND-Split*	*AND-Join*	*AND-Split*	*AND-Join*
E	*AND-Join*	*Sequence*	*AND-Join*	*Sequence*
F	*Sequence*	*OR-Split*	*Sequence*	*OR-Split*
G	*OR-Split*	*OR-Join*	*OR-Split*	*OR-Join*
H	*OR-Split*	*OR-Join*	*OR-Split*	*OR-Join*
I	*OR-Join*	*Sequence*	*OR-Join*	*Sequence*
J	*Sequence*	*Loop/ Sequence*	*Sequence*	*Loop/ Sequence*
K	*Sequence*	Selesai	*Sequence*	Selesai

6.5.2. Evaluasi terhadap Data *Event Log* 2

Rincian evaluasi ditulis pada Tabel 6.17. Pada Tabel 6.17 merincikan evaluasi *process discovery* antarsistem. Sistem A adalah sistem yang dikembangkan. Sistem B adalah sistem yang

dikembangkan pada buku Tugas Akhir dengan judul "Dekomposisi Proses Bisnis untuk Optimasi Proses *Mining*".

Tabel 6.17 Evaluasi Sistem dengan Sistem Lain terhadap *Process Discovery* Proses Bisnis *Event Log* 2

Aktivitas	Sistem A		Sistem B	
	Relasi Masuk	**Relasi Keluar**	**Relasi Masuk**	**Relasi Keluar**
A	Mulai	*XOR-Split*	Mulai	*XOR-Split*
B	*XOR-Split*	*XOR-Join*	*XOR-Split*	*XOR-Join*
C	*XOR-Split*	*XOR-Join*	*XOR-Split*	*XOR-Join*
D	*XOR-Join*	*Sequence*	*XOR-Join*	*Sequence*
E	*Sequence*	Selesai	*Sequence*	Selesai

6.5.3. Evaluasi terhadap Data *Event Log* 3

Rincian evaluasi ditulis pada Tabel 6.18. Pada Tabel 6.18 merincikan evaluasi *process discovery* antarsistem. Sistem A adalah sistem yang dikembangkan. Sistem B adalah sistem yang dikembangkan pada buku Tugas Akhir dengan judul "Dekomposisi Proses Bisnis untuk Optimasi Proses *Mining*".

Tabel 6.18 Evaluasi Sistem dengan Sistem Lain terhadap *Process Discovery* Proses Bisnis *Event Log* 3

Aktivitas	Sistem A		Sistem B	
	Relasi Masuk	**Relasi Keluar**	**Relasi Masuk**	**Relasi Keluar**
A	Mulai	*Sequence*	Mulai	*Sequence*
B	*Sequence*	*AND-Split*	*Sequence*	*AND-Split*
C	*AND-Split*	*OR-Split*	*AND-Split*	*OR-Split*
D	*AND-Split*	*Sequence*	*AND-Split*	*Sequence*
E	*OR-Split*	*OR-Join*	*OR-Split*	*OR-Join*
F	*OR-Split*	*OR-Join*	*OR-Split*	*OR-Join*
G	*OR-Join*	*AND-Join*	*OR-Join*	*AND-Join*
H	*Sequence*	*AND-Join*	*Sequence*	*AND-Join*
I	*AND-Join*	*Sequence*	*AND-Join*	*Sequence*
J	*Sequence*	Selesai	*Sequence*	Selesai

6.5.4. Evaluasi terhadap Data *Event Log* 4

Rincian evaluasi ditulis pada Tabel 6.19. Pada Tabel 6.19 merincikan evaluasi *process discovery* antarsistem. Sistem A adalah sistem yang dikembangkan. Sistem B adalah sistem yang dikembangkan pada buku Tugas Akhir dengan judul "Dekomposisi Proses Bisnis untuk Optimasi Proses *Mining*".

Tabel 6.19 Evaluasi Sistem dengan Sistem Lain terhadap *Process Discovery* Proses Bisnis *Event Log* 4

Aktivitas	Sistem A		Sistem B	
	Relasi Masuk	**Relasi Keluar**	**Relasi Masuk**	**Relasi Keluar**
K	Mulai	*Sequence*	Mulai	*Sequence*
L	*Sequence*	*OR-Split*	*Sequence*	*OR-Split*
M	*OR-Split*	*Sequence*	*OR-Split*	*Sequence*
N	*OR-Split*	*XOR-Split*	*OR-Split*	*XOR-Split*
O	*Sequence*	*Sequence*	*Sequence*	*Sequence*
P	*XOR-Split*	*XOR-Join*	*XOR-Split*	*XOR-Join*
Q	*XOR-Split*	*XOR-Join*	*XOR-Split*	*XOR-Join*
R	*XOR-Join*	*OR-Join*	*XOR-Join*	*OR-Join*
S	*Sequence*	*OR-Join*	*Sequence*	*OR-Join*
T	*OR-Join*	*Sequence*	*OR-Join*	*Sequence*
U	*Sequence*	Selesai	*Sequence*	Selesai

6.5.5. Evaluasi terhadap Data *Event Log* 5

Rincian evaluasi ditulis pada Tabel 6.20. Pada Tabel 6.20 merincikan evaluasi *process discovery* antarsistem. Sistem A adalah sistem yang dikembangkan. Sistem B adalah sistem yang dikembangkan pada buku Tugas Akhir dengan judul "Dekomposisi Proses Bisnis untuk Optimasi Proses *Mining*".

Tabel 6.20 Evaluasi Sistem dengan Sistem Lain terhadap *Process Discovery* Proses Bisnis *Event Log* 5

Aktivitas	Sistem A		Sistem B	
	Relasi Masuk	**Relasi Keluar**	**Relasi Masuk**	**Relasi Keluar**
A	Mulai	*Sequence*	Mulai	*Sequence*

Aktivitas	Sistem A		Sistem B	
	Relasi Masuk	Relasi Keluar	Relasi Masuk	Relasi Keluar
B	*Sequence*	*AND-Split*	*Sequence*	*AND-Split*
C	*AND-Split*	*OR-Split*	*AND-Split*	*OR-Split*
D	*AND-Split*	*Sequence*	*AND-Split*	*Sequence*
E	*OR-Split*	*OR-Join*	*OR-Split*	*OR-Join*
F	*OR-Split*	*OR-Join*	*OR-Split*	*OR-Join*
G	*OR-Join*	*AND-Join*	*OR-Join*	*AND-Join*
H	*Sequence*	*AND-Join*	*Sequence*	*AND-Join*
I	*AND-Join*	*Sequence*	*AND-Join*	*Sequence*
J	*Sequence*	Selesai	*Sequence*	Selesai

6.5.6. Evaluasi terhadap Data *Event Log* 6

Rincian evaluasi ditulis pada Tabel 6.21. Pada Tabel 6.21 merincikan evaluasi *process discovery* antarsistem. Sistem A adalah sistem yang dikembangkan. Sistem B adalah sistem yang dikembangkan pada buku Tugas Akhir dengan judul "Dekomposisi Proses Bisnis untuk Optimasi Proses *Mining*".

Tabel 6.21 Evaluasi Sistem dengan Sistem Lain terhadap *Process Discovery* Proses Bisnis *Event Log* 6

Aktivitas	Sistem A		Sistem B	
	Relasi Masuk	Relasi Keluar	Relasi Masuk	Relasi Keluar
K	Mulai	*Sequence*	Mulai	*Sequence*
L	*Sequence*	*OR-Split*	*Sequence*	*OR-Split*
M	*OR-Split*	*Sequence*	*OR-Split*	*Sequence*
N	*OR-Split*	*XOR-Split*	*OR-Split*	*XOR-Split*
O	*Sequence*	*Sequence*	*Sequence*	*Sequence*
P	*XOR-Split*	*XOR-Join*	*XOR-Split*	*XOR-Join*
Q	*XOR-Split*	*XOR-Join*	*XOR-Split*	*XOR-Join*
R	*XOR-Join*	*OR-Join*	*XOR-Join*	*OR-Join*
S	*Sequence*	*OR-Join*	*Sequence*	*OR-Join*
T	*OR-Join*	*Sequence*	*OR-Join*	*Sequence*
U	*Sequence*	Selesai	*Sequence*	Selesai

6.6 Pengujian Validitas Hasil Perhitungan Data untuk Optimasi

Pada bagian ini akan dijelaskan pengujian hasil perhitungan data untuk optimasi. Pengecekan dilakukan dengan membandingkan perhitungan secara manual dengan menggunakan Excel. Data langkah perhitungan Excel untuk *time* dan *cost* dapat dilihat pada Tabel 6.23, Tabel 6.24, dan Tabel 6.25. Sedangkan hasil perhitungan Excel untuk *time* dan *cost* dapat dilihat pada Tabel 6.22.

Tabel 6.22 Hasil Perhitungan Excel untuk *Time* dan *Cost*

Activity	Normal Duration	Crash Duration	Normal Cost	Crash Cost	Time Slope	Cost Slope
A	7.56	3.08	656.67	760.99	4.48	23.29
B	7.05	3.25	673.33	767.61	3.80	24.84
C	9.00	4.13	610.00	714.04	4.87	21.37
D	8.32	3.37	675.00	768.15	4.95	18.82
E	10.40	4.42	650.00	749.34	5.98	16.62
F	8.28	2.50	660.00	753.76	5.77	16.24
G	9.65	3.22	750.00	837.71	6.43	13.64
H	13.85	8.84	670.00	760.23	5.01	18.00
I	8.40	3.56	680.00	784.67	4.84	21.62
J	8.47	4.68	668.33	746.36	3.79	20.58
K	5.77	2.58	642.86	742.42	3.18	31.27

Tabel 6.23 Langkah Perhitungan Excel untuk Mengubah *Time*

Case ID	Activity	Start Stamp	End Stamp	Cost	Substract Duration	Duration	Final Duration	Resource
PP1	A	6/20/2014 8:32	6/20/2014 13:42	800	5:10:00 AM	310	5.166666667	1
	B	6/20/2014 13:42	6/20/2014 23:41	600	9:59:00 AM	599.0081068	9.983468446	1
	C	6/20/2014 23:41	6/21/2014 9:30	800	9:48:09 AM	588.1561011	9.802601685	1
	D	6/21/2014 8:16	6/21/2014 10:46	500	2:30:00 AM	150.005173	2.500086217	1
	E	6/21/2014 10:46	6/21/2014 16:57	650	6:11:00 AM	371.0021242	6.183368736	1
	F	6/21/2014 16:57	6/21/2014 18:09	700	1:12:00 AM	71.99670509	1.199945085	1
	G	6/21/2014 18:09	6/22/2014 4:14	750	10:05:29 AM	605.4854814	10.09142469	1
	H	6/21/2014 19:15	6/22/2014 10:51	600	3:36:00 PM	936.0028301	15.60004717	1
	I	6/22/2014 10:51	6/22/2014 16:28	800	5:36:48 AM	336.7944341	5.613240568	1
	J	6/22/2014 16:28	6/22/2014 23:42	600	7:14:00 AM	434.0076512	7.233460854	1
	K	6/22/2014 23:42	6/23/2014 4:48	500	5:06:00 AM	306.0056043	5.100093404	1
PP2	A	6/23/2014 4:48	6/23/2014 16:44	650	11:56:24 AM	716.4044923	11.94007487	1
	B	6/23/2014 16:44	6/23/2014 23:26	700	6:41:04 AM	401.0601507	6.684335845	1
	C	6/23/2014 23:26	6/24/2014 5:40	750	6:14:32 AM	374.5347072	6.24224512	1
	D	6/24/2014 4:19	6/24/2014 7:48	600	3:29:16 AM	209.2644562	3.487740937	1
	E	6/24/2014 7:48	6/25/2014 1:08	800	5:19:50 PM	1039.82788	17.33046467	1
	F	6/25/2014 1:08	6/25/2014 3:02	500	1:53:57 AM	113.9466098	1.899110163	1
	G	6/25/2014 3:02	6/25/2014 5:11	650	2:09:14 AM	129.241171	2.154019516	1
	I	6/25/2014 5:11	6/25/2014 8:25	700	3:13:56 AM	193.9286449	3.232144082	1
	J	6/25/2014 8:25	6/25/2014 12:45	750	4:20:31 AM	260.5109711	4.341849519	1
	K	6/25/2014 12:45	6/26/2014 0:12	600	11:26:15 AM	686.2511963	11.43751994	1
PP3	A	6/26/2014 0:12	6/26/2014 4:48	800	4:36:06 AM	276.1039089	4.601731815	1
	B	6/26/2014 4:48	6/26/2014 15:16	600	10:28:37 AM	628.624645	10.47707742	1
	D	6/26/2014 15:16	6/27/2014 1:52	600	10:35:29 AM	635.4833333	10.59138889	1
	C	6/26/2014 22:49	6/27/2014 5:19	500	6:29:27 AM	389.453788	6.490896466	1
	E	6/27/2014 5:19	6/27/2014 11:40	650	6:20:47 AM	380.7888482	6.346480804	1
	F	6/27/2014 11:40	6/27/2014 20:00	700	8:20:25 AM	500.4201919	8.340336532	1
	H	6/27/2014 20:00	6/28/2014 7:10	750	11:09:54 AM	669.906446	11.16510743	1
	I	6/28/2014 7:10	6/28/2014 9:10	600	1:59:38 AM	119.625781	1.993763017	1
	J	6/28/2014 9:10	6/28/2014 16:40	800	7:30:29 AM	450.487777	7.508129617	1
	K	6/28/2014 16:40	6/28/2014 19:29	800	2:49:00 AM	168.9991653	2.816652755	1

Tabel 6.24 Langkah Perhitungan *Time*

Time	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14	Case 15	Case 16	Case 17	Case 18	Sum	Standard Deviasi
A	5.16666667	11.9400749	4.601732	3.984755	3.764118	13.40367	6.936112	9.761314	11.9164053	5.039766	2.642015	2.427662	5.32043149	4.634916	1.581654	11.1215	2.38033	6.720761	113.34	3.73
B	9.98346845	6.68433585	10.47708	9.736551	6.713569	3.029985	4.650592	6.587281	2.02254834	8.177554	11.50428	6.68588	3.83064206	3.099361	5.02	1.05	1.96289	4.46	105.68	3.16
C	9.80260169	6.24224512	6.490896	5.382012	6.77165	4.558431	9.200415	9.073077	4.72787569	12.99341	12.48061	17.87476	9.41312036	1.897688	2.21	3.88	5.038922	6.97	135.01	4.06
D	2.50008622	3.48774094	10.59139	9.472212		3.154693	2.80693	4.963998	13.8969039	15.47886	8.792279	7.550065	10.7467519	3.612275	4.346355	7.730064	3.592093	3.778729	116.50	4.08
E	6.18336874	17.33	6.35	6.081668	10.34	5.040521	0.584315	18.56124	7.0189874	7.17196	15.20603	10.08435	7.24810309	7.68	13.61942	0.466308	6.981073	10.04223	155.99	4.98
F	1.19994508	1.89911016	8.340337	6.33	12.2664	10.36	2.805175	1.867071	6.09	12.02	0.65063	10.535	14.6993974	2.10	3.35001	15.54336	6.689268	7.382517	124.13	4.81
G	10.0914247	2.15401952		6.143697	18.90277	10.30851	8.3137	6.037096		4.89	2.517627	2.586957	1.39275453		6.970492	13.9379	11.89732		106.14	5.05
H	15.6000472		11.16511	5.38353	10.98643		7.304251		6.61427421	9.781676			7.36710879	17.63	15.52499	13.46494	8.857278	8.863643	138.55	3.86
I	5.61324057	3.23214408	1.993763	13.24664	7.368658	5.424212	10.20588	1.798614	8.15107268	7.056012	7.913854	6.837844	6.51500755	13.10	7.997951	8.181078	5.218782	6.120664	125.98	3.13
J	7.23346085	4.34184952	7.50813	2.883449	8.173822	14.50329	12.17503	5.343909	3.48920264	6.671313	4.667345	7.955767	5.9991674	5.288474	12.2864	5.140123	8.130936	5.333034	127.12	3.16
K	5.1000934	11.4375199	2.816653	1.567221	3.734517	7.29016	9.130524		3.58739425	3.872948	4.222969	4.221235	1.7374413	5.025852	2.077146	6.423098	3.172335	5.319186	80.74	2.62

Tabel 6.25 Langkah Perhitungan *Cost*

Cost	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14	Case 15	Case 16	Case 17	Case 18	Sum	Standard Deviasi
A	800	650	800	500	800	650	600	600	500	750	600	700	500	800	600	650	600	600	9850.00	104.32
B	600	700	600	650	600	700	600	800	650	600	800	750	650	600	800	800	800	500	10100.00	94.28
C	800	750	500	750	500	750	500	600	600	600	500	600	700	500	500	600	600	700	9150.00	104.04
D	500	600	600	700		600	650	600	800	800	650	800	750	600	800	600	600	650	9450.00	93.15
E	650	800	650	600	650	800	700	500	600	600	700	500	600	750	650	500	500	750	9750.00	99.34
F	700	500	700	800	700	600	750	650	500	500	750	650	800	600	700	650	650	600	9900.00	93.76
G	750	650		600	750	600	600	800		800	600	700	800		600	750	800		8250.00	87.71
H	600		750	600	600		800		650	650			500	800	750	700	600	700	6700.00	90.23
I	800	700	600	500	800	500	800	600	700	600	800	750	650	600	800	600	600	750	10200.00	104.6703509
J	600	750	800	750	800	650	700	600	675	625	550	600	725	600	600	750	650	600	10025.00	78.02976221
K	500	600	800	600	500	800	750		700	650	600	700	600	700	500	775	625	700	9000.00	99.56246192

[Halaman ini segaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

7.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Dengan menggunakan proses bisnis dari salah satu departemen dapat diketahui hubungan antara satu departemen dengan departemen yang lain dengan *multi-source event log* dan teknik proses *mining.*
2. *Top level* dari proses model dapat mengandung *abstract transitions* dan *bottom level* dari proses model dapat digunakan untuk memperbaiki *abstract procedures* dari proses model *top level.*
3. Menggabungkan proses model dari *top level* dan proses model dari *bottom level* dengan proses *refinement of petri nets.*
4. Metode *top-to-down process mining based on refinement of petri nets from multi-source event log* dapat membantu men-*discover* proses bisnis yang kompleks dengan benar.
5. Modifikasi aktivitas di dalam *event log* sebagai interval waktu dapat menangkap aktivitas yang *overlap* dalam proses eksekusi realita di perusahaan.
6. Modifikasi terhadap algoritma *heuristics miner* dengan penentuan *threshold* yang otomatis dapat menghasilkan model sesuai dengan model proses bisnis yang sebenarnya.
7. Modifikasi terhadap algoritma *heuristics miner* dapat memodelkan proses bisnis yang mengandung relasi *parallel split* dan *join* OR.

8. *Modified Time-Based Heuristics Miner* memiliki hasil lebih baik dibandingkan dengan algoritma *original heuristics miner, heuristics miner for time intervals* dan *process model discovery based on activity lifespan* karena dapat mengidentifikasi relasi *conditional OR.*
9. *Modified Time-Based Heuristics Miner* memiliki kualitas *fitness* lebih baik dibandingkan dengan algoritma *original heuristics miner, heuristics miner for time intervals* dan *process model discovery based on activity lifespan.*
10. Proses model semantik valid karena dimodelkan berdasarkan pada aturan teoritis yang didefinisikan (ditunjukkan pada subbab 3.11) dan dengan menerapkan dua langkah utama untuk memodelkan *dependency graph* menjadi proses model semantik dengan benar.
11. *Completeness* dari proses model dapat mengatasi permasalahan *Length Two Loop.*
12. Data optimasi dapat dihitung dengan menggunakan rata-rata dan standar deviasi dari setiap aktivitas yang ada pada *event log* (ditunjukkan pada subbab 3.12).
13. Mendapatkan nilai durasi dan tambahan biaya yang paling optimal dengan *non-linear programming*.
14. Hasil keluaran dari perangkat lunak yang dibangun yaitu *input, output,* relasi antar aktivitas dan *short loop.*

7.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Penentuan *threshold* dapat ditentukan secara otomatis oleh sistem maupun dapat dimasukkan sendiri oleh pengguna, hal ini dilakukan untuk memberikan fasilitas pengguna yang sudah ahli dibidang penentuan model proses bisnis.
2. Penentuan rumus dan implementasi untuk percabangan paralel *split* dan *join* yang lebih dari dua.

3. Aktivitas *event log* sebagai interval waktu dapat dianalisis lebih lanjut lagi untuk menentukan batasan dari *timestamp* pada *event log* yang akan dikategorikan sebagai kegiatan yang *overlap.*
4. Penyerderhanaan pengimplementasian agar *running* sistem menjadi lebih cepat.
5. Fitur memasukkan dan menyimpan data pada sistem dapat dikembangkan sehingga dapat menyimpan dokumen selain format Excel.

DAFTAR PUSTAKA

(2015, August 20). (Wikipedia) Retrieved 6 September, 2015, from http://en.wikipedia.org/wiki/NonLinear_programming

(2015, November 28). (Working with Excel Files in Python) Retrieved 5 Desember, 2015, from http://www.python-excel.org/

(2015, November 28). (Python Package Index) Retrieved 5 Desember, 2015, from https://pypi.python.org/pypi/numpy

A.J.M.M. Weijters, J.T.S. Ribeiro. (2010). Flexible Heuristics Miner (FHM). *Beta : Research School for Operations Management and Logistics.*

A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros. (2006). Process Mining with the Heuristics Miner Algorithm. *BETA Working Paper Series, WP 166.* Eindhoven University of Technology, Eindhoven.

Aalst, W. M. (2010). Process Modelling and Analysis. Dalam *Process Mining* (hal. 31-42). Netherlands: Springer.

Aalst, W. v. (2013). *Business process management : A comprehensive survey.* ISRN Software Engineering.

Andrea Burattin, Alessandro Sperduti. (2010). Heuristics Miner for Time Intervals. *ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning*, 41-46.

Cnude, S. (2014). Improving the quality of the Heuristics Miner in ProM 6.2.

Goedertier, S., De Weerdt, J., Martens, D., Vanthienen, J., & Baesens, B. (2011). Process Discovery in Event Log: An Application in the Telecom Industry. *Applied Soft Computing*, *11*(2), 1697-1710.

J. E. Kelley and M. R. Walker. (1959). Critical path planning and scheduling. *In: Proceedings of the Eastern Joint Computational Conference, 16*, 160-172.

J. W. Zhang Jingwen, Y. Xu and Z. W. He. (2006). Discrete Time/Cost Trade-offs in Project Scheduling with Time-

switch Constraints. *Chinese Journal of Management Science, 2*, 58-64.

J. W. Zhang Jingwen, Y. Xu and Z. W. He. (2007). A Review on the Time/Cost Trade-Offs Problem in Project Scheduling. *Journal of Industrial Engineering and Engineering Management, 1*, 92-97.

Lijie Wen, W.M.P van der Aalst, Jianmin Wang, and Jiaguang Sun. (2012). Mining Process Models with Non-Free-Choice Construct, School of Software Tsinghua University,100084, Beijing,China.

M. de Leoni, F.M. Maggi, W.M. van der Aalst. (2014). *An alignment-based framework to check the conformance of declarative process models and to preprocess event log data.* Information Systems.

M. Song and W.M.P. van der Aalst. (2008). Towards Comprehensive Support for Organizational Mining. *Decision Support Systems, 46*, 300-317.

Mathias, W. (2011). *Business Process Management - Concepts, Languages, Architectures.* Springer.

Prawira, B. (2014, Agustus 12). *Pixelbali*. (Pixelbali) Retrieved April 2, 2015, from http://pixelbali.com/informasi-teknologi/critical-path-method.html

Qingtian Zeng, Hua Duan, Cong Liu. (2015). Top-to-down Process Mining from Multi-source Logs Based on Refinement of Petri Nets. 1-22.

Rakesh Agrawal, Dimitrios Gunopulos, Frank Leymann. (1998). Mining Process Models from Workflow Logs.

Richard F. Deckro, John E. Hebert, William A. Verdini, Per Henning Grimsrud, Satya Venkateshwar. (1994). Nonlinear Time/Cost Tradeoff Models in Project Management. *Elsavier Science Ltd*, 219-229.

Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Minsoo Kim. (2014). Process Model Discovery based on Activity Lifespan. *International Conference on Technology Innovation and Industrial Management,* (hal. 137-156). Seoul.

Riyanarto Sarno, B. Sanjoyo, A. Mukhlash and H.M. Astuti. (2013). Petri Net model of ERP business process

variation for Small and Medium Enterprises. *Journal of Theoretical and Applied Information Technology, 54*, 31–38.

Riyanarto Sarno, Djeni C.A., Mukhlas I, Dwi Sunaryono. (Februaty 2015). Developing a workflow management system for Enterprise Resource Planning. *Journal of Theoritical and Applied Information Technology, vol.72, No.3*, 412-421.

Riyanarto Sarno, Hari Ginardi, Endang Wahyu Pamungkas, Dwi Sunaryono. (2013). Clustering of ERP Business Process Fragments. *Proceeding IEEE International conference on computer, control, informatics, and its applications*, 319-324.

Riyanarto Sarno, Putu Linda Indita Sari, Hari Ginardi, Dwi Sunaryono, Imam Mukhlash. (2013). Decision Mining for Multi Choice Workflow Patterns. *2013 International Conference on Computer, Control, Informatics and Its Application*, (hal. 337-342).

Riyanarno Sarno, Rahadian Dustrial Dewandono, Tohari Ahmad, Mohammad Farid Naufal, dan Fernandes Sinaga. (2015). Hybrid Association Rule Learning and Process Mining for Fraud Detection. *IAENG International Journal of Computer Science*, (hal. 59-72).

S. E. Elmaghraby. (2000). On criticality and sentivity in activity networks. *European Journal of Operational Research, 127*, 220-238.

Solichul Huda, Riyanarno Sarno, Tohari Ahmad, Heru Agus Santoso. (2014). Identification of Process-based Fraud Patterns in Credit Application. *2nd International Conference on Information and Communication Technology (ICoICT)*, 84-89.

van der Aalst, W. (2011). *Process Mining - Discovery, Conformance and Enhancement of Business Process.* Netherlands: Springer.

van der Aalst, W. (2013, Oktober). *Process Mining: Beyond Business Intelligence*. Dipetik Januari 2015, 21, dari www.processmining.org

van der Aalst, W., Adriansyah, A., & van Dongen, B. (2011). Causal Nets: A Modeling Language Tailored Towards Process Discovery. *In J.P. Katoen and B. Koenig, editors, 22nd International Conference on Concurrency Theory (CONCUR 2011)*, 28-42.

van der Aalst, W., Schonenberg, M., & Song, M. (2011). Time Prediction Based on Process Mining. *Information System Sciencedirect*, 450-475.

van der Aalst (2009). Process-Aware Information Systems: Lessons to be Learned from Process Mining. *Department of Mathematics and Computer Science, Eindhoven University of Technology, p.2*.

van der Aalst, B.F. van Dongen. (2012). *Discovering petri nets from event logs, in: Transactions on Petri Nets and Other Models of Concurrency VII.* Springer.

van der Aalst., A. Adriansyah, and B.F. van Dongen. (2011). Causal Nets: A Modeling Language Tailored Towards Process Discovery. *In J.P. Katoen and B. Koenig, editors, 22nd International Conference on Concurrency Theory (CONCUR 2011)*, 28-42.

Weske, M. (2011). Business Process Management- Concepts, Languages, Architectures. *Springer*, 128-135.

Wicaksono, S., Atastina, I., & Kurniati, A. P. (2014). Evaluasi Proses Bisnis ERP dengan Menggunakan Process Mining (Studi Kasus : Goods Receipt (GR) Lotte Mart Bandung). *Informatika Telkom University*.

X. L. Wu and Z. Yin. (2007). Solving time-cost trade-off model for activity network by minimum cost flow principle. *Journal of Huazhong University of Science and Technology(Nature Science Edition), 1*, 42-45.

BIODATA PENULIS

Yutika Amelia Effendi lahir 21 tahun yang lalu di Solok, Sumatera Barat 14 April 1994 dan tumbuh besar di Riau. Anak pertama dari dua bersaudara ini menempuh pendidikan mulai dari SD YPPI Tualang (2000-2006), SMPN 1 Tualang (2006-2009), SMAN 1 Tualang (2009-2012) dan S1 Teknik Informatika ITS (2012-2016).

Penyuka warna merah ini menyenangi *travelling* dan membaca novel. *An introvert girl and easily distracted by romantic movie and internet world*. Selama kuliah, penulis pernah mengemban amanah sebagai asisten dosen pada mata kuliah Aljabar Linear, Organisasi Komputer, Kecerdasan Buatan, Analisis dan Perancangan Sistem Informasi serta Tata Kelola Teknologi Informasi. Selain itu, penulis juga aktif berorganisasi menjadi staf Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS 2014/2015 dan staf Badan Eksekutif Mahasiswa (BEM) ITS 2014/2015. Penulis dalam menyelesaikan pendidikan S1 mengambil rumpun mata kuliah (RMK) Manajemen Informasi serta memiliki ketertarikan di bidang Manajemen Basis Data, *Process Mining*, *Enterprise Resource Planning (ERP)*, Tata Kelola Teknologi Informasi, Audit Sistem serta Rekayasa Pengetahuan. Untuk komunikasi, penulis dapat dihubungi melalui surel:yutika.effendi@gmail.com.