TESIS – IF185401

SIMULASI PARALELISASI AKTIVITAS DAN OPTIMASI UNTUK PROSES BISNIS BONGKAR MUAT KAPAL PADA TPS SURABAYA

AZIZ FAJAR
NRP. 5111750010023

DOSEN PEMBIMBING :
Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.
NIP. 195908031986011001

PROGRAM MAGISTER
BIDANG KEAHLIAN MANAJEMEN INFORMASI
DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember

oleh:

AZIZ FAJAR
NRP. 5117201023

Dengan judul :
Simulasi Paralelisasi Aktivitas Dan Optimasi Untuk Proses Bisnis Bongkar Muat Kapal Pada TPS Surabaya

Tanggal Ujian : 9 Januari 2019
Periode Wisuda : Maret 2019

Disetujui oleh:

Prof. Ir. Drs. Ec. Riyanarto Sarno, M.Sc., Ph.D.
NIP. 195908031986011001 (Dosen Pembimbing I)

Dr. Ir. Raden Venantius Hari Ginardi, M.Sc.
NIP. 196505181992031003 (Penguji 1)

Dr.Eng. Darlis Herumurti, S.Kom., M.Kom.
NIP. 197712172003121001 (Penguji 2)

Hadziq Fabroyir, S.Kom., Ph.D
NIP. 1986201911089 (Penguji 3)

Dekan Fakultas Teknologi Informasi dan Komunikasi,

Dr. Agus Zainal Arifin, S.Kom., M.Kom.
NIP. 197208091995121001

i

[Halaman ini sengaja dikosongkan]

SIMULASI PARALELISASI AKTIVITAS DAN OPTIMASI UNTUK PROSES BISNIS BONGKAR MUAT KAPAL PADA TPS SURABAYA

Nama mahasiswa : Aziz Fajar

NRP : 05111750010023

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

ABSTRAK

Log data yang didapatkan dari Terminal Peti Kemas (TPS) Surabaya merupakan *Log data* yang *asynchronous*. Dengan menjalankan *Log data* ini pada simulasi berbasis *agent*, performa ketika proses bisnis tersebut dijalankan dapat diperoleh. Performa yang didapatkan dari simulasi ini adalah waktu dan biaya yang dibutuhkan untuk menjalankan masing-masing aktivitas (*task*). Setelah dilakukan simulasi tersebut, kami menemukan adanya *Asynchronous Waiting Time* (AWT). AWT adalah waktu tunggu yang diperoleh karena sebuah *agent* sibuk melakukan suatu aktivitas. Karena itu, dilakukan paralelisasi terhadap banyaknya *agent* yang bekerja dalam satu waktu. Setelah dilakukan paralelisasi, dilakukan proses optimasi untuk mengetahui jumlah *agent* parallel yang paling optimal menggunakan SMAA-2, MOORA, dan COPRAS. Hasil dari penelitian ini menujukkan bahwa MOORA dan COPRAS seimbang dengan MOORA memiliki akurasi lebih tinggi dengan 80% akurasi dengan *sensitivity coefficient* 7, sedangkan COPRAS lebih rendah akurasinya dengan 78% namun memiliki *sensitivitity coefficient* 6 sehingga lebih stabil. SMAA-2 menghasilkan akurasi 40% dan *sensitivity coefficient* 13.

Kata kunci: Simulasi, *Log data*, Simulasi Berbasis Agen, Paralelisasi, SMAA-2, MOORA, COPRAS

[Halaman ini sengaja dikosongkan]

SIMULATION OF ACTIVITY PARALLELIZATION AND OPTIMIZATION ON SHIP UNLOADING BUSINESS PROCESS OF TPS SURABAYA

Student's Name : Aziz Fajar

Student ID : 05111750010023

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

ABSTRACT

Log data gathered from Surabaya Port Container Terminal (PCT) is an asynchronous log. By simulating the log data using agent based simulation, the performance of the current system can be gathered. The performance of the current system is measured by time and cost that is required to do activities (task). After we simulate the log data, we discover Asynchronous Waiting Time (AWT). AWT occurs when an agent is still working on an activity while a new task is assigned to the agent. Therefore, we parallelize the number of agent. After the parallelizing process, we optimize the number of parallel agent to find the optimal number of parallel agent using SMAA-2, MOORA, and COPRAS. MOORA has the highest accuracy with 80% accuracy and sensitivity coefficient of 7. COPRAS score lower on accuracy with 78% but with lower sensitivity coefficient of 6. SMAA-2 has the lowest accuracy with only 40% accuracy and secured the highest sensitivity coefficient of 13.

Keywords: Simulation, Log Data, Agent Based Simulation, Parallelization, SMAA-2, MOORA, COPRAS

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Alhamdulillahirabbil'alamin. Puji dan syukur penulis panjatkan kehadirat Allah SWT atas berkat, rahmat dan hidayah-Nya, penyusunan Tesis ini dapat diselesaikan. Tesis ini dibuat sebagai salah satu syarat dalam menyelesaikan Program Studi Magister di Institut Teknologi Sepuluh Nopember Surabaya. Penulis menyadari bahwa Thesis ini dapat diselesaikan karena dukungan dari berbagai pihak, baik dalam bentuk dukungan moral dan material.

Melalui kesempatan ini dengan kerendahan hati penulis mengucapkan terima kasih dan penghargaan setinggi-tingginya kepada semua orang untuk semua bantuan yang telah diberikan, antara lain kepada:

1. Ibu dan Ayah, mbak dan adik tercinta yang tiada henti selalu mendukung anaknya dan saudaranya ini, tetap sabar mendengar keluhannya, selalu mendoakan yang terbaik dan sabar dengan segala hal yang penulis lakukan.
2. Bapak Prof. Drs. Ec. Ir. Riyanarto Sarno M.Sc., Ph.D. selaku pembimbing yang senantiasa memberikan arahan dan bimbingan kepada penulis. Semoga Allah SWT senantiasa merahmati bapak dan keluarga.
3. Seluruh dosen S2 Teknik Informatika yang telah memberikan ilmu dan pengetahuan kepada penulis selama menempuh studi.
4. Teman seperjuangan dan sahabat serta teman lainnya yang tidak dapat disebutkan satu persatu, baik di jenjang SMA, S1, dan terutama teman-teman di jenjang S2 yang selalu memberikan dukungan dalam penyelesaian Thesis ini.

Akhirnya dengan segala kerendahan hati penulis menyadari masih banyak terdapat kekurangan pada Thesis ini. Oleh karena itu, segala tegur sapa dan kritik yang sifatnya membangun sangat penulis harapkan demi kesempurnaan Tesis ini. Penulis berharap bahwa perbuatan baik dari semua orang yang dengan tulus memberikan kontribusi terhadap penyusunan Tesis ini mendapatkan pahala dari Allah. Aamiin Alluhamma Aamiin.

Surabaya, 28 Januari 2019

Aziz Fajar

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam sebuah organisasi, dalam menjalankan bisnisnya diperlukan komunikasi dengan pihak lain di luar organisasi tersebut. Untuk implementasinya, suatu sistem yang diterapkan dapat menggunakan *web service*. Masing-masing *web service* organisasi tersebut harus berkomunikasi dengan baik sehingga tujuan yang diinginkan pihak-pihak yang terlibat di dalamnya dapat tercapai. Komunikasi antar *web service* untuk mencapai tujuan yang sama inilah yang disebut sebagai koreografi (Parsa, Ebrahimifard, Amiri, & Arani, 2016). Koreografi antar organisasi pasti berbeda untuk setiap sistem, hal ini menyebabkan harus adanya pengaturan ulang koreografi setiap kali akan diterapkan dalam lingkungan yang baru. Untuk melakukan hal tersebut tentu membutuhkan waktu dan biaya serta sumber daya yang ada (Milanovi, 2010). Karena itu, dibutuhkan evaluasi performa pada organisasi yang akan menerapkan atau mengubah sistem koreografinya. Evaluasi performa ini akan memberikan gambaran yang jelas atas performa dari sistem yang akan diterapkan. Simulasi pada koreografi lintas organisasi dapat digunakan untuk mengevaluasi performa dari sistem yang ada dengan memanfaatkan event log yang dihasilkan sistem tersebut.

Simulasi banyak digunakan untuk menunjukkan bagaimana proses pada dunia nyata berjalan (Fauzan, Sarno, & Yaqin, 2017). Hal ini akan berguna sebagai masukan untuk memperbaiki kinerja dari organisasi tersebut. Pada penelitian sebelumnya, telah dilakukan simulasi untuk menunjukkan bagaimana proses bisnis impor barang berjalan pada Terminal Peti Kemas (TPS) Surabaya. Simulasi tersebut menggunakan metode *Discrete Event Simulation* (DES) dan *Agent Based Simulation* (ABS). Dalam simulasi tersebut terdapat 4 organisasi yang terlibat didalamnya. Antara lain, *customer,* PT.TPS, Bea Cukai, dan Karantina. Pada proses bisnis yang tidak melibatkan organisasi lainnya dalam setiap aktivitasnya, *Discrete Event*

Simulation secara umum sudah cukup untuk mensimulasikan proses bisnis yang berjalan karena aktivitas (*task*) didalamnya yang berlangsung secara diskrit. Penerapan *Discrete Event Simulation* menggunakan CPN Tools untuk proses bisnis dengan menjadikan event log sebagai acuannya telah berhasil dilakukan (Rozinat, Mans, Song, & van der Aalst, 2008). Namun, *Agent Based Simulation* menunjukkan hasil yang lebih mendalam terhadap proses bisnis bongkar muat kapal.

Hasil dari *Discrete Event Simulation* menunjukkan bahwa proses bisnis yang berjalan tidak memiliki masalah yang besar selain adanya aktivitas yang membutuhkan waktu eksekusi yang sangat lama. Setelah dilakukan *Discrete Event Simulation* ini, ditemukan penyebab atas lamanya waktu bongkar muat kapal (*dwelling time*) yang melebihi waktu yang ditetapkan pemerintah selama 2-3 hari. Sementara itu, hasil dari *Agent Based Simulation* tidak hanya berhasil menunjukkan aktivitas yang memiliki waktu eksekusi yang lama, namun juga menunjukkan adanya proses komunikasi yang asinkron dalam berjalannya proses bisnis organisasi-organisasi yang terlibat dalam proses bisnis impor barang di TPS Surabaya. Proses komunikasi yang asinkron adalah proses komunikasi dimana organisasi-organisasi yang terlibat dalam proses bisnis dapat berkomunikasi kapan saja walaupun organisasi yang menjadi target komunikasi masih melakukan aktivitas yang lainnya. Karena adanya proses komunikasi yang asinkron tersebut, *Agent Based Simulation* menunjukkan adanya *waiting time* yang muncul karena aktivitas yang belum selesai dieksekusi. *Asynchronous Waiting Time* (AWT) adalah *waiting time* yang terjadi pada saat sebuah *agent* tidak dapat melakukan eksekusi suatu aktivitas dikarenakan *agent* sedang mengerjakan aktivitas sebelumnya sehingga memperpanjang *waiting time* aktivitas selanjutnya.

Untuk mengurangi *Asynchronous Waiting Time* tersebut, proses paralelisasi diperlukan. Penelitian untuk paralelisasi sehingga dapat mengurangi *waiting time* dalam suatu aktivitas telah dilakukan (Hyytiä, Righter, Bilenne, & Wu, 2017). Dalam penelitian tersebut, apabila suatu aktivitas tidak dapat memenuhi *deadline* yang telah ditetapkan, maka

pengerjaan aktivitas selanjutnya akan dialihkan kepada aktivitas parallelnya, sehingga *waiting time* untuk aktivitas tersebut tidak bertambah dikarenakan aktivitas sebelumnya yang belum selesai dieksekusi. Sedangkan, proses optimasi juga dibutuhkan dalam proses paralelisasi ini karena dengan adanya proses optimasi, jumlah aktivitas yang berjalan dalam satu waktu tidak akan memakan banyak biaya namun tetap dapat mengurangi *Asynchronous Waiting Time*.

Dari latar belakang yang telah dipaparkan, peneliti mengusulkan adanya proses paralelisasi pada *agent* untuk mengurangi waiting time yang terjadi karena adanya *Asynchronous Waiting Time* tersebut. Selain itu, peneliti juga mengusulkan untuk melakukan penelitian optimasi jumlah *agent* yang akan diparalelisasi sehingga tidak memakan biaya yang terlalu banyak.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan perumusan masalah dalam penelitian ini adalah sebagai berikut.

1. Bagaimana melakukan simulasi *asynchronous* untuk multi-agen?
2. Bagaimana mengukur *Asynchronous Waiting Time?*
3. Bagaimana melakukan paralelisasi *agent* dan *task*-nya?
4. Bagaimana melakukan optimasi jumlah *agent* yang diparalelisasi?
5. Metode apakah yang paling baik untuk optimasi *agent* paralel?

1.3 Batasan Masalah

Untuk permasalahan yang akan diselesaikan, maka batasan masalah dalam penelitian ini adalah:

1. Studi kasus yang diujikan adalah proses bisnis bongkar muat kapal pada PT. TPS.
2. Multi organisasi yang terlibat antara lain: *customer*, karantina, bea cukai dan PT. TPS.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk melakukan paralelisasi dan optimasi jumlah *agent* yang dapat bekerja pada satu waktu sehingga *Asynchronous Waiting Time* yang tinggi dapat berkurang dan tidak menambah waktu tunggu dari aktivitas yang lain. Hal ini berguna sebagai saran untuk perbaikan proses bisnis yang sedang berjalan.

1.5 Manfaat Penelitian

Paralelisasi dan Optimasi *agent* untuk aktivitas yang memiliki *waiting time* tinggi dikarenakan aktivitas yang belum selesai dieksekusi pada proses bisnis bongkar muat kapal ini diharapkan dapat membantu para pemangku kebijakan (*stakeholder*) yang terlibat dalam proses bisnis untuk menentukan kebijakan strategis dan perbaikan proses bisnis di masa mendatang.

1.6 Kontribusi Penelitian

1. Mengusulkan metode untuk untuk melakukan paralelisasi *agent* sehingga *Asynchronous Waiting Time* dapat berkurang.
2. Optimasi jumlah *agent* yang optimal setelah dilakukan paralelisasi.
3. Membandingkan hasil optimasi dari beberapa metode.
4. Menemukan metode yang paling baik untuk optimasi.

BAB II
KAJIAN PUSTAKA

Pada bab ini akan diuraikan mengenai konsep dasar tentang teori yang akan digunakan dalam penelitian ini. Pemaparan tersebut meliputi penjelasan tentang *web service composition*, koreografi proses bisnis, simulasi proses bisnis, simulasi berbasis agen (*agent based simulation*), *log data*, paralelisasi *agent* dan optimasi jumlah *agent*.

2.1 Penelitian Terdahulu di Terminal Petikemas

Penelitian untuk studi kasus terminal petikemas telah banyak dilakukan sebelumnya. Penelitian yang dilakukan untuk menentukan metode evaluasi performa dari terminal petikemas juga telah dilakukan (Zouari & Khayech, 2011). Selain itu penelitian untuk melakukan efisiensi pada terminal petikemas juga telah dilakukan sehingga *bottleneck* yang terjadi pada terminal petikemas yang diteliti berhasil diatasi (Yu, Cui, & Lu, 2016). Selain itu, penelitian pada terminal petikemas juga sudah dilakukan dengan melibatkan keadaan organisasi-organisasi yang terlibat didalamnya (J. Wang, Zhu, Wang, & Huang, 2016).

Penelitian yang dilakukan oleh Zouari (Zouari & Khayech, 2011) menunjukkan bahwa terdapat beberapa pendekatan untuk menentukan performa dari terminal petikemas yang telah diteliti. Pendekatan pertama adalah dengan meneilai performa berdasarkan *Cost-Quality-Delay*. Jadi, performa dari sebuah terminal petikemas dapat dinilai berdasarkan biaya yang dibutuhkan selama berada di terminal tersebut, lalu bagaimana memberikan pelayanan yang memuaskan dan menunjukkan keinginan untuk meningkatkan produktivitas. Dan yang terakhir adalah mengurangi *delay* dari masuknya kapal dan pengelolaan muatan.

Pendekatan kedua menggunakan pendekatan enam dimensi model COFOSC (*Commercial, Operational, Financial, Organizational, Social, Citizen*). Kriteria-kriteria ini digunakan untuk memenuhi harapan pelanggan. Namun, kriteria *Commercial* dan *Operational* merupakan kriteria yang dianggap paling penting dibandingkan dengan kriteria-kriteria yang lain.

Pada penelitian yang lain, penelitian untuk mengurangi *bottleneck* dengan menggunakan paralelisasi pada *crane* yang digunakan di terminal petikemas telah dilakukan (Yu et al., 2016). Pada kasus yang diteliti, *crane* dan *lock* yang digunakan akan diubah jumlahnya, lalu dilakukan simulasi dengan *rate* masuknya kapal dan lama waktu yang sama. Variabel yang dipertimbangkan dalam penelitian ini adalah *workload* tiap *crane*, rata-rata waktu penggunaan *crane*, jumlah kontainer yang bisa dikerjakan, dan jumlah kontainer yang ditolak. Hasil dari simulasi tersebut menunjukkan bahwa paralelisasi *crane* dan *lock* dapat mengurangi waktu tunggu dari tiap proses secara signifikan.

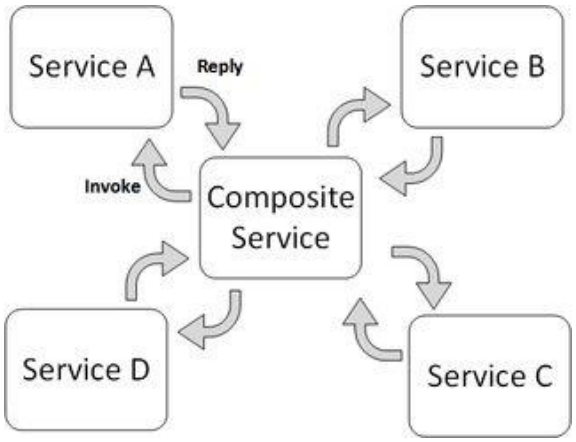
Selanjutnya, penelitian lain mencoba menunjukkan bahwa kolaborasi antara organisasi sangat penting untuk mencapai suatu tujuan (J. Wang et al., 2016). Dengan adanya kolaborasi, kualitas pelayanan dari organisasi yang terlibat. Dalam penelitian ini, kolaborasi dari organisasi yang terlibat telah dibuktikan dengan melakukan *process mining* dan *social network analysis. Process mining* dilakukan dengan menggunakan log dari *database* terminal petikemas. Selanjutnya, hasil *mining* yang berupa pola pengerjaan suatu proses didapat dengan menggunakan *heuristic miner*. Dengan diketahuinya aktivitas-aktvitas yang dilakukan dari log yang didapat, selanjuntya adalah menentukan aktor yang mengerjakan tiap aktivitas. Hasil akhir dari penelitian ini didapat dengan menggunakan *social network analysis* untuk menemukan relasi dari tiap aktor, hal ini menunjukkan pengaruh dari tiap organisasi ke studi kasus yang dikerjakan.

2.2 *Web Service Composition*

Komposisi *web service* pada dasarnya adalah membentuk suatu layanan (*service*) yang terdiri dari beberapa layanan berbeda dalam suatu aplikasi. Hal ini memungkinkan sebuah aplikasi untuk melayani *request* yang berbeda hanya dengan satu aplikasi. *Web Service Composition* dapat diimplementasikan menggunakan teknologi SOAP atau REST untuk mendeskripsikan, menemukan, dan memanggil layanan sebagai sebuah entitas. Namun, teknologi tersebut tidak menggambarkan bagaimana layanan-layanan yang ada bekerjasama. Dalam kerjasama antar layanan, layanan-layanan yang berperan akan berkolaborasi untuk melakukan kegiatan dan

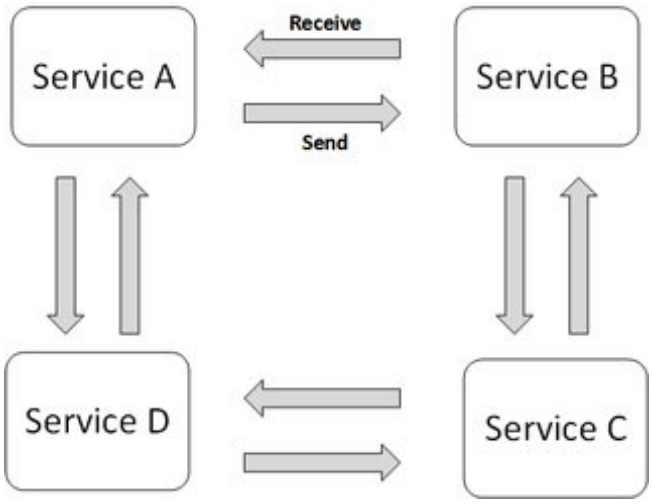
hubungan antar aktivitas untuk membentuk proses bisnis. Dalam Web Service Composition ada dua acara untuk membangunnya, yaitu orkestrasi *web service* dan koreografi *web service* (Peltz, 2003).

Dalam orkestrasi Web Service, setiap layanan yang terlibat tidak perlu mengetahui keterlibatan layanan tersebut dalam sebuah proses bisnis. Karena itu, dibutuhkan sebuah proses yang terpusat untuk mengatur layanan-layanan yang tersedia. Jadi, orkestrasi adalah sebuah sistem terpusat untuk komposisi layanan-layanan yang terlibat dalam suatu proses bisnis seperti yang ditunjukkan oleh Gambar 2.1.



Gambar 2.1 Orkestrasi *web service*

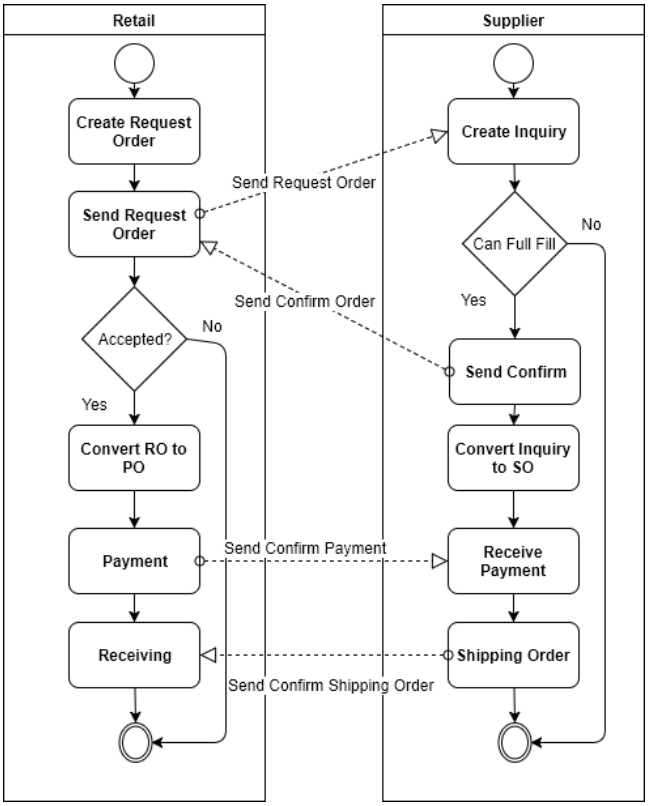
Berlawanan dengan orkestrasi, sebuah komposisi *web service* yang menggunakan konsep koreografi tidak bergantung pada koordinator sentral. Koreografi adalah upaya kolaboratif untuk pertukaran *message* dimana setiap layanan yang terlibat tahu dengan siapa dan kapan mereka harus mengirimkan message yang dibutuhkan. Pada Gambar 2.2 setiap *web service* yang terlibat dalam suatu proses bisnis dapat berinteraksi langsung dengan *web service* yang membutuhkan message tanpa harus melalui sistem yan terpusat seperti pada orkestrasi *web service*.



Gambar 2.2 Koreografi *web service*

2.3 Koreografi Proses Bisnis

Pada proses bisnis yang melibatkan multi organisasi, dibutuhkan pola komunikasi yang baik sebagai sarana berinteraksi agar tujuan bersama organisasi-organisasi yang terlibat dapat tercapai dengan baik. Interaksi inilah yang disebut sebagai koreografi proses bisnis. Sebuah koreografi adalah suatu kondisi dimana *agent* yang independen berinteraksi satu sama lain dengan bertukar pesan untuk mencapai tujuan bersama (El Kholy, Bentahar, El Menshawy, Qu, & Dssouli, 2014).



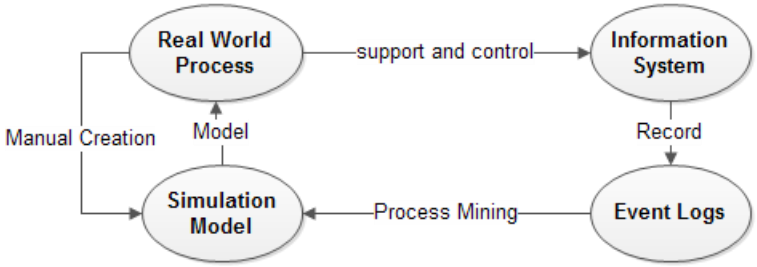
Gambar 2.3 Koreografi dua agen (retail dan supplier)

Gambar 2.3 menunjukkan conoh koreografi oleh 2 organisasi. Dalam menjalankan proses bisnisnya, ketika mencapai suatu titik kedua organisasi saling mengirimkan pesan agar proses bisnis pada masing-masing organisasi dapat berjalan dengan baik. Pada umumnya, koreografi diterapkan pada sebuah aplikasi berbasis *web service* oleh organisasi-organisasi di dalamnya. Dengan cara ini, organisasi-organisasi ini memiliki sistem yang saling terkait sehingga masing-masing dapat mendapatkan tujuan yang diinginkan berasama.

2.4 Simulasi Proses Bisnis

Simulasi adalah sebuah proses untuk meniru eksekusi proses atau sistem yang terjadi di dunia nyata (Fauzan et al., 2017). Dengan simulasi, perubahan yang akan diterapkan pada sistem yang sedang berjalan dapat diketahui bagaimana peribahan tersebut akan memberikan dampak pada proses dimana perubahan tersebut akan diimplementasikan. Koreografi *web service* yang diimplementasikan pada simulasi

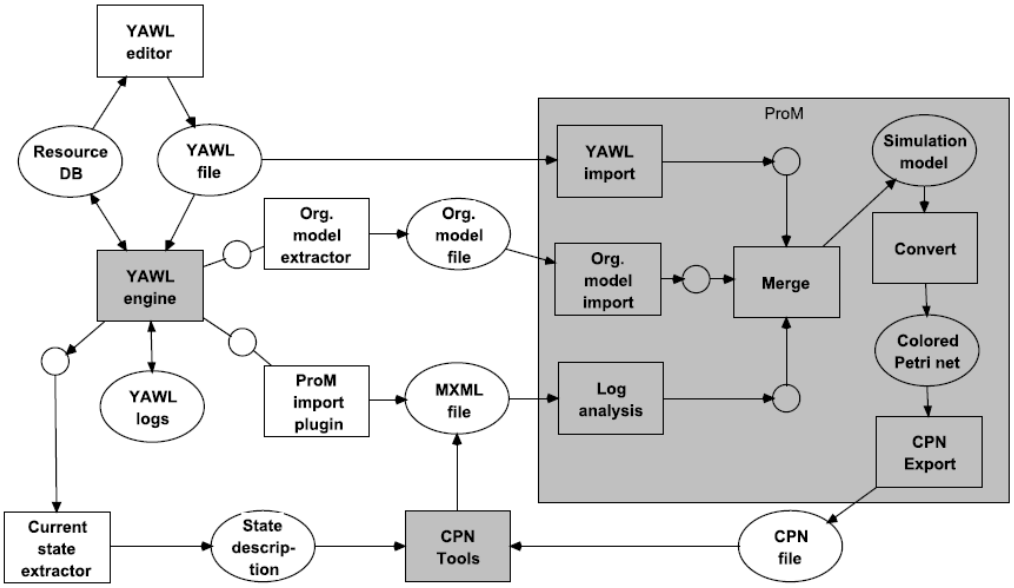
akan memberikan gambaran jelas tentang bagaimana performa dari koreografi tersebut ketika diterapkan pada sistem yang sedang berjalan.



Gambar 2.4 Alur pembuatan simulasi

Dari Gambar 2.4 telah memperlihatkan bahwa simulasi dibentuk dari proses yang berjalan di dunia nyata. Proses yang berjalan di dunia nyata akan dikumpulkan oleh sistem yang dapat mendukung manajemen dan operasi dari organisasi yang menjalankannya, sistem seperti ini biasa dikenal dengan sistem informasi. Sistem informasi akan secara otomatis merekam aktivitas yang terjadi kedalam database. Dari database inilah kita dapat memperoleh *log data* dari aktivitas yang berjalan. Dengan teknik *process mining*, model proses dapat dibentuk dari *log data* yang tersedia.

Pada penelitian sebelumnya, simulasi proses bisnis telah dibuat menggunakan CPN Tools. CPN Tools digunakan untuk membuat beberapa kondisi pada simulasi sehingga hasil simulasi dapat memperkirakan bagaimana proses bisnis akan berjalan bila diterapkan pada dunia nyata (Rozinat et al., 2008). Gambar 2.5 menunjukkan bagaimana pembuatan model simulasi dengan hasil dari *process mining*.



Gambar 2.5 Prosedur pembuatan simulasi (Rozinat et al., 2008)

Pada Gambar 2.5 ditunjukkan bagaimana penelitian sebelumnya membuat model proses. Pada awalnya, model proses yang telah dibuat oleh peneliti menggunakan YAWL akan di-*discover* menggunakan metode *process mining* dengan bantuan ProM *framework.* Hasil dari ProM adalah sebuah file dalam bentuk *Colored Petri Net* (CPN) yang dapat dieksekusi oleh CPN Tools. Dari hasil simulasi CPN Tools yang berupa file .mxml dapat dijalankan pada ProM kembali untuk mengetahui bagaimana performa simulasi yang dibuat.

2.5 Simulasi Berbasis Agen

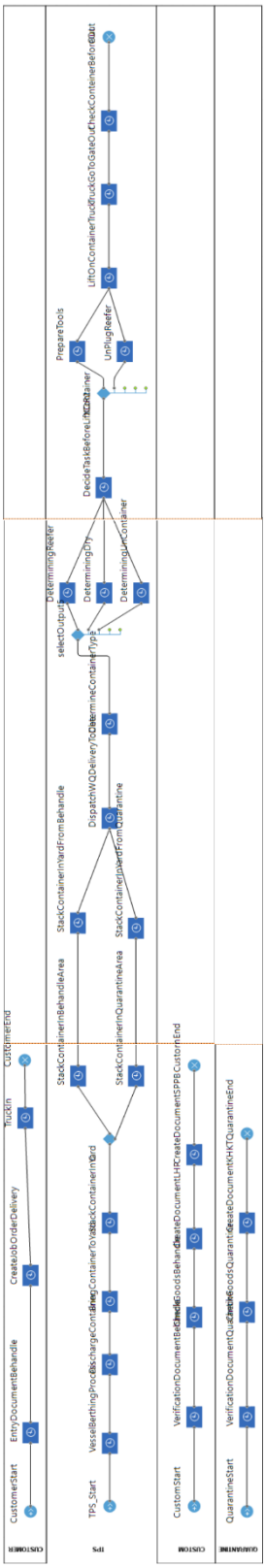
Simulasi berbasis agen atau *Agent Based Simulation* (ABS) merupakan sebuah metode simulasi yang dimana setiap pelaku aktivitas didalamnya dianggap sebagai sebuah *agent*. Sebuah *agent* memiliki 4 karakteristik dasar(Büth, Broderius, Herrmann, & Thiede, 2017) diantaranya adalah:

- Otonomi: dengan sifat otonomi ini, sebuah *agent* dapat bertindak sendiri tanpa harus adanya interferensi dari user yang lain.
- Proaktif : *agent* yang bekerja dalam *agent based simulation* dapat mengubah lingkungan agar sesuai dengan kebutuhan *agent* tersebut sehingga tujuan yang diinginkan dapat tercapai.

- Reaktif : reaktif disini adalah kemampuan *agent* untuk merespon pada perubahan lingkungan.
- Kemampuan sosial: *agent* yang bekerja pada *agent based simulation* dapat berinteraksi dengan *agent* lain yang ada untuk menyelesaikan tugas yang sedang dijalankannya.

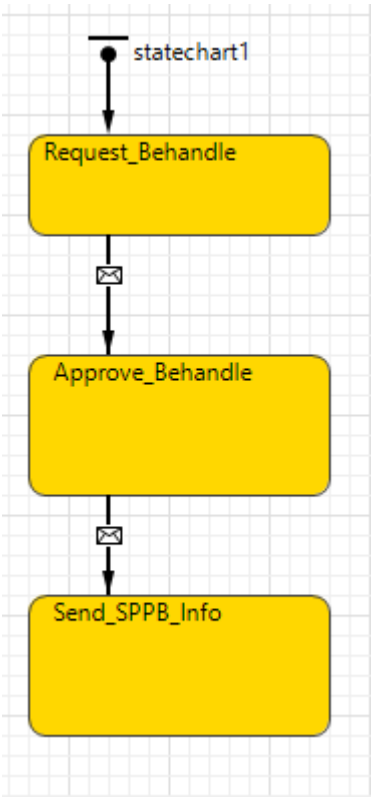
Agent Based simulation telah digunakan untuk membuat model simulasi dari berbagai macam kasus. Beberapa diantaranya adalah simulasi untuk menganalisis lalu lintas penerbangan (C. Wang, Ge, & Xu, 2009), simulasi untuk analisis proses pembelajaran terhadap sebuah kelompok belajar (Ivashkin & Nazoikin, 2009), serta simulasi untuk melihat performa sebuah terminal petikemas dan perkiraan performanya di masa depan (Fajar, Sarno, & Fauzan, 2018).

Pada penelitian sebelumnya, *agent based simulation* untuk proses bisnis impor barang pada PT. TPS telah dilakukan. Model simulasi pada penelitian tersebut dibuat berdasarkan survei langsung di lapangan, sedangkan *log data* yang didapatkan dari PT. TPS digunakan untuk menjalankan proses simulasinya. Gambar 2.6 menunjukkan model simulasi dari penelitian yang telah dilakukan.



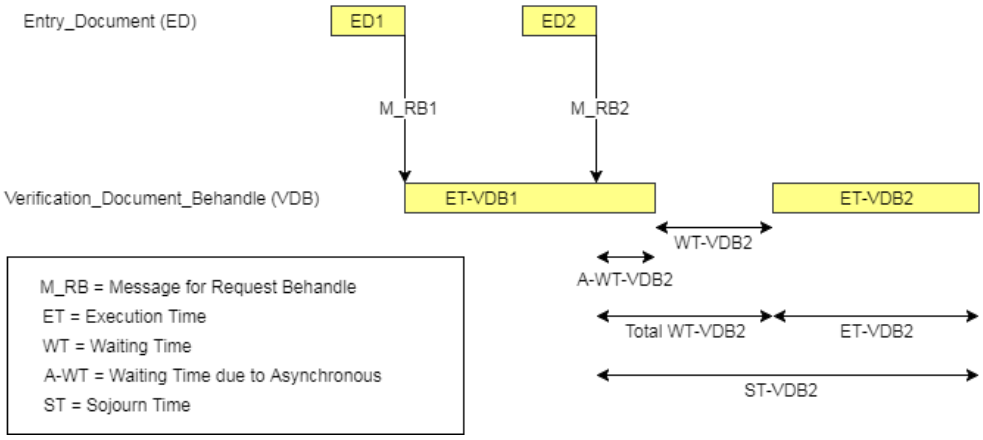
Gambar 2.6 Model simulasi pada *AnyLogic 8.0 Personal Learning Edition*

Sedangkan untuk proses komunikasi, message pada *Anylogic* menggunakan *statechart* untuk memodelkannya. *Statechart* inilah yang akan dihubungkan dengan model simulasi seperti Gambar 2.6. *Statechart* yang telah dibuat seperti pada Gambar 2.7.



Gambar 2.7 Implementasi message pada Anylogic

Hasil dari simulasi dengan menggunakan model diatas tersebut menunjukkan adanya *waiting time* yang disebabkan adanya proses yang *asynchronous* seperti ditunjukkan pada Gambar 2.8.



Gambar 2.8 Hasil *agent based simulation*

Pada Gambar 2.8 hasil dari simulasi menunjukkan adanya *Asynchronous Waiting Time* (AWT). *Asynchronous Waiting Time* adalah waktu tunggu yang muncul karena sebuah aktivitas mendapatkan waktu tunggu lebih karena *agent* yang mengerjakan aktivitas tersebut belum selesai melakukan eksekusi pada suatu aktivitas.

2.6 *Log Data*

Pada *process mining*, untuk menganalisis performa dari sebuah sistem yang berjalan, *log data* dari proses bisinis yang sedang berjalan dijadikan sebagai bahan utama sebagai acuan. *Log data* merupakan kumpulan rangkaian aktivitas yang telah berjalan dalam suatu sistem. *Log data* merekam aktivitas apa saja yang berjalan dalam proses bisnis yang berjalan. Umumnya, *log data* terdiri dari *case id*, *activity id*, *timestamp*, dan *originator*. Dalam sebuah *log data*, timestamp yang tersedia dapat berupa *single timestamp* atau *double timestamp*. *Log data* dengan *single timestamp* hanya menyediakan waktu sebuah aktivitas mulai berjalan (*start time*) atau waktu sebuah aktivitas selesai dieksekusi (*end time*). Sedangkan, *originator* yang ada dalam *log data* adalah entitas yang menjalankan aktivitas yang ada dalam *log data*. Gambar 2.9 menunjukkan sebuah contoh dari *log data*.

Case ID	Activity	Start Stamp	End Stamp
PP1	A	8/20/2016 10:32	8/20/2016 13:42
PP1	B	8/20/2016 13:42	8/20/2016 16:52
PP1	D	8/20/2016 18:27	8/20/2016 21:37
PP1	E	8/20/2016 21:37	8/21/2016 0:47
PP1	G	8/21/2016 0:47	8/21/2016 3:57
PP1	H	8/21/2016 3:57	8/21/2016 7:07
PP1	I	8/21/2016 7:07	8/21/2016 10:17
PP1	K	8/21/2016 10:17	8/21/2016 13:27
PP2	A	8/21/2016 13:27	8/21/2016 16:37
PP2	B	8/21/2016 16:37	8/21/2016 19:47
PP2	D	8/21/2016 21:22	8/22/2016 0:32
PP2	E	8/22/2016 0:32	8/22/2016 3:42
PP2	G	8/22/2016 3:42	8/22/2016 6:52
PP2	H	8/22/2016 6:52	8/22/2016 10:02
PP2	I	8/22/2016 10:02	8/22/2016 13:12
PP2	K	8/22/2016 13:12	8/22/2016 16:22

Gambar 2.9 Contoh *log data*

Gambar 2.9 merupakan contoh *log data* dengan *double timestamp* yang dimana *log data* tersebut memiliki *start time* dan *end time*. Dalam *log data* tersebut terdapat dua *case id* yaitu PP1 dan PP2. Masing-masing *case id* menunjukkan satu kasus pada suatu *log data*, sedangkan setiap kasus dapat memiliki jejak yang berbeda untuk setiap kasus.

2.6.1 Kasus dan Jejak

Kasus merupakan rangkaian aktivitas pada untuk melakukan sesuatu. Contohnya, dalam sebuah perusahaan roti, sebuah kasus merupakan proses pembuatan sebuah roti. Hal ini dapat berarti bahwa pembuatan masing-masing roti merupakan satu kasus dalam sebuah *log data*. Jejak merupakan alur aktivitas dalam sebuah kasus. Dalam hal ini, bila sebuah perusahaan roti memiliki 5 jenis roti, jejak yang yang ada dalam proses bisnisnya ada 5 jenis jejak. Misalkan dalam suatu *log data* (E):

$$E = [< a, c, d, e >^{40}, < a, b, c, d >^{32}, < b, c, e, f >^{35}, < a, c, e, f >^{20}]$$

Dalam log tersebut dapat diuraikan bahwa:

1. Terdapat 4 jejak yaitu (a, c, d, e), (a, b, c, d), (b, c, e, f), (a, c, e, f).
2. Terdapat 127 kasus karena terdapat jejak (a, c, d, e) yang dilakukan sebanyak 40 kali, jejak (a, b, c, d) sebanyak 32 kali, jejak (b, c, e, f) sebanyak 35 kali, dan jejak (a, c, e, f) sebanyak 20 kali.

2.6.2 Aktivitas

Aktivitas merupakan bagian dari kasus. Serangkaian aktivitas akan membentuk suatu kasus, dan rangkaian aktivitas yang berbeda akan membentuk jejak yang berbeda. Misal pada *log data* (*E*):

$$E = [< a, c, d, e >^{40}, < a, b, c, d >^{32}, < b, c, e, f >^{35}, < a, c, e, f >^{20}]$$

Dapat disimpulkan bahwa terdapat 5 aktivitas dalam *log data* tersebut yang terdiri dari aktivitas {a, b, c, d, e, f}.

2.6.3 Log Data TPS Surabaya

Dari TPS Surabaya, data yang didapatkan berupa event log dari database sistem yang ada. *Log data* ini dappat menggambarkan secara umum bagaimana situasi di TPS Surabaya tersebut. *Log data* yang didapatkan merupakan *log data* dari bulan Januari 2016 sampai Maret 2016. Namun, sebagian dari *log data* tersebut memiliki *dwelling time* untuk setiap container melebihi peraturan yang ditetapkan pemerintah yaitu maksimal 3 hari[1]. Log Data yang digunakan memiliki atribut sebagai berikut[2]:

a. Nomor kontainer (*Container ID*)
b. Tipe kontainer
c. Waktu mulai pengerjaan aktivitas
d. Durasi masing-masing aktivitas
e. Biaya yang dibutuhkan untuk aktivitas
f. Aktor dari aktivitas

Data bulan Januari 2016 terdiri dari 24.440 *records*, Februari 2016 terdiri dari 23.999 *records*, dan Maret 2016 terdiri dari 24.293 *records*. Dari *log data* tersebut, terdapat 12 trace yang menggambarkan alur untuk tipe kontainer yang datang ke TPS Surabaya. *Traces* yang didapatkan dari *log data* tersebut adalah sebagai berikut:

a. Quarantine; Dry; Green Line
b. Quarantine; Dry; Red Line

[1] UU No.17 tahun 2008
[2] Log data didapatkan dari PT.TPS Surabaya

c. Quarantine; Reefer; Green Line
d. Quarantine; Reefer; Red Line
e. Quarantine; Uncontainer; Green Line
f. Quarantine; Uncontainer; Red Line
g. Dry; Green Line
h. Dry; Red Line
i. Reefer; Green Line
j. Reefer; Red Line
k. Uncontainer; Green Line
l. Uncontainer; Red Line

Masing-masing *records* dalam *log data* juga akan dikerjakan dengan kerjasama dari aktor yang berbeda-beda. Sehingga, komunikasi yang terjadi antara aktor yang merupakan organisasi yang berbeda merupakan hal yang sangat vital. Beberapa organisasi yang terlibat adalah:

a. *Customer*
b. PT. Terminal Petikemas Surabaya
c. Karantina
d. Bea Cukai

Selain *log data*, dari observasi dilapangan didapatkan pula *Standard Operational Procedure* (SOP) yang akan digunakan untuk membentuk model simulasi pada bab selanjutnya. Dari SOP ini akan diketahui alur bongkar muat barang dari PT. TPS.

2.7 Paralelisasi Agen

Pada sistem yang berjalan, terkadang terjadi adanya waktu tunggu yang disebabkan oleh kurangnya jumlah aktivitas yang dapat dikejakan dalam satu waktu untuk mengeksekusi suatu aktivitas. Hal ini bisa dikarenakan banyaknya jumlah *request* yang masuk namun kemampuan *agent* untuk mengeksekusi suatu aktivitas tidak dapat mengimbangi banyaknya *request* yang datang. Paralelisasi dibutuhkan untuk mengimbangi semakin banyaknya *request* yang datang (Hyytiä et al., 2017).

Untuk mengatasi hal ini, maka paralelisasi menjadi hal yang dibutuhkan sehingga *request* yang datang dapat dikerjakan tanpa harus menunggu proses sebelumnya selesai. Dalam penelitian ini, terdapat 4 *agent* yang akan mengerjakan tiap kasus. Masing-masing *agent* tersebut akan diparalelisasi sehingga ada beberapa *agent* yang dapat bekerja dalam satu waktu.

2.8 Optimasi Jumlah Agen

Optimasi merupakan proses untuk mencari kondisi yang paling optimal dari banyaknya kemungkinan kondisi yang mungkin terjadi. Untuk paralelisasi *agent* dalam kasus ini, jumlah a*gent* yang dibutuhkan harus dapat mengeksekusi aktivitas sehingga dapat meminimalisir waktu yang dibutuhkan untuk menyelesaikan suatu kontainer namun juga harus mempertimbangkan biaya yang dibutuhkan. Dalam menentukan jumlah *agent* tentu bila terlalu banyak akan memakan biaya yang terlalu besar dan ketika *agent* tersebut tidak digunakan maka *agent* yang tidak digunakan tersebut akan sia-sia.

Karena dalam penentuan jumlah *agent* diperlukan pertimbangan atas beberapa faktor, maka metode optimasi multikriteria adalah metode yang tepat untuk digunakan. Sebuah metode optimasi multikriteria adalah metode untuk mencari satu set solusi yang disebut "*Pareto Optimal Solutions*" yang merupakan hasil kompromi atau *trade-off* dari beberapa kriteria (Mohammed, Wang, & Filip, 2017). Sebagai contoh beberapa kasus:

- Kasus 1
    1. Jumlah *agent* yang digunakan sejumlah 2 *agent*
    2. Dwelling time berkurang menjadi 3 hari
    3. Biaya yang dibutuhkan menjadi 1.5 dari biaya saat menggunakan 1 *agent*
- Kasus 2
    1. Jumah *agent* yang digunakan menjadi 3 *agent*
    2. Dwelling time berkurang menjadi hanya 1 hari
    3. Biaya yang dibutuhkan menjadi 3 kali lipat biaya asal saat menggunakan 1 *agent*

Dari 2 kasus diatas, metode optimasi yang diterpakan harus dapat memilih jumlah *agent* yang sesuai. Karena *dwelling time* yang ditetapkan untuk dipenuhi adalah 2-3 hari, maka penggunaan 3 *agent* terlalu berlebihan karena mengurangi *dwelling time* sampai hanya 1 hari dengan biaya yang terlalu mahal. Beberapa metode yang ada akan dibandingkan dan dicariyang terbaik berdasarkan nilai agregat dari seluruh metode.

2.8.1 SMAA-2

Stochastic Multicriteria Acceptability Analysis (SMAA) telah dikembangkan untuk mendukung pengambilan keputusan tanpa memberikan bobot pada kriteria yang ada (Lahdelma, Risto; Salminen, 2001). SMAA memperhitungkan *acceptability* dari kriteria yang ada dan memberikan peringkat terbaik untuk setiap alternatif. Metode ini baik untuk menilai alternatif dengan kriteria yang perbedaanya cukup ekstrim. Namun, hal ini menjadikan metode ini sulit digunakan untuk mengidentifikasi alternatif yang merupakan *trade-off* dari setiap kriteria yang ada.

SMAA-2 menyempurnakan kekurangan dari metode SMAA dengan menganalisis dan memberikan bobot pada setiap kriteria dan memberikan peringkat pada setiap alternatif. Dengan begitu, setiap alternatif akan diperhitungkan tingkat *acceptability*–nya berdasarkan bobot kriterianya juga.

2.8.2 MOORA

Multi Objective Optimization on the Basis of Ratio Analysis (MOORA) adalah metode optimasi yang sering digunakan untuk kasus dengan satu atau lebih kriteria. Kriteria optimasi yang digunakan untuk melakukan optimasi dengan metode ini dapat dibagi menjadi kriteria yang menguntungkan (*beneficial*) dan merugikan (*cost*). Sebagai contoh adalah optimasi untuk memaksimalkan performa dari mesin dan meminimalkan penggunaan bahan bakar, meminimalkan berat bahan dan memaksimalkan kekuatan bahan tersebut dalam untuk suatu material (Karande & Chakraborty, 2012).

Ide dasar dari MOORA adalah menentukan indeks performa dari alternatif-alternatif yang ada. Setiap kriteria dalam suatu alternatif akan dikategorikan dalam

kriteria yang *beneficial* atau *cost*. Jumlah total dari kriteria-kriteria tersebut akan digunakan untuk menentukan peringkat dari suatu alternatif terhadap alternatif yang lain (Stanujkic, Đorđević, & Đorđević, 2013).

2.8.3 COPRAS

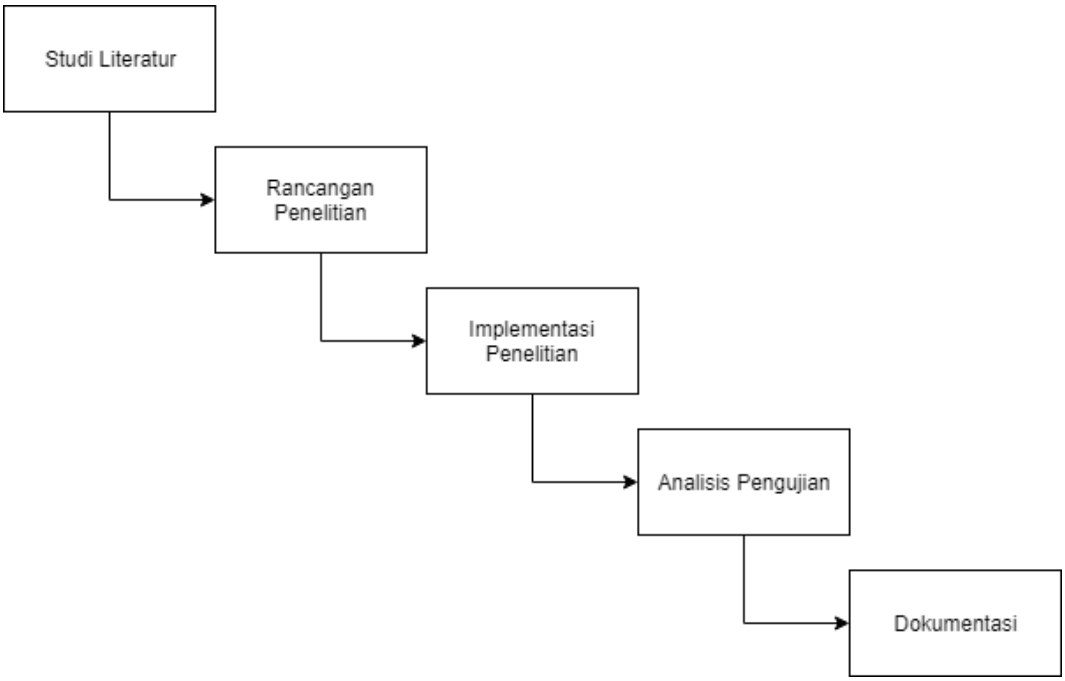
Complex Proportional Assessment (COPRAS) merupakan sebuah metode optimasi yang memiliki proses agregasi yang lebih kompleks daripada metode yang umum digunakan seperti SAW dan MOORA (Stanujkic et al., 2013). Sama dengan MOORA, metode ini memisahkan kriteria yang *beneficial* dan *cost*. Metode ini telah digunakan pada beberapa masalah optimasi sebelumnya, diantaranya adalah untuk menentukan alternatif terbaik untuk peremajaan gedung yang memiliki kriteria kuantitatif dan kualitatif (Kaklauskas, Zavadskas, & Raslanas, 2005), dan pemilihan kontraktor yang akan mengerjakan suatu proyek gedung (Kaklauskas et al., 2006).

COPRAS berhasil menunjukkan nilai optimum yang dibutuhkan peneliti pada penelitian sebelumnya. Nilai optimum yang paling tinggi dalam metode ini menunjukkan rekomendasi alternatif yang dipilih. Metode ini juga berhasil melakukan optimasi multikriteria dengan baik walaupun kriteria yang digunakan untuk optimasi merupakan kriteria kualitatif dan kuantitatif.

[Halaman ini sengaja dikosongkan]

BAB III
METODE PENELITIAN

Pada bab ini akan dijelaskan tentang metodologi penelitian yang digunakan pada penelitian ini. Adapun alur penelitian terdiri dari 5 tahap, meliputi (1) studi literatur, (2) rancangan penelitian, (3) implementasi penelitian, (4) analisis pengujian, (5) dokumentasi sistem. Ilustrasi alur metodologi penelitian ditujukkan pada Gambar 3.1.



Gambar 3.1 Desain Penelitian

Subbab pada bab ini akan menjelaskan metodologi penelitian pada Gambar 3.1 secara lebih terperinci.

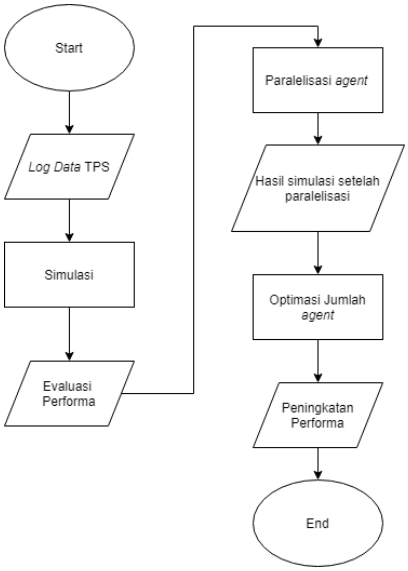
3.1 Studi Literatur

Penelitian ini diawali dengan studi literatur terkait dengan topik yang akan dibahas dalam penelitian ini. Beberapa topik yang akan dibahas adalah *agent based simulation*, paralelisasi *agent* dan metode-metode optimasi. Untuk topik *agent based simulation,* literatur yang akan digunakan adalah yang membahas *agent based simulation* untuk multiorganisasi. Dengan adanya literatur-literatur ini,

diharapkan kerangka kerja untuk penelitian ini untuk pemodelan agent based simulation dapat dibuat dengan baik. Untuk paralelisasi *agent*, literatur yang digunakan adalah literature yang membahas penerapan paralelisasi *agent* yang sudah digunakan sebelumnya. Untuk metode optimasi, dikarenakan kriteria yang harus dipenuhi untuk optimasi adalah waktu, biaya, dan jumlah *parallel agent*, literatur untuk metode optimasi yang akan digunakan adalah literatur yang terkait dengan *multicriteria optimization*. Untuk sumber literatur yang akan digunakan adalah jurnal dan konferensi internasional yang dipublikasi melalui sciencedirect, IEEE, atau springer. Literatur yang diambil menggunakan kata kunci *agent based simulation*, paralelisasi proses bisnis, paralelisasi *agent*, dan *multicriteria optimization*.

3.2 Rancangan Penelitian

Untuk kerangka kerja yang diusulkan yaitu: (1) mendapatkan event log; (2) simulasi dengan *agent based simulation*; (3) mendapatkan hasil evaluasi performa; (4) melakukan paralelisasi *agent*; (4) mendapatkan hasil simulasi setelah paralelisasi; (5) melakukan optimasi jumlah *agent*; (6) mendapatkan hasil peningkatan performa proses bisnis.



Gambar 3.2 Rancangan Penelitian

3.2.1 Simulasi

Simulasi akan dijalankan berdasarkan pada *log data* yang didapatkan dari PT.TPS untuk periode bulan Januari 2016 sampai Maret 2016. Gambar 3.3 menunjukkan contoh *log data* yang digunakan.

CONTAINER_NO	Start Time	End Time	Activity	CC_TT_No	HT_No	Ves_ID	NAMA_IMP
TRHU1782109		12/06/2015 17:15:03	Discharge	004	173	AJAI001	CV. CIPTA KARYA MAKMUR
WHLU8040720		12/06/2015 17:45:28	Discharge	003	212	AJAI001	CV. TRI JAYA MAKMUR
TRHU1782109	12/06/2015 17:49:50	02/07/2015	Yard	004	173	AJAI001	CV. CIPTA KARYA MAKMUR
WHLU8040720	12/06/2015 18:09:39	09/07/2015	Yard	003	212	AJAI001	CV. TRI JAYA MAKMUR
WHLU2801550		12/06/2015 20:36:28	Discharge	004	216	AJAI001	
WHLU2801550	12/06/2015 20:52:13	24/07/2015	Yard	004	216	AJAI001	
TRLU8819050		21/06/2015 19:06:29	Discharge	003	208	ALDI003	CV.YUDHA SOLUSI PRATAMA
DFSU1964638		21/06/2015 19:08:04	Discharge	003	208	ALDI003	PT. LAUTAN LUAS TBK
DFSU1964941		21/06/2015 19:14:21	Discharge	003	212	ALDI003	PT. JAKARANA TAMA
DFSU7474140		21/06/2015 19:19:49	Discharge	004	221	ALDI003	PT.GALANGCITRAMITRA MAJUMAPAN
DFSU1965043		21/06/2015 19:32:48	Discharge	004	230	ALDI003	PT. JAKARANA TAMA
CMAU2136921		21/06/2015 19:38:10	Discharge	004	187	ALDI003	PT. WIJAYA INDONESIA MAKMUR BICYCLE INDUSTRIES
TEMU2611128		21/06/2015 19:41:23	Discharge	004	217	ALDI003	PT. JINDAL STAINLESS INDONESIA
TRLU8819050	21/06/2015 19:25:02	02/07/2015	Yard	003	208	ALDI003	CV.YUDHA SOLUSI PRATAMA
DFSU1964638	21/06/2015 19:26:30	06/07/2015	Yard	003	208	ALDI003	PT. LAUTAN LUAS TBK
DFSU7474141	21/06/2015 19:31:53	01/07/2015	Yard	004	221	ALDI004	PT.GALANGCITRAMITRA MAJUMAPAN
DFSU1965043	21/06/2015 19:46:18	30/06/2015	Yard	004	230	ALDI003	PT. JAKARANA TAMA
BSIU3004563		21/06/2015 19:57:03	Discharge	004	173	ALDI003	PT. GUDANG GARAM TBK

Gambar 3.3 Contoh *log data* dari PT. TPS

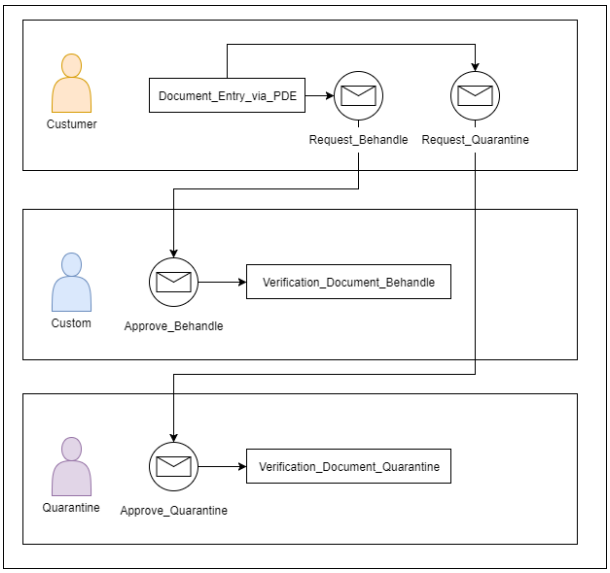
Pada Gambar 3.3 *log data* tersebut memiliki beberapa atribut, antara lain nomor kasus (*case_id*), nomor kontainer, waktu mulai, waktu selesai, dan aktivitas yang dijalankan.

Untuk membentuk model proses yang akan dijadikan sebagai model simulasi, *standard operation procedure* (SOP) untuk proses bisnis impor barang yang didapatkan dari survei lapangan dijadikan sebagai acuan. Dari hasil survei

lapangan, terdapat 4 organisasi yang terlibat dalam proses bisnis impor barang. Keempat organisasi tersebut antara lain:

e. *Customer*
f. PT. Terminal Petikemas Surabaya
g. Karantina
h. Bea Cukai

Masing-masing organisasi yang terlibat akan diwakili oleh *agent* dalam *agent based simulation* yang telah dibuat. Beberapa dari *agent* diatas melakukan komunikasi satu sama lain pada saat mengerjakan suatu *task* yang ada pada proses bisnis impor barang. Gambar 3.4 menunjukkan komunikasi antar organisasi yang terjadi.



Gambar 3.4 Komunikasi antar organisasi

Pada Gambar 3.4 telah ditunjukkan bagaimana gambaran komunikasi antar organisasi untuk dimodelkan dalam simulasi. Pada proses berjalannya proses bisnis impor barang ini, Karantina dan Bea Cukai/Custom memiliki karakteristik *asynchronous* yang dimana aktivitas pada kedua organisasi tersebut tidak akan dimulai bila belum menerima pesan dari *Customer*. Untuk lebih detailnya:

1. Karantina

   Pihak karantina diharuskan menerima *message* dari pihak *customer*, dalam hal ini disebut *requestQuarantine*. Pihak karantina tidak dapat mengatur kapan permintaan karantina dari *customer* akan diterima. Karena hal ini, kontainer dapat menumpuk pada *yard* saat menunggu proses karantina. Hal ini tentu berakibat pada lamanya *dwelling time* kontainer tersebut.

2. Bea Cukai

   Pihak Bea Cukai juga memiliki pola yang sama dengan pihak karantina. *Customer* akan mengirim pesan *requestBehandle* sebelum pihak Bea Cukai mulai melakukan proses *behandle*. Pihak Bea Cukai juga tidak dapat mengatur kapan pesan dari *customer* akan diterima.

Berikut ini adalah prosedur standar yang dilakukan pada proses bisnis impor barang di PT. TPS mengacu pada penelitian di lapangan:

Prosedur layanan pembongkaran petikemas (*discharge*)

1. Perencanaan; pelanggan harus melengkapi dokumen :
   a. *Master Cable*
   b. CVIA (*Container Vessel Identification Advice* = Pemberitahuan Identifikasi Kapal Petikemas)
   c. *Statement of Fact* (Surat Pernyataan Keadaan)
   d. *Statement Letter* (email baplie file)
   e. *Import Summary List* (ISL = Daftar Ringkasan Impor)
   f. *Dangerous Cargo List* (Daftar Kargo Berbahaya)
   g. *Approval from Harbor Master* (Surat Ijin dari Syahbandar)
   h. *Reefer List* (Daftar Reefer)
   i. *Crane Sequence List* (Daftar Urutan Crane)
   j. *Discharge Stowage Plan* (Rencana Penyimpanan Pembongkaran)
   k. *Discharge Bay Plan* (Rencana Bay Pembongkaran)
   l. *Manifes*t
   m. *Special Cargo List* (Daftar Kargo Khusus)

2. *Yard and Berth Planning Sub-department* (Sub-departemen Perencanaan Lapangan dan Dermaga) memeriksa dokumen. Mereka mengadakan rapat harian, bersama dengan Departemen Teknik, dengan Perusahaan Pelayaran, untuk merencanakan jadwal layanan penanganan petikemas.
3. *Vessel Berth Planning Sub-department* (Sub-departemen Perencanaan Lapangan dan Dermaga) memproses rencana pembongkaran ke dalam sistem komputer berdasarkan data yang dikirimkan oleh Perusahaan Pelayaran lewat email, dan mencetak *Discharge List* (Daftar Pembongkaran) dan menyerahkannya kepada *Berth Operations* (Operasi Dermaga).
4. Berdasarkan *Discharge List* (Daftar Pembongkaran), *Berth Operations Superintendent* (Superitenden Operasi Dermaga) memerintahkan Operator CC, lewat Petugas *Tally* Dermaga, untuk membongkar petikemas dari atas kapal dan memuatnya ke atas *chassis Head Truck*, dan membawanya ke Lapangan Penumpukan Petikemas, dan mengkonfirmasi posisi pembongkaran ke dalam sistem komputer (HHT/Teklogix)
5. Setelah Head Truck tiba di Lapangan Penumpukan Petikemas, *Yard Operations Superintendent* (Superintenden Operasi Lapangan) memerintahkan Operator RTG, lewat Petugas *Tally* Lapangan, untuk menumpuk petikemas, dan mengkonfirmasi posisi petikemas ke dalam sistem komputer (HHT/Teklogix). Petugas *Tally* Lapangan memerintahkan pengemudi *Head Truck* untuk kembali ke Dermaga untuk mengambil petikemas selanjutnya yang akan dibongkar.
6. Pada akhir shift, Petugas *Tally* Lapangan melaporkan hasil pekerjaan kepada Superintenden Operasi Lapangan, sedangkan Petugas *Tally* Dermaga melaporkan hasil pekerjaan kepada Superintenden Operasi Dermaga.

Prosedur Pemeriksaan Bea Cukai (*Behandle*)

1. Pelanggan menyerahkan Surat Permohonan Behandle Barang kepada TPS lewat Petugas Layanan Administrasi dilengkapi dengan Dokumen Asli, Warkat Dana, dan Perintah Pengeluaran.

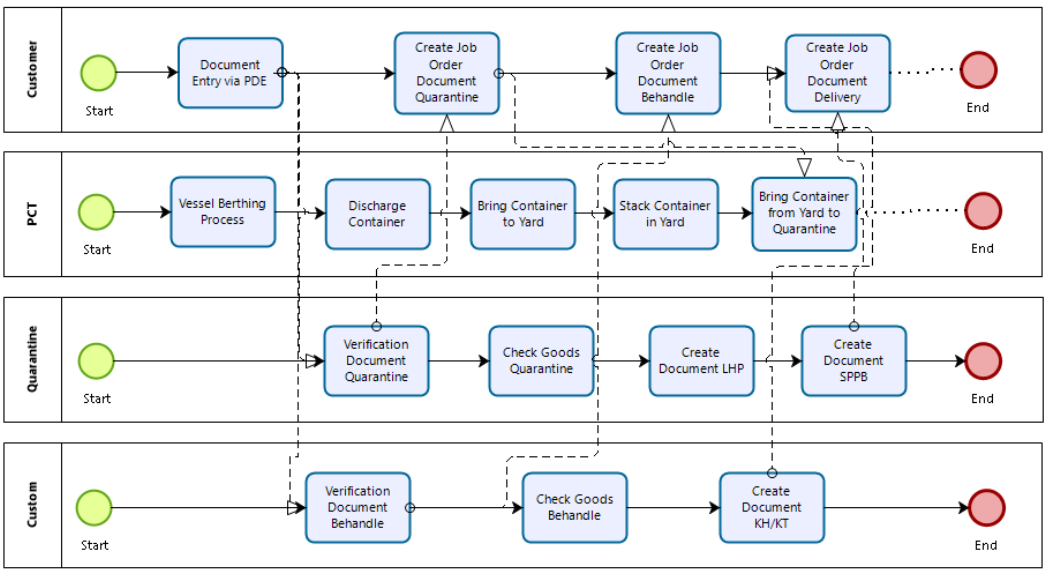
2. Petugas Administrasi memeriksa dan mencetak Job Order dan menyerahkan kepada Pelanggan, dengan salinan Perintah Pengeluaran, dan menyerahkan 2 lembar salinan kepada Petugas Layanan Operasi.
3. Pelanggan menyerahkan *Job Order* kepada CFS *Operations Assistant Manager*.
4. Asisten Manajer Operasi CFS atau Staff yang ditunjuk memeriksa dokumen dan menerbitkan *Container Movement Job* (Pekerjaan Pergerakan Petikemas) untuk menarik petikemas dari Lapangan Penumpukan ke CFS.
5. Setelah petikemas telah dipindahkan ke CFS, keadaan fisik petikemas akan diperiksa sebelum pemeriksaan Behandle dilakukan.
6. Setelah pengeluaran barang telah selesai dilakukan, laporan harus disiapkan dan diketahui oleh Petugas CFS, dan disetujui oleh Pelanggan.

Prosedur layanan pengeluaran petikemas (*delivery*)

1. Perencanaan pelanggan harus melengkapi dokumen:
   a. Surat Permohonan Pengeluaran Petikemas
   b. Surat Asli Perintah Pengeluaran (DO = Delivery Order)
   c. Penyediaan Warkat Dana (Pembayaran di Depan) (masing-masing 4 lembar) untuk diserahkan kepada Import Service Staff (Petugas Layanan Impor).
   d. SPPB = Surat Persetujuan Pengeluaran Barang dan Surat Pernyataan PP (Pencekalan dan Pencegahan) dari Bea Cukai
   e. Surat Kuasa dari Importir
2. Petugas Layanan Impor mencetak CEIR/Job Order yang telah disetujui oleh *Import Superintendent* (Superintenden Impor). Lembar ke 1, 2, dan 3 CEIR diserahkan kepada Pelanggan. Pelanggan menyerahkan kepada pengemudi *Head Truck*.
3. Pengemudi *Head Truck* menuju ke *In-Gate* (Gerbang Masuk) dan menyerahkan *Job Order*/CEIR kepada *In-Gate Staff* (Petugas Gerbang Masuk).

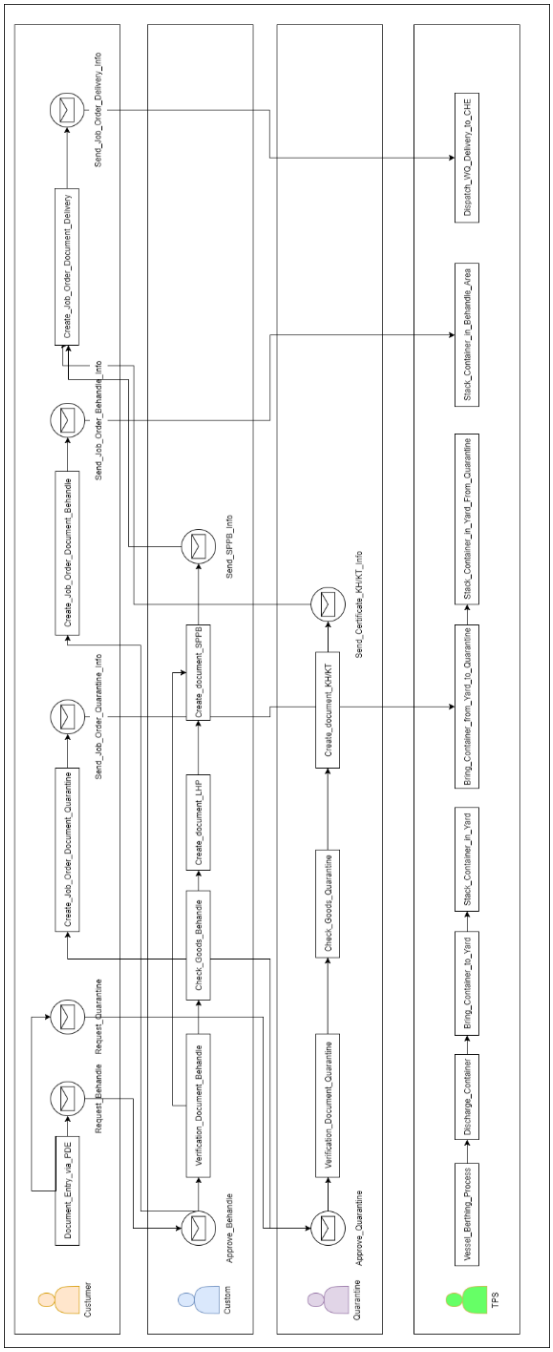
4. *In-Gate Staff* mencetak *In-Gate Terminal Job Slip* berdasarkan *Job Order*/CEIR dan mengembalikan lembar ke 1 dan 2 kepada pengemudi *Head Truck*.
5. Pengemudi *Head Truck* menyerahkan *In-Gate Terminal Job Slip* dan Job Order/CEIR kepada Petugas *Tally* Lapangan.
6. Petugas Tally Lapangan memerintahkan Operator RTG untuk mengangkat petikemas dari Lapangan Penumpukan ke atas *chassis Head Truck* sesuai dengan posisi yang tercantum dalam *In-Gate Terminal Job Slip*.
7. Pengemudi *Head Truck* menerima *Job Order*/CEIR dan *In-Gate Terminal Job Slip* dari Petugas Tally Lapangan bergerak menuju *Out-Gate* (Gerbang Keluar) dan menyerahkan *In-Gate Terminal Job Slip* dan *Job Order*/CEIR lembar ke 3 kepada Petugas *Out-Gate*, dan Surat Pernyataan Pecekalan dan Pencegahan (PP) kepada Petugas Bea Cukai.

Petugas Out-Gate mengkonfirmasi nomor polisi *Head Truck* dan nomor referensi kerja *Head Truck* berdasarkan *In-Gate Terminal Job Slip* ke dalam sistem computer dengan dilampiri lembar ke 1 CEIR kepada pengemudi *Head Truck*. Untuk lebih jelasnya, sebagian SOP yang didapat digambarkan seperti pada Gambar 3.5.



Gambar 3.5 Bagian dari SOP

Selanjutnya, dibuatlah model simulasi proses bisnis impor barang menggunakan *library* SPADE dengan Bahasa pemrograman Python berdasarkan pada SOP yang telah didapatkan seperti pada Gambar 3.5.



Gambar 3.6 Model koreografi proses bisnis yang diterapkan

Pada Gambar 3.6 telah ditunjukkan model simulasi yang dipakai dalam penelitian ini. Masing-masing agent harus mengerjakan *task* yang diberikan sesuai urutan yang ada masing-masing satu *task* dalam satu waktu. Sedangkan, untuk waktu

mulai, *waiting time*, *execution time* dan *cost* untuk mengerjakan suatu *task* akan didasarkan pada event log yang ada. Karena simulasi ini merupakan simulasi *asynchronous*, maka pengerjaan *task* selanjutnya akan bergantung pada proses berjalannya simulasi.

3.2.2 Evaluasi Performa

Pada tahap ini, dilakukan analisis performa berdasarkan hasil simulasi. Mengacu pada penelitian terdahulu oleh (Rozinat et al., 2008), performa didapatkan melalui perhitungan waktu tunggu (*waiting time*), waktu eksekusi (*execution time*) dan *sojourn time*.

a. Waktu eksekusi (*execution time*) adalah waktu yang dibutuhkan selama berlangsungnya perintah eksekusi dari awal mulai proses hingga selesainya proses.
b. Waktu tunggu (*waiting time*) adalah waktu yang dibutuhkan oleh suatu proses selama menunggu di *ready queue*.
c. Waktu sojourn (*sojourn time*) adalah total waktu untuk keseluruhan pelaksanaan proses dari awal hingga akhir dan termasuk *waiting time*, atau dengan kata lain sojourn time adalah *execution time* ditambahkan dengan *waiting time*.

Pada penelitian terdahulu, metode untuk menghitung performa didasarkan pada simulasi *discrete-event*. Artinya, simulasi bergantung pada aktivitas proses bisnis yang direpresentasikan dalam bentuk *log data*. Proses bisnis pada simulasi *discrete-event* hanya memiliki single organisasi. Sehingga tidak terdapat *message* untuk komunikasi antar *agent* ketika simulasi dijalankan.

Input: event log
Output: sojourn time (st), execution time (et), waiting time (wt)

Inisialisasi case_id
for *iter* = 1 to N_{iter} do
    for case_{id} to Ncase_{id} do
        if label == activity
        $\text{et\_activity}^{N} = (\text{t2\_activity}^{N} - \text{t1\_activity}^{N})$
        $\text{wt\_activity}^{N} = (\text{t1\_activity}^{N+1} - \text{t2\_activity}^{N})$
        $\text{st\_activity}^{N} = \text{wt\_activity}^{N} + \text{et\_Activity}^{N}$
    end
end

Gambar 3.7 Algoritma evaluasi performa pada descrete-event simulation

Jika menggunakan algoritma pada Gambar 3.7, maka tidak dapat diterapkan pada kasus yang sedang diteliti saat ini. Oleh karena itu, penelitian ini mengusulkan untuk membuat metode yang modifikasi agar dapat menghitung performa multi organisasi yang bersifat *asynchronous*. Oleh karena itu, memungkinkan untuk memilah antara *message* dan aktivitas pada *log data* yang hanya memiliki *single time stamp*.

```
Input: event log
Output: sojourn time (st), execution time (et), waiting time (wt)

Inisialisasi case_id
for iter = 1 to N_iter do
        for case_id  to Ncase_id do
                if label == message
                        st_message = t_message+……n
                elseif label == activity
                        st_activity = t_activity+……n
        end
end

hitung
        total_st = st_message + st_activity
        average st_message ; average st_activity
        st_dev st_message ; st_dev st_activity


if label ==message
        then
                et_message=RandomNormal(min st_message, st_dev
        message)
                wt_message= st_message-et_message

if_label == activity
        then
                et_activity=RandomNormal(min st_activity, st_dev
        activity)
                wt_activity= st_activity-et_activity
```

Gambar 3.8 Metode modifikasi untuk menangani *single time stamp*

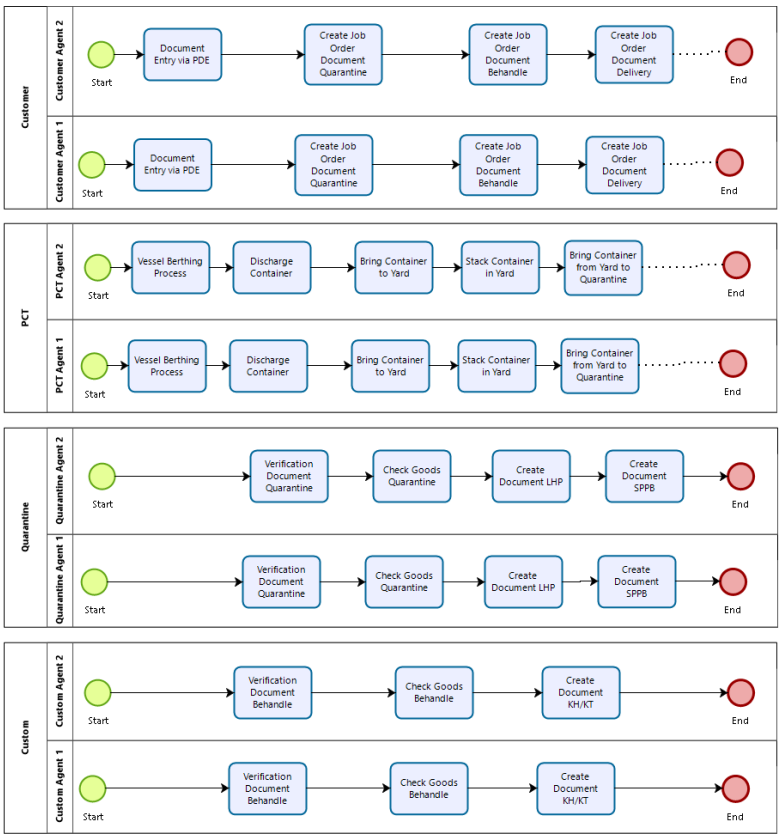
Gambar 3.8 menunjukan metode modifikasi yang digunakan pada penelitian sebelumnya (Fajar et al., 2018) untuk memperoleh *execution time* dan *waiting time* pada aktivitas/*message* pada *single time stamp log data*. Penentuan *execution time* dan *waiting time* dilakukan per aktivitas/*message* mengacu pada hasil *sojourn time* tiap aktivitas/*message*. Berikut ini adalah metode untuk membentuk *execution time* dan *waiting time*:

a. Tentukan nilai minimum (MIN) *sojourn time* aktivitas/*message* n
b. Tentukan standar deviasi dari *sojourn time* untuk aktivitas/*message* n
c. *Execution time* = NormalRandom(MIN *sojourn time* aktivitas/*message* n, standar deviasi dari *sojourn time* untuk aktivitas/*message* n)
d. *Waiting time = Sojourn time – execution time*

Metode yang digunakan didasarkan bahwa nilai minimum *sojourn time* adalah nilai *execution time* yang memiliki waiting time = 0, sehingga *sojourn time* pasti akan memiliki *execution time* dan *waiting time* tidak sama dengan 0 ketika *sojourn time* lebih dari nilai minimumnya. Kemudian, pembentukan bilangan random berbasis distribusi normal berdasarkan 2 parameter, yaitu nilai minimum *sojourn time* dan standar deviasi dari *sojourn time.* Adapun syarat pembentukan bilangan random tersebut adalah (a) *execution time* tidak boleh kurang atau sama dengan 0; (b) *execution time* tidak boleh lebih besar dari *sojourn time.*

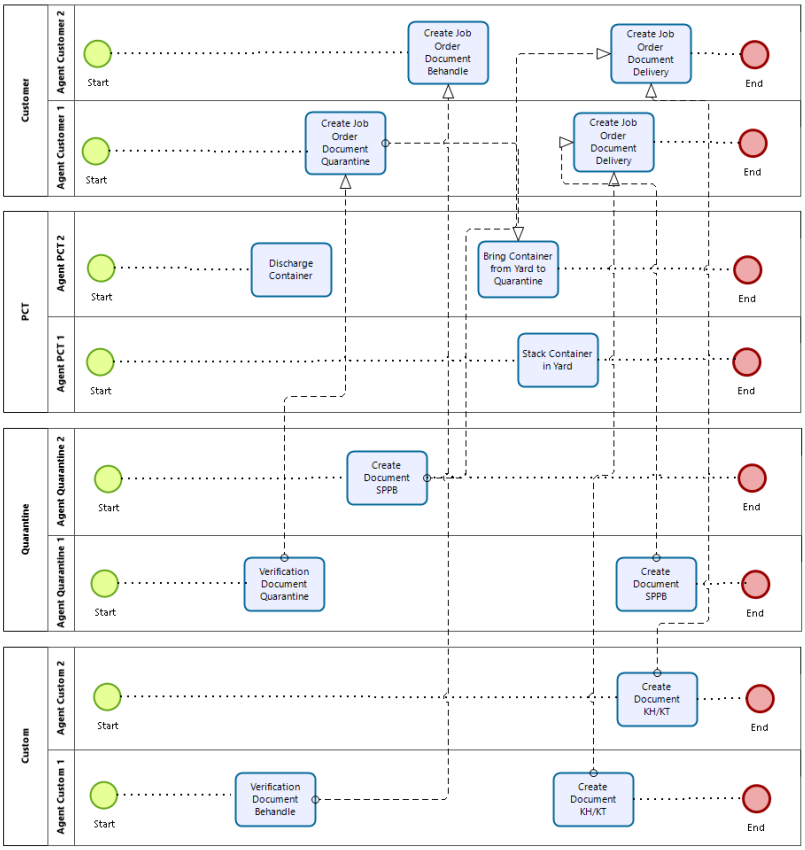
3.2.3 Paralelisasi Agen

Dari penelitian sebelumnya (Fajar et al., 2018), ditemukan adanya *Asynchronous Waiting Time* (AWT). Paralelisasi diharapkan dapat mengurangi *asynchronous waiting time* ini sehingga *dwelling time* dari kontainer dapat berkurang. Gambar 3.9 menunjukkan bagaimana paralelisasi dilakukan.



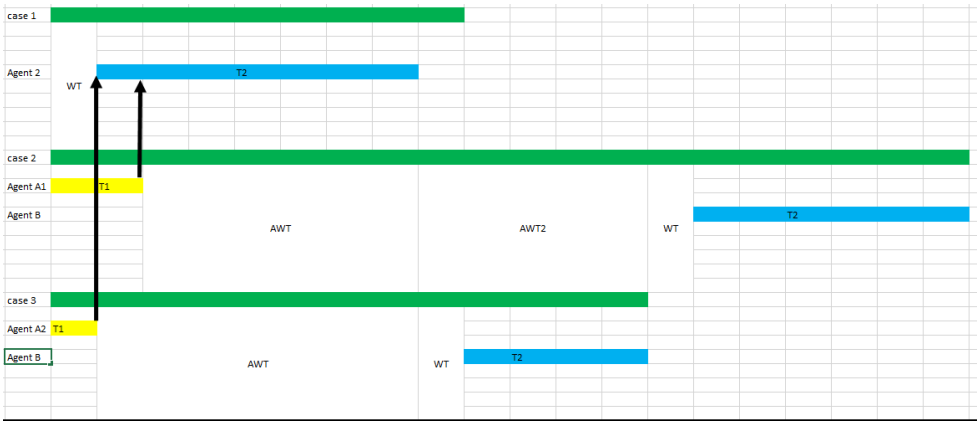
Gambar 3.9 Paralelisasi *agent*

Seperti terlihat pada Gambar 3.9, setiap organisasi akan diparalelkan *agent*-nya. Masing-masing *agent* harus dapat mengerjakan *task* sesuai SOP yang ada. Sehingga, *agent* yang sedang tidak mengerjakan *task* dapat mengambil *task* yang harus dikerjakan karena adanya *message* yang masuk. Dengan begitu, AWT dapat berkurang. Untuk lebih jelasnya, Gambar 3.10 menunjukkan bagaimana jalannya paralelisasi saat ada *message*.



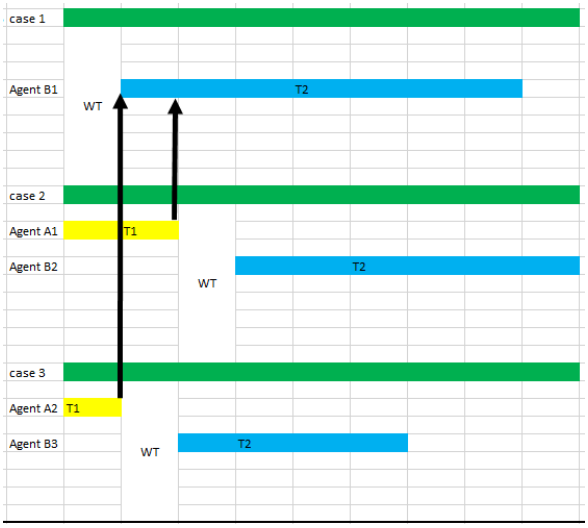
Gambar 3.10. Hasil paralelisasi

Dari Gambar 3.10 ditunjukkan bagaimana sebuah task dikerjakan apabila sebuah *agent* masih mengerjakan *task* lain. *Agent* parallelnya dapat mengambil *task* yang harus dikerjakan karena adanya *message* yang masuk. Untuk melakukan paralelisasinya, tiap organisasi akan diberi jumlah *agent* yang sama. Hal ini dikarenakan apabila hanya agent dari organisasi tertentu saja yang diparalelkan, agent dari organisasi untuk *task* selanjutnya akan mengalami *bottleneck* juga. Sebagai contoh, *task Create Job Order Delivery* memerlukan 2 *message* untuk mulai mengerjakan *task*, apabila hanya *agent* yang mengirim *message* saja yang diparalelisasi maka saat beberapa *agent* pengirim menyelesaikan beberapa *task* yang sama secara bersamaan, *agent* dari *customer* tidak dapat menyelesaikan dengan cepat dan bertambahlah AWT dari *task Create Job Order Delivery*. Gambar 3.10 menunjukkan secara detail bagaimana sebuah *task* akan dilakukan apabila sebuah *agent* yang diberikan *task* tersebut sedang mengerjakan *task* lain.



Gambar 3.11 Eksekusi aktivitas satu *task*

Dengan memberikan lebih banyak *agent* dalam mengerjakan *task* T2 yang diberikan yang ada pada Gambar 3.11, diharapkan AWT dari aktivitas-aktivitas tersebut akan berkurang. Sebagai contoh, jumlah *agent* untuk TPS, Bea Cukai, *Customer*, dan karantina ketika diparallelisasi 10 *agent* adalah 40 *agent*, sehingga ada 10 agent untuk masing-masing dari organsisasi. Gambar 3.11 menunjukkan bagaimana aktivitas dijalankan setelah dilakukan penambahan *agent* untuk mengerjakan *task* T2.



Gambar 3.12 eksekusi aktivitas setelah paralelisasi *agent*

Dengan berkurangnya AWT seperti pada Gambar 3.12, biaya untuk melakukan aktivitas yang ada juga lebih sedikit. Untuk melakukan perhitungan terhadap biaya yang dikeluarkan setelah berkurangnya AWT, kita harus

memisahkan biaya saat dilakukannya eksekusi aktivitas (*Execution cost*) dan sebelum dilakukan eksekusi pada aktivitas tersebut (*Waiting cost*). Untuk menentukan *execution cost*, biaya minimal dari sebuah aktivitas dalam *log data* dianggap hanya memiliki *execution cost*. Selanjutnya, dilakukan distribusi normal menggunakan standar deviasi biaya sebuah aktivitas dan biaya minimal aktivitas tersebut. Fungsi random normal ditunjukkan pada Persamaan 1.

$$C_e = ae^{\left(-\frac{(x-b)^2}{2c^2}\right)} \quad (1)$$

C_e = biaya saat aktivitas dieksekusi

e = 2.71828

x = Variabel normal random

a = 1/(c * $\sqrt{2\pi}$)

b = Biaya minimal suatu aktivitas

c = Standar deviasi

Setelah *execution cost* didapatkan, kita dapat memperoleh biaya yang dibutuhkan tiap satuan waktu ketika aktivitas belum dieksekusi. Untuk menghitung total *cost* setelah paralelisasi, digunakan Persamaan 2.

$$C = \left(\frac{W}{W_{Tot}} \times C_w\right) + \left(\frac{W_a}{W_{Tot}} \times C_w\right) + C_e \quad (2)$$

W = waktu tunggu aktivitas

W_a = waktu tunggu karena eksekusi aktivitas lain (AWT)

W_{Tot} = waktu tunggu total

C_w = biaya sebelum aktivitas dieksekusi

C = biaya total

3.2.4 Optimasi Jumlah Agen

Pada tahap ini, setelah paralelisasi *agent* dilakukan, maka dilakukanlah evaluasi performa kembali pada proses bisnis yang telah diparalelisasi. Hasil dari evaluasi performa ini adalah waktu, biaya, dan jumlah *agent* yang diterapkan.

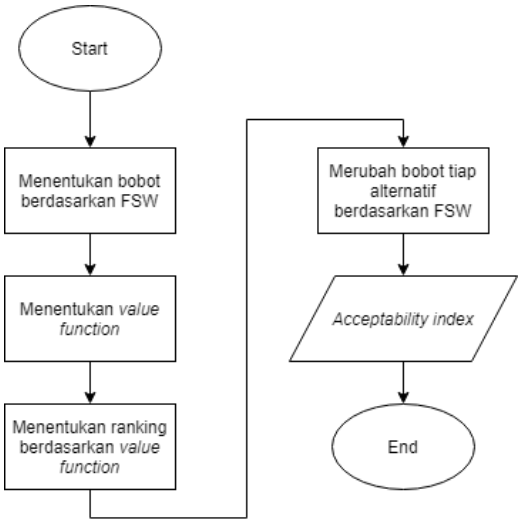
Beberapa parameter inilah yang akan dijadikan masukan untuk metode optimasi jumlah *agent*. Sedangkan, fungsi objektif dari optimasi ini adalah seperti ditunjukkan pada Persamaan 3.

$$BA = \min W_{Tot} + \min C \tag{3}$$

Karena parameter yang dijadikan input lebih dari satu, maka metode optimasi multikriteria akan digunakan untuk melakukan optimasi. Metode optimasi multikriteria yang akan digunakan dalam penelitian ini adalah metode SMAA-2, MOORA, dan COPRAS. Hasil dari tiap simulasi akan dibandingkan dengan agregat dari semua simulasi untuk menentukan metode yang paling baik (Lee & Chang, 2018)

3.2.4.1 SMAA-2

Salah satu metode optimasi multikriteria adalah *Stochastic Multicriteria Acceptability Analysis-2* (SMAA-2). Gambar 3.13 menunjukkan beberapa langkah untuk melakukan optimasi dengan menggunakan metode SMAA-2 adalah sebagai berikut (Tervonen & Lahdelma, 2007):



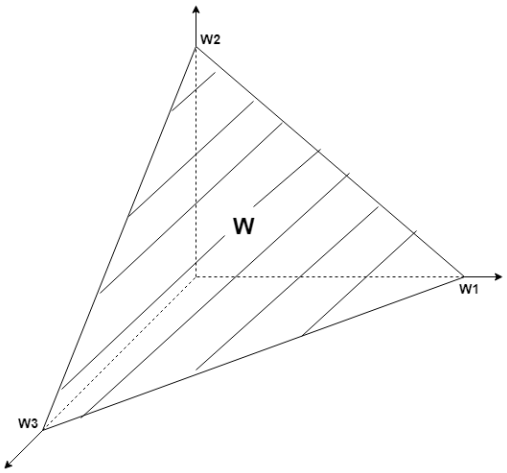
Gambar 3.13 *Flowchart* SMAA-2

SMAA-2 menggunakan analisis daerah bobot (*weight space*) untuk mendeskripsikan tiap alternatif pilihan sehingga dapat menunjukkan preferensi

yang memiliki bobot tertinggi sehingga dapat diberikan peringkat pada tiap alternatif pilihan berdasarkan bobot tersebut. Setiap alternatif pilihan dilambangkan dengan sebuah set $m\ \{x_1, x_2, x_3, \dots, x_m\}$ yang akan dievaluasi berdasarkan kriteria n. Pada kasus ini, preferensi dari pengambil keputusan akan direpresentasikan dalam *weight space* W. Persamaan 4 mendefinisikan *weight space* dalam *n* dimensi.

$$W = \left\{ w \in R^n : \ w \geqslant 0 \quad \text{and} \quad \sum_{j=1}^{n} w_j = 1 \right\}. \tag{4}$$

Gambar 3.14 menunjukkan *feasible weight space* (FSW) dari 3 kriteria. Masing-masing sudut dari segitiga terhubung dengan titik pojok dari tiap kriteria (1,0,0), (0,1,0), dan (0,0,1).



Gambar 3.14 FSW dari tiga kriteria

Selanjutnya, fungsi nilai digunakan untuk memetakan kriteria dan distribusi bobot menjadi distribusi nilai $u(x_i, w)$. Persamaan 5 menunjukkan fungsi nilai linear.

$$u(x_i, w) = \sum_{j=1}^{n} u_j(g_j(x_i))w_j \tag{5}$$

g_j = kriteria dalam alternatif *j*

Berdasarkan distribusi nilai ini, tiap alternatif diberikan rangking antara 1 sampai m dengan Persamaan 6.

$$rank(1, x, w) = 1 + \sum_{k=1}^{m} \rho\big(u(x_k, w) > u(x_i, w)\big) \tag{6}$$

Dimana bila ρ(true) = 1 dan ρ(false) = 0. Persamaan 7 menunjukkan perubahan bobot dimana $w \in W_i^r(\xi)$ dapat dilakukan sehingga alternatif x_i diberikan *rank* r .

$$W_i^r(\xi) = \{w \in W : \text{rank}(i, \xi, w) = r\}. \tag{7}$$

Hasil dari SMAA-2 ini merupakan *rank acceptability index* b_i^r yang menunjukkan variasi preferensi (bobot) yang memberi sebuah alternatif *rank r*. Sebuah alternatif dianggap dapat menempati *rank r* apabila bobot yang masuk dalam FSW dari tiap kriteria diubah. Persamaan 8 menunjukkan perhitungan dari *rank acceptability index*.

$$b_i^r = \int_{\xi \in X} f_X(\xi) \int_{w \in W_i^r(\xi)} f_W(w)\, \mathrm{d}w\, \mathrm{d}\xi. \tag{9}$$

Alternatif terbaik adalah alternatif yang memiliki tingkat *acceptability* paling tinggi. Nilai dari b_i^r ini adalah antara 0-1 dimana nilai 1 menunjukkan bahwa alternatif tersebut akan mendapatkan peringkat tersebut walaupun pilihan bobot kriterianya diganti. Sedangkan 0 menunjukkan bahwa alternative tersebut tidak akan pernah mendapatkan peringkat tersebut.

3.2.4.2 MOORA

Multi Objective Optimization on the Basis of Ratio Analysis (MOORA) merupakan sistem pendukung keputusan multiobjektif yang memungkinkan kita mendapatkan nilai optimal dari dua atau lebih kriteria. Langkah-langkah dalam melakukan optimasi menggunakan metode MOORA adalah sebagai berikut:

Menentukan fungsi objektif adalah langkah pertama dalam metode ini. Karena fungsi objektifnya telah ditentukan, langkah selanjutnya adalah membuat *decision matrix* yang memuat semua informasi berdasarkan kriteria yang ada. *Decision matrix* ini ditunjukkan seperti pada Gambar 3.15.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \cdots & x_{mn} \end{bmatrix}$$

Gambar 3.15 *Decision matrix*

Setelah didapatkan *decision matrix*, langkah selanjutnya adalah menentukan denominator dari tiap kolom/kriteria pada *decision matrix* tersebut menggunakan Persamaan 10.

$$d_j = \sqrt{\sum_{i=1}^{m} (x_{ij})^2} \tag{10}$$

d_j= denominator kriteria *j*

Setelah denominatornya ditemukan, maka nilai kriteria tiap alternatif dibagi dengan denominatornya. Dari perhitungan ini akan ditemukan *normalized performance* (r) tiap kriteria pada setiap alternatif yang ada. Hal ini ditunjukkan pada Persamaan 11.

$$r_{ij} = \frac{x_{ij}}{d_j} \tag{11}$$

r_{ij} = rasio dari tiap kriteria *j* untuk alternatif i

Langkah terakhir dari metode ini adalah memberikan peringkat pada tiap alternatif. Kriteria yang ada akan dipisah berdasarkan atribut yang menguntungkan atau merugikan pada penentuan keputusan. Dari tiap alternatif, nilai dari atribut

yang menguntungkan akan ditambahkan sehingga menambah nilai dari alternatif tersebut. Sebaliknya, nilai atribut yang merugikan akan mengurangi nilai dari alternatif tersebut.

$$Q_i = S_{+i} - S_{-i} \tag{12}$$

dimana,

$$S_{+i} = \sum_{j \in \Omega_{max}} w_j r_{ij} \tag{13}$$

$$S_{-i} = \sum_{j \in \Omega_{min}} w_j r_{ij} \tag{14}$$

Q_i = nilai optimasi alternatif i

Ω_{max} = kriteria *beneficial*

Ω_{min} = kriteria *non-beneficial*

w_j = bobot kriteria j

Dari nilai Q_i dapat diketahui peringkat dari tiap alternatif. Nilai Q_i yang besar menunjukkan bahwa alternatif tersebut merupakan alternatif yang lebih baik daripada alternatif dengan nilai Q_i yang lebih rendah. Algoritma untuk metode MOORA ditampilkan pada Tabel 3.1.

Tabel 3.1 Algoritma MOORA

Input: dataset kriteria positif pdata, dataset kriteria negatif ndata
Output: Nilai optimasi $\boldsymbol{Q_i}$
Algorithm MOORA

Inisialisasi bobot $\boldsymbol{w}$, normalized performance r, array $\boldsymbol{rpos}$, array $\boldsymbol{rneg}$, array $\boldsymbol{S_{+i}}$, array $\boldsymbol{S_{-i}}$

Array $\boldsymbol{k}$ = pdata

Array $\boldsymbol{l}$ = ndata

Foreach row $\boldsymbol{\epsilon\ k, l}$

Transpose $\boldsymbol{k, l}$

$\boldsymbol{x_{ij}\ \epsilon\ k, l}$

$\boldsymbol{r_{ji} = x_{ij}\ /\ sqrt(sum(x_i^2))}$

$\boldsymbol{rpos}$ = **append** $\boldsymbol{r_{ji},\ x_{ij}\ \epsilon\ k}$

$\boldsymbol{rneg}$ = **append** $\boldsymbol{r_{ji}, x_{ij}\ \epsilon\ l}$

End for

Foreach row $\boldsymbol{\epsilon\ rpos, rneg}$

$\boldsymbol{S_{+i}}$ = **append** sum$(\boldsymbol{w_j . r_{ij}}), \boldsymbol{r_{ij}\ \epsilon\ rpos}$

$\boldsymbol{S_{-i}}$ = **append** sum$(\boldsymbol{w_j . r_{ij}}), \boldsymbol{r_{ij}\ \epsilon\ rneg}$

Foreach row $\boldsymbol{\epsilon\ S_{+i}, S_{-i}}$

$\boldsymbol{Q_i = S_{+i} + S_{-i}}$

End for

3.2.4.3 COPRAS

Complex proportional assessment (COPRAS) merupakan metode yang memiliki agregasi kompleks. Langkah-langkah dalam melakukan optimasi menggunakan COPRAS adalah sebagai berikut:

Langkah pertama dalam metode ini adalah membuat *decision matrix* seperti pada Gambar 3.15 sehingga semua kriteria dalam tiap alternatif akan masuk didalamnya. Selanjutnya, setiap kriteria akan dimasukkan dalam satu kategori, yaitu kategori positif (*benefit*) atau negatif (*cost*). Setelah *decision matrix* dibuat, maka denominator d_j ditentukan menggunakan Persamaan 15.

$$d_j = \sum_{i=1}^{m} x_{ij}$$

(15)

Langkah selanjutnya adalah menentukan nilai r dari tiap kriteria dari setiap alternatif dengan Persamaan 11. Dalam menentukan nilai optimasinya, metode COPRAS memberikan peringkat pada setiap alternatif berdasarkan Persamaan 16.

$$Q_i = S_{+i} + \frac{S_{-min} \sum_{i=1}^{m} S_{-i}}{S_{-i} \sum_{i=1}^{m} \frac{S_{-min}}{S_{-i}}}$$

(16)

dimana,

$$S_{-min} = \min_i S_{-i}$$

(17)

Persamaan 14 dapat disederhanakan menjadi seperti yang ditunjukkan pada Persamaan 18 untuk melakukan optimasi menggunakan metode COPRAS ini.

$$Q_i = S_{+i} + \frac{\sum_{i=1}^{m} S_{-i}}{S_{-i} \sum_{i=1}^{m} \frac{1}{S_{-i}}}$$

(18)

Dari nilai Q_i dapat diketahui peringkat dari tiap alternatif. Nilai Q_i yang paling tinggi menunjukkan alternatif yang paling optimal dibandingkan alternatif-alternatif yang lain. Tabel 3.2 menunjukkan algoritma untuk optimasi menggunakan metode COPRAS.

Tabel 3.2 Algoritma COPRAS

Input: dataset kriteria positif pdata, dataset kriteria negatif ndata
Output: Nilai optimasi $\boldsymbol{Q_i}$
Algorithm COPRAS
Inisialisasi bobot $\boldsymbol{w}$, rasio r, array $\boldsymbol{rpos}$, array $\boldsymbol{rneg}$, array $\boldsymbol{S_{+i}}$, array $\boldsymbol{S_{-i}}$

Array $\boldsymbol{k}$ = pdata

Array $\boldsymbol{l}$ = ndata

Tentukan bobot masing-masing kriteria $\boldsymbol{w_j}$

Foreach row $\boldsymbol{\epsilon\ k, l}$

Transpose $\boldsymbol{k, l}$

$\boldsymbol{x_{ij}\ \epsilon\ k, l}$

$\boldsymbol{r_{ji} = x_{ij}\ /\ sum(x_i))}$

$\boldsymbol{rpos}$ = **append** $\boldsymbol{r_{ji},\ x_{ij}\ \epsilon\ k}$

$\boldsymbol{rneg}$ = **append** $\boldsymbol{r_{ji}, x_{ij}\ \epsilon\ l}$

End for

Foreach row $\boldsymbol{\epsilon\ rpos, rneg}$

$\boldsymbol{S_{+i}}$ = **append** sum$(\boldsymbol{w_j . r_{ij}}), \boldsymbol{r_{ij}\ \epsilon\ rpos}$

$\boldsymbol{S_{-i}}$ = **append** sum$(\boldsymbol{w_j . r_{ij}}), \boldsymbol{r_{ij}\ \epsilon\ rneg}$

End for

Foreach row $\boldsymbol{\epsilon\ S_{+i}, S_{-i}}$

$\boldsymbol{Q_i = S_{+i} + ((\min(S_{-i}) * sum(S_{-i}))/(S_{-i} * (sum(S_{-min}/S_{-i}))))}$

End for

3.2.5 Menentukan Metode Terbaik

Berdasarkan penjelasan dari SMAA-2, MOORA, dan COPRAS, metode-metode tersebut dipilih karena sudah terbukti untuk menyelesaikan masalah *multicriteria decision making*. Karena yang akan dioptimasi dalam penelitian ini adalah waktu dan biaya yang dibutuhkan untuk melakukan simulasi, yang dimana merupakan masalah yang memiliki kriteria lebih dari satu untuk optimasinya, maka metode-metode tersebut diyakini dapat menyelesaikan masalah yang ada.

Penelitian yang dilakukan sebelumnya (Lee & Chang, 2018) mencoba menentukan urutan alternatif terbaik yang didapat dengan melakukan agregasi pada hasil rangking metode-metode yang ada. Dalam penelitian tersebut, masing-masing

metode MCDM diberikan skenario berbeda, dimana bobot masing-masing kriteria dalam skenario tersebut akan diubah. Setiap perubahan rangking alternatif pada tiap skenario dari suatu metode akan dibandingkan dengan rangking alternatif saat bobot yang digunakan setara saat menggunakan metode yang sama. Hasil dari perbandingan ini adalah *sensitivity coefficient* yang menunjukkan sensitivitas dari metode tersebut. Metode yang memiliki *sensitivity coefficient* yang kecil dianggap sebagai metode yang memiliki sensitivitas rendah dan merupakan metode yang lebih stabil.

Selanjutnya, tiap alternatif dari semua metode pada masing-masing skenario akan diberi poin. Apabila rangking suatu alternatif dari k alternatif pada suatu skenario mendapatkan rangking terbaik, maka alternatif tersebut diberikan k poin. Untuk rangking kedua, alternatif tersebut diberikan $k-1$ poin, dan seterusnya. Hasil dari agregasi tersebut akan digunakan sebagai *ground truth*, untuk menentukan akurasi dari masing-masing metode. Dari hasil akurasi dan sensitivitas diatas, metode yang digunakan akan dibandingkan. Metode yang memiliki akurasi tertinggi dan sensitivitas terendah dianggap sebagai metode terbaik.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Lingkungan Uji Coba

Simulasi dijalankan menggunakan komputer dengan spesifikasi processor Intel® Core™ i5-8250U CPU @ 1.60 GHz (8 CPUs), memori 8 GB, sistem operasi yang digunakan adalah Windows 10 Home Single Language 64-bit dan bahasa pemrograman yang digunakan adalah Python dengan bantuan library SPADE untuk Agent Based Simulation. Untuk melakukan optimasi menggunakan SMAA-2, JSMAA-1.0.3 yang dapat diunduh di SMAA.fi/JSMAA. Sedangkan, untuk optimasi menggunakan MOORA dan COPRAS, Bahasa pemrograman yang digunakan adalah Python.

4.2 Simulasi Berbasis Agen

Langkah pertama untuk melakukan simulasi adalah membuat model simulasi. Model simulasi yang dibuat mengacu pada model simulasi pada penelitian sebelumnya seperti pada Gambar 2.6 sebelumnya. Selanjutnya, simulasi dikembangkan menggunakan SPADE. Simulasi akan dijalankan menggunakan *log data* yang didapatkan dari TPS Surabaya. Log yang akan digunakan dalam simulasi sebanyak 1100 *records*. Hal ini dilakukan karena keterbatasan mesin. *Log data* tersebut terdiri dai 4 *traces* yang akan disimulasikan.

Untuk menjalankan simulasi menggunakan SPADE, dibutuhkan server *Extensible Messsaging and Presence Protocol* (XMPP) sebagai fasilitator pengiriman *message*. Untuk server XMPP yang digunakan dalam penelitian ini adalah Openfire. Setiap agent dalam simulasi harus melakukan *login* pada server Openfire agar dapat mengirimkan *message*.

Dalam proses pengembangannya, setiap *agent* akan mewakili satu organisasi. Masing-masing *agent* mengerjakan *task* sesuai dengan yang dikerjakan oleh organisasi yang direpresentasikan. Karena simulasi berbasis agen merupakan simulasi yang asinkron, yang dimana setiap *task* yang diberikan pengerjaannya bergantung pada lama penyelesaian *task* sebelumnya, maka apabila sebuah agen

membutuhkan komunikasi dengan agen lain digunakanlah *message*. *Message* ini jugalah yang akan memicu pengejaan suatu *task* apabila *task* tersebut dikerjakan oleh *agent* yang berbeda.

Karena adanya proses asinkron ini, maka muncullah Asynchronous Waiting Time (AWT). Tabel 3 menunjukkan pseudocode untuk *sender* dari *message*.

Tabel 4.1 Pseudocode *sender*

Input: waktu mulai eksekusi *task* $\boldsymbol{t}$ **Output:** AWT $\boldsymbol{W_a}$
msg = Message(to="TPS@localhost") msg.body = "msg_from_CJODD" await self.send(msg)

Dari *message* yang dikirim oleh sebuah *agent* dengan menggunakan pseudocode yang ada pada Tabel 3, maka *agent* yang dituju oleh *agent* pengirim dapat menerima *message* tersebut. Table 4 menunjukkan peseudocode untuk *agent* yang menerima pesan dari *agent* yang mengirim *message*.

Tabel 4.2 Pseudocode receiver

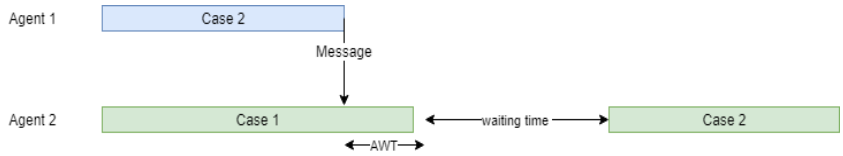
Input: waktu mulai eksekusi *task* $\boldsymbol{t}$ **Output:** AWT $\boldsymbol{W_a}$
$\boldsymbol{t}$ = time.time() msg = await self.receive (timeout=20) print("TPSAgent received message {msg.body}") awt=time.time()-$\boldsymbol{t}$

Untuk menghitung AWT dari aktivitas selanjutnya, pada pseudocode di Tabel 4 telah ditunjukkan perhitungannya. Nilai t merupakan waktu mulai *task* tersebut diberikan pada suatu *agent*. Selama *agent* tersebut belum menerima *message*, maka *agent* tersebut tidak akan bisa mengerjakan *task* yang diberikan.

Setelah message berhasil diterima, maka waktu saat diterimanya *message* dikurangi dengan waktu mulainya *agent* tersebut diberikan *task* akan dijadikan nilai AWT-nya.

4.3 Paralelisasi Agen

Hasil dari simulasi menunjukkan adanya AWT. Dari simulasi tersebut ditemukan bagaimana AWT terjadi. Gambar 4.1 menunjukkan terjadinya AWT secara detail.



Gambar 4.1 AWT kasus pertama

Pada Gambar 4.1 tersebut, *agent* 1 telah selesai melakukan suatu *task* dan mengirim *message* pada *agent* 2. Sementara itu, *agent* 2 masih mengerjakan suatu *task* dari *case* sebelumnya. Karena itu, ada AWT yang terjadi. Untuk menghitung AWT ini, digunakan Persamaan (19).

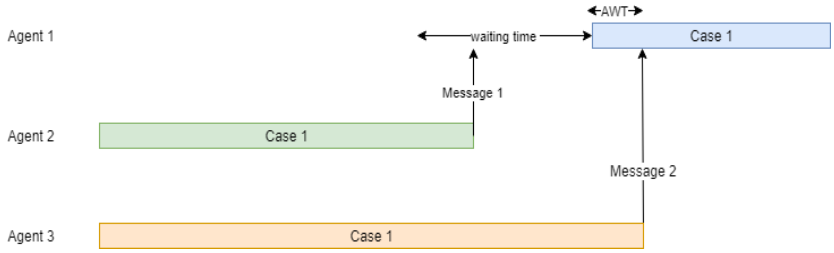
$$Wa_{mn} = (Ts_{mn} - W_{mn}) - Te_{mn-1} \qquad (19)$$

dimana,

Ts_{mn} = Start time dari *task* mn

Te_{mn-1} = End time dari *task* sebelum of *task* mn

AWT juga dapat terjadi saat simulasi ketika suatu *agent* sudah menerima *task* untuk dikerjakan namun masih belum menerima *message* dari *agent* yang mengerjakan *task* sebelumnya. Gambar 4.2 menunjukkan bagaimana AWT terjadi pada kasus yang lain.



Gambar 4.2 AWT kasus kedua

Gambar 4.2 menunjukkan bagaimana bila suatu *task* membutuhkan lebih dari satu *message* untuk mulai dijalankan. Untuk menghitung AWT pada kasus kedua, digunakan Persamaan (20).

$$Wa_{mn} = |Ts_{mn} - Te_{mn-1}| \qquad (20)$$

Selanjutnya, untuk melakukan paralelisasi, penelitian ini menggunakan distribusi *Least Work Left* (LWL) untuk memilih agent yang akan mengerjakan suatu *task*. *Agent* yang sedang mengerjakan suatu *task* dengan sisa waktu yang paling sedikit akan diberikan *task* baru untuk dikerjakan setelah selesai melakukan *task* yang sedang dikerjakannya.

Hasil simulasi dari paralelisasi ini menunjukkan bahwa paralelisasi berhasil mengurangi AWT yang terjadi. Total AWT dan biaya yang diperlukan saat melakukan simulasi dengan *agent* parallel dari 1-10 ditunjukkan pada Tabel 5.

Tabel 4.3 Hasil simulasi

Jumlah Paralel	**AWT**	**Biaya**
1	9300469.69	65977283.31
2	4564010.08	67010216.77
3	3035485.56	66320886.88
4	2194768.81	67574410.70
5	1743519.78	65790427.32
6	1502409.05	64163855.60

7	1269282.91	64451610.57
8	1109564.39	64745119.53
9	973846.38	66812484.07
10	870362.55	64088425.60

4.4 Optimasi Jumlah Agen

Dengan hasil simulasi yang ada pada Tabel 5, optimasi untuk menentukan jumlah *agent* parallel dapat dilakukan. Untuk melakukan optimasi, penelitian ini menggunakan metode *Multiple Criteria Decision Making* (MCDM). Metode-metode yang digunakan adalah SMAA-2, MOORA, dan COPRAS. Metode-metode ini akan dibandingkan untuk mencari metode terbaik dalam kasus ini dengan memberikan bobot pada tiap kriteria antara lain 1) 0,5 dan 0,5; 2) 0,1 dan 0,9; 3) 0,3 dan 0,7; 4) 0,7 dan 0,3; 5) 0,9 dan 0,1.

Untuk decision matriks yang digunakan dalam penelitian ini berdasarkan pada Tabel 4.3. Jumlah *agent* paralel pada Tabel 4.3 akan dianggap sebagai sebuah alternatif, sehingga terdapat 10 alternatif dalam penelitian kali ini. Gambar 4.3 menunjukkan *decision* matriks untuk penelitian ini.

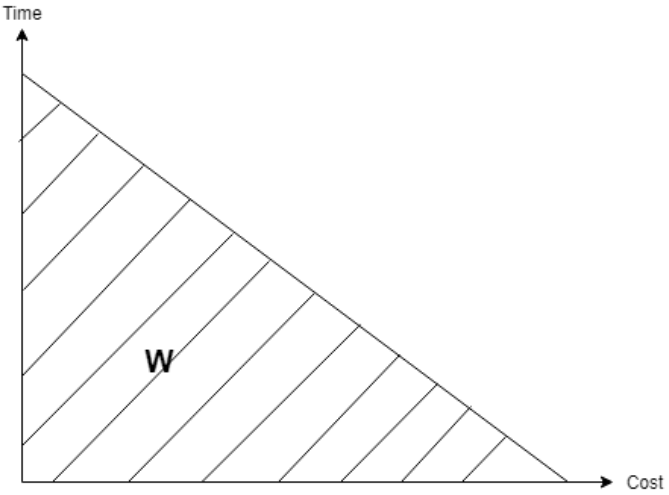
$$\begin{bmatrix} 9300469.69 & 65977283.31 \\ 4564010.08 & 67010216.77 \\ 3035485.56 & 66320886.88 \\ 2194768.81 & 67574410.70 \\ 1743519.78 & 65790427.32 \\ 1502409.05 & 64163855.60 \\ 1269282.91 & 64451610.57 \\ 1109564.39 & 64745119.53 \\ 973846.38 & 66812484.07 \\ 870362.55 & 64088425.60 \end{bmatrix}$$

Gambar 4.3 *Decision* matriks

Selanjutnya, Decision matriks pada Gambar 4.3 akan digunakan untuk masing-masing metode MCDM.

4.4.1 SMAA-2

SMAA-2 memiliki keunggulan dimana apabila pembuat keputusan tidak memiliki preferensi (bobot) terhadap sebuah alternatif, SMAA-2 akan memberikan kemungkinan sebuah alternatif untuk menempati sebuah peringkat. Langkah pertama dalam SMAA-2 ini adalah menentukan *weight*-nya. Apabila pengambul keputusan belum menentukan *weight* dari masing-masing kriteria, maka *weight* untuk metode ini dapat diambil dari FSW seperti pada Gambar 4.4.



Gambar 4.4 FSW

Dalam penelitian kali ini, saat hasil simulasi dioptimasi menggunakan SMAA-2, kemungkinan sebuah alternatif untuk menempati suatu peringkat berhasil dilakukan. Hasil dari optimasi tanpa menggunakan bobot ditampilkan pada Tabel 4.4 berikut.

Tabel 4.4 *Acceptability index* SMAA-2 tanpa bobot

Jumlah Agen	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
1	0	0	0	0	0	0,1131	0,079	0,1493	0,0056	0,653
2	0	0	0	0	0	0	0	0,0211	0,9789	0
3	0	0	0	0	0	0,2495	0,5349	0,2156	0	0
4	0	0	0	0	0	0	0,2156	0,4219	0,0155	0,347
5	0	0	0	0	0,7646	0,2354	0	0	0	0
6	0	0,751	0,0325	0,1404	0,0761	0	0	0	0	0

7	0	0,0687	0,882	0,0493	0	0	0	0	0	0
8	0	0,1548	0,0617	0,7835	0	0	0	0	0	0
9	0	0,0255	0,0238	0,0268	0,1593	0,402	0,1705	0,1921	0	0
10	1	0	0	0	0	0	0	0	0	0

Dari Tabel 4.4 dapat dilihat bahwa tanpa memberikan bobot SMAA-2 dapat menunjukkan *acceptability index* untuk memberikan rangking pada alternatif yang ada. Alternatif yang menggunakan 1 agen memiliki *acceptability index* 0,653 untuk peringkat 10. Sehingga, alternatif tersebut diberikan peringkat 10 karena memiliki *acceptability index* paling tinggi untuk peringkat tersebut. Alternatif 4 dengan *acceptability index* 0,345 memiliki kemungkinan untuk mendapat peringkat 10 dengan apabila bobot dari alternatifnya dirubah. Sebagai ringkasan, hasil dari SMAA dapat dilihat pada Tabel 4.5.

Tabel 4.5 Peringkat alternatif berdasarkan SMAA-2 (tanpa bobot)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	6	7	8	5	9	3	4	2	1
Acceptability index	1.000	0.751	0.882	0.784	0.765	0.402	0.535	0.422	0.979	0.653

Untuk membandingkan SMAA-2 dengan metode lain, amak setiap kriteria harus diberikan bobot. Untuk penelitian kali ini, *weight* untuk masing-masing kriteria telah ditentukan. Untuk yang pertama, tiap kriteria akan diberikan bobot 0,5. Selanjutnya, kita tentukan nilai fungsi $u(x_i, w)$ masing-masing alternatif seperti pada Persamaan 21.

$$u(x_1, w) = (9300469.69 * 0.5) + (65977283.31 * 0.5) = 32479394.08 \quad (21)$$

Untuk hasil lengkap dari semua alternatif, Tabel 4.6 menunjukkan hasil masing-masing nilai fungsinya.

Tabel 4.6 nilai fungsi masing-masing alternatif

Jumlah Paralel	$u(x_i, w)$
1	37638876.5
2	35787113.43
3	34678186.22
4	34884589.76
5	33766973.55
6	32833132.33
7	32860446.74
8	32927341.96
9	33893165.23
10	32479394.08

Nilai fungsi $u(x_i, w)$ pada Tabel 4.6 tersebut kemudian digunakan untuk memberikan peringkat pada tiap alternatif. Untuk memberikan peringkat pada tiap alternatif, digunakan Persamaan 6. Persamaan 22 menunjukkan bagaimana memberikan peringkat pada alternatif pada penelitian ini.

$$rank(i, x, w) = 1 + \sum_{k=1}^{10} \rho\big(u(x_k, w) > u(x_i, w)\big) \tag{22}$$

Persamaan 22 tersebut digunakan pada 10 alternatif yang ada, sehingga peringkat dari semua alternatif adalah seperti pada Tabel 4.7.

Tabel 4.7 Peringkat alternatif dan nilai fungsinya

Rank	**Jumlah Paralel**	$u(x_i, w)$
Rank I	10	37638876.5
Rank II	6	35787113.43
Rank III	7	34678186.22
Rank IV	8	34884589.76
Rank V	5	33766973.55
Rank VI	9	32833132.33
Rank VII	3	32860446.74
Rank VIII	4	32927341.96
Rank IX	2	33893165.23
Rank X	1	32479394.08

Dari Tabel 4.7, peringkat dari masing-masing alternatif dapat diketahui. Karena pada Tabel 4.7 tersebut telah ditentukan, maka *acceptability index* untuk masing-masing alternative adalah 1. Hasil dari optimasi menggunakan SMAA-2 dengan bobot 0,5 seperti yang terlihat pada Tabel 4.8 berikut.

Tabel 4.8 *Acceptability index* SMAA-2 dengan bobot 0,5 dan 0,5

Jumlah Agen	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	1	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0

8	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	1	0	0	0	0
10	1	0	0	0	0	0	0	0	0	0

Berdasarkan Tabel 4.8 dapat diketahui bahwa SMAA-2 akan memberikan nilai 1 pada tiap alternatif untuk menempati sebuah rangking apabila diberikan bobot. Tabel 4.9 menunjukkan peringkat yang didapat masing-masing alternatif dengan bobot kriteria 0,5 dan 0,5.

Tabel 4.9 Peringkat alternatif berdasarkan SMAA-2 (bobot 0,5 dan 0,5)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	6	7	8	5	9	3	4	2	1
Acceptability index	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Selanjutnya, untuk setiap skenario yang lain hasilnya akan ditunjukkan oleh Tabel 4.10, Tabel 4.11, Tabel 4.12, dan Tabel 4.13.

Tabel 4.10 Peringkat alternatif berdasarkan SMAA-2 (bobot 0,1 dan 0,9)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	6	7	8	5	1	3	9	2	4
Acceptability index	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Tabel 4.11 Peringkat alternatif berdasarkan SMAA-2 (bobot 0,3 dan 0,7)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	6	7	8	5	3	9	1	2	4
Acceptability index	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Tabel 4.12 Peringkat alternatif berdasarkan SMAA-2 (bobot 0,7 dan 0,3)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	6	7	8	5	9	3	4	2	1
Acceptability index	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Tabel 4.13 Peringkat alternatif berdasarkan SMAA-2 (bobot 0,9 dan 0,1)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	8	7	6	9	5	4	3	2	1
Acceptability index	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

4.4.2 MOORA

Dalam implementasi MOORA, bobot setiap kriteria harus ditentukan. Setelah menentukan bobot untuk tiap kriteria, maka kriteria-kriteria tersebut harus dibagi kedalam salah satu dari 2 kategori, *beneficial* atau *cost*. Karena *objective function* yang kita gunakan adalah minimisasi pada kedua kriteria, maka kedua kriteria yang ada pada Gambar 4.3 merupakan kriteria *cost*. Tabel 4.14 menunjukkan pembagian kriteria.

Tabel 4.14 Pembagian kategori kriteria

Beneficial	**Cost**
-	Asynchronous Waiting Time (AWT)
-	Biaya yang dibutuhkan (cost)

Hasil dari MOORA adalah *optimization value* Q_i. Nilai Q_i yang besar menunjukkan bahwa alternatif tersebut merupakan alternatif yang lebih baik

daripada alternatif dengan Q_i yang kecil. Untuk optimasi dengan MOORA penelitian ini dimulai menggunakan bobot 0,5 dan 0,5 untuk tiap kriteria. Untuk perhitungan menggunakan MOORA dimulai dengan menemukan denominator masing-masing kriteria. Untuk menghitung denominatornya, maka dilakukan perhitungan seperti pada Persamaan 23.

$$d_j = \sqrt{\sum_{i=1}^{10} (x_{ij})^2} \tag{23}$$

Tabel 4.15 menunjukkan hasil perhitungan denominatornya berdasarkan Persamaan 23.

Tabel 4.15 Denominator masing-masing kriteria pada MOORA

Kriteria	d_j
AWT	11454489.13
Cost	207775570.5

Setelah diketahui denominator masing-masing kriteria, maka nilai r_{ij} dapat dihitung seperti pada Persamaan 24.

$$r_{11} = \frac{x_{11}}{d_1} = \frac{9300469.69}{11454489.13} = 0.811949759 \tag{24}$$

Tabel 4.16 menunjukkan hasil perhitungan nilai r dari semua kriteria pada masing-masing alternatif.

Tabel 4.16 Hasil perhitungan *normalized performance* MOORA

Jumlah Paralel	r_1	r_2
1	0.811949759	0.3175411
2	0.398447284	0.322512491
3	0.265004011	0.319194825
4	0.191607743	0.325227891
5	0.152212793	0.316641784
6	0.13116334	0.308813281
7	0.110810958	0.310198212
8	0.096867209	0.311610837
9	0.085018753	0.321560826
10	0.075984406	0.308450245

Setelah normalized performance dari masing-masing kriteria untuk setiap alternatif diketahui, selanjutnya nilai optimisasi (Q_i) dapat diketahui menggunakan Persamaan 25.

$$Q_i = S_{+i} - S_{-i} = 0 - (0.5 * (0.811949759 + 0.3175411)) = -0.56474543 \quad (25)$$

Hasil dari MOORA dengan bobot kriteria masing-masing 0,5 setelah diberi peringkat berdasarkan nilai Q_i tersebut ditampilkan pada Tabel 4.17.

Tabel 4.17 Peringkat alternatif berdasarkan MOORA (bobot 0,5 dan 0,5)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	9	8	7	6	5	4	3	2	1
Q_i	-0.192	-0.203	-0.204	-0.211	-0.220	-0.234	-0.258	-0.292	-0.360	-0.565

Karena metode agregasi MOORA yang merupakan selisih dari kategori *beneficial* dan *cost* maka kategori yang anggotanya memiliki nilai lebih besar yang akan menentukan bagaimana hasil akhir dari optimasi tersebut. dalam kasus ini karena semua kriteria merupakan kriteria *cost* yang perlu diminimisasi, maka nilai Q_i MOORA menjadi nilai negatif. Selanjutnya, kita melakukan optimisasi dengan menggunakan bobot yang sama dengan yang diberikan pada SMAA-2. Hasil dari optimisasinya ditampilkan pada Tabel 4.18, Tabel 4.19, Tabel 4.20, dan Tabel 4.21.

Tabel 4.18 Peringkat alternatif berdasarkan MOORA (bobot 0,1 dan 0,9)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	8	7	6	9	5	4	3	2	1
Q_i	-0.285	-0.290	-0.290	-0.291	-0.298	-0.300	-0.312	-0.314	-0.330	-0.367

Tabel 4.19 Peringkat alternatif berdasarkan MOORA (bobot 0,3 dan 0,7)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	8	7	9	6	5	4	3	2	1
Q_i	-0.239	-0.247	-0.250	-0.251	-0.256	-0.267	-0.285	-0.303	-0.345	-0.465

Tabel 4.20 Peringkat alternatif berdasarkan MOORA (bobot 0,7 dan 0,3)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	9	8	7	6	5	4	3	2	1
Q_i	-0.146	-0.156	-0.161	-0.171	-0.184	-0.202	-0.232	-0.281	-0.376	-0.664

Tabel 4.21 Peringkat alternatif berdasarkan MOORA (bobot 0,9 dan 0,1)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	9	8	7	6	5	4	3	2	1
Q_i	-0.099	-0.109	-0.118	-0.131	-0.149	-0.169	-0.205	-0.270	-0.391	-0.763

4.4.3 COPRAS

COPRAS menggunakan metode agregasi yang lebih kompleks daripada MOORA. Namun hasil akhir agregasi COPRAS merupakan penjumlahan dari kedua kategori, sehingga hasil akhirnya tidak mencerminkan kategori mana yang lebih kuat. Langkah awal dari COPRAS sama dengan yang ada pada MOORA. Langkah pertma ayang harus dilakukan adalah memberikan kategori pada kriteria seperti pada Tabel 4.14. Selanjutnya, denominator untuk masing-masing kriteria pada metode ini dihitung seperti pada Persamaan 26.

$$d_j = \sum_{i=1}^{10} x_{ij} \tag{26}$$

Hasil perhitungan denominator untuk metode ini ditunjukkan pada Tabel 4.22.

Tabel 4.22 Denominator masing-masing kriteria pada COPRAS

Kriteria	d_j
AWT	26563719.19
Cost	656934720.4

Setelah denominator dari setiap kriteria diketahui, kita dapat menghitung *normalized performance* dari COPRAS menggunakan cara yang sama dengan MOORA seperti pada Persamaan 27.

$$r_{11} = \frac{x_{11}}{d_1} = \frac{9300469.69}{26563719.19} = 0.350119259 \quad (27)$$

Untuk hasil dari semua alternatif pada perhitungan normalized performance metode COPRAS seperti ditunjukkan pada Tabel 4.23.

Tabel 4.23 Hasil perhitungan *normalized performance* COPRAS

Jumlah Paralel	r_1	r_2
1	0.350119259	0.100432024
2	0.17181367	0.102004377
3	0.114271858	0.100955064
4	0.082622798	0.102863205
5	0.065635379	0.100147587
6	0.056558686	0.097671585
7	0.047782575	0.098109612
8	0.041769919	0.098556398
9	0.036660769	0.101703384
10	0.032765086	0.097556764

Selanjutnya, kita dapat menghitung nilai S_{-i} setelah nilai r dari masing masing kriteria diketahui. Karena semua kriteria masuk dalam kriteria *cost*, maka nilai S_{+i} adalah 0. Persamaan 28 menunjukkan perhitungan S_{-i}.

$$S_{-1} = \sum_{j \in \Omega_{min}} w_j r_{1j}$$

$$= 0.5 * (0.350119259) + 0.5 * (0.100432024) = 0.225275641 \quad (28)$$

Tabel 4.24 menunjukkan hasil perhitungan S_{-i} dari masing-masing alternatif.

Tabel 4.24 Nilai S_{-i} setiap alternatif

Jumlah Paralel	S_{-i}
1	0.225275641
2	0.136909023
3	0.107613461
4	0.092743002
5	0.082891483
6	0.077115136
7	0.072946094
8	0.070163158
9	0.069182077
10	0.065160925
$\sum_{i=1}^{m} S_{-i}$	1

Setelah nilai S_{+i} dan S_{-i} diketahui, maka perhitungan Q_i untuk metode ini dapat dilakukan. Persamaan 29 menunjukkan perhitungan dari nilai Q_i untuk metode COPRAS dengan bobot masing-masing kriteria 0,5.

$$Q_1 = S_{+1} + \frac{S_{-min}\sum_{i=1}^{m} S_{-i}}{S_{-1}\sum_{i=1}^{m} \frac{S_{-min}}{S_{-i}}}$$

$$= 0 + \frac{0.065160925 * 1}{0.225275641 * 7.468236782}$$

$$= 0.038730663$$

(29)

Nilai Q_i dari tiap alternatif menggunakan COPRAS menggunakan bobot kriteria 0,5 dan 0,5 ditunjukkan pada Tabel 4.25.

Tabel 4.25 Peringkat alternatif berdasarkan COPRAS (bobot 0,5 dan 0,5)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	9	8	7	6	5	4	3	2	1
Q_i	0.134	0.126	0.124	0.12	0.113	0.105	0.094	0.081	0.064	0.039

Selanjutnya, metode ini juga akan menggunakan bobot yang sama seperti yang digunakan oleh metode optimasi yang lain. Hasil optimasi dengan masing-masing bobot ditunjukkan pada Tabel 4.26, Tabel 4.27, Tabel 4.28, dan Tabel 4.29.

Tabel 4.26 Peringkat alternatif berdasarkan COPRAS (bobot 0,1 dan 0,9)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	9	8	7	6	5	4	3	2	1
Q_i	0.134	0.126	0.124	0.12	0.113	0.105	0.094	0.081	0.064	0.039

Tabel 4.27 Peringkat alternatif berdasarkan COPRAS (bobot 0,3 dan 0,7)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	9	8	7	6	5	4	3	2	1
Q_i	0.134	0.126	0.124	0.12	0.113	0.105	0.094	0.081	0.064	0.039

Tabel 4.28 Peringkat alternatif berdasarkan COPRAS (bobot 0,7 dan 0,3)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	9	8	7	6	5	4	3	2	1
Q_i	0.134	0.126	0.124	0.12	0.113	0.105	0.094	0.081	0.064	0.039

Tabel 4.29 Peringkat alternatif berdasarkan COPRAS (bobot 0,9 dan 0,1)

Rank	**Rank I**	**Rank II**	**Rank III**	**Rank IV**	**Rank V**	**Rank VI**	**Rank VII**	**Rank VIII**	**Rank IX**	**Rank X**
Jumlah Agen Paralel	10	9	8	7	6	5	4	3	2	1
Q_i	0.134	0.126	0.124	0.12	0.113	0.105	0.094	0.081	0.064	0.039

4.4.4 Menentukan Metode Terbaik

Dari hasil yang didapat semua metode, dapat kita simpulkan bahwa perubahan bobot kriteria akan mempengaruhi hasil optimasi. Hasil dari semua optimasi dapat dilihat pada Tabel 4.30.

Tabel 4.30 Hasil optimasi dari semua metode dengan beberapa skenario

Skenario	Metode	Rank									
		I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**IX**	**X**
Bobot Seimbang [0.5,0.5]	SMAA-2	10	6	7	8	5	9	3	4	2	1
	MOORA	10	9	8	7	6	5	4	3	2	1
	COPRAS	10	9	8	7	6	5	4	3	2	1
Cost [0.1,0.9]	SMAA-2	10	6	7	8	5	1	3	9	2	4
	MOORA	10	8	7	6	9	5	4	3	2	1
	COPRAS	10	8	7	6	9	5	4	3	2	1
Cost [0.3,0.7]	SMAA-2	10	6	7	8	5	3	9	1	2	4
	MOORA	10	8	7	9	6	5	4	3	2	1
	COPRAS	10	8	9	7	6	5	4	3	2	1
Time [0.7,0.3]	SMAA-2	10	6	7	8	5	9	3	4	2	1
	MOORA	10	9	8	7	6	5	4	3	2	1
	COPRAS	10	9	8	7	6	5	4	3	2	1
Time [0.9,0.1]	SMAA-2	10	8	7	6	9	5	4	3	2	1
	MOORA	10	9	8	7	6	5	4	3	2	1
	COPRAS	10	9	8	7	6	5	4	3	2	1

Dari Tabel 4.30, tiap alternatif diberikan peringkat yang berbeda pada skenario yang berbeda. Dari semua scenario yang ada, alternatif dengan jumlah agen parallel berjumlah 10 yang memiliki hasil paling baik dari seluruh metode dengan beberapa skenario. Tabel 4.31 menunjukkan perhitungan *sensitivity coefficient* dari semua metode.

Tabel 4.31. Hasil *sensitivity coefficient*

Metode	*Sensitivity Coefficient*			
	Cost [0.1,0.9]	Cost [0.3,0.7]	Time [0.7,0.3]	Time [0.9,0.1]
SMAA-2	3	4	0	6
MOORA	4	3	0	0
COPRAS	4	2	0	0

Dari Tabel 4.31 dapat disimpulkan bahwa COPRAS merupakan metode yang memiliki sensitivitas terendah dengan *sensitivity coefficient* sebesar 6. Lebih kecil dari MOORA dengan *sensitivity coefficient* 7 dan SMAA-2 sebesar 13. Hal ini disebabkan karena COPRAS menggunakan $\min(S_{-i})$ dalam perhitungannya. Dengan menggunakan nilai minimum dari tiap kriteria, apabila suatu kriteria memiliki nilai minimum yang lebih besar dari nilai minimum kriteria yang lain, maka hal ini akan mempengaruhi hasil optimasinya. Dengan begitu, bobot yang diberikan harus timpang ke kriteria dengan nilai minimum terendah agar hasil optimasinya berubah.

Selanjutnya, untuk menghitung akurasi harus dilakukan agregasi pada Tabel 4.30 diatas. Tabel 4.32 menunjukkan hasil agregasi untuk tiap alternatif dari masing-masing skenario.

Tabel 4.32 Agregasi nilai untuk menentukan urutan peringkat

Skenario	Rank									
	I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**IX**	**X**
Bobot Seimbang [0.5,0.5]	3	6	10	11	16	21	22	23	23	30
Cost [0.1,0.9]	7	6	10	9	16	23	24	25	15	30
Cost [0.3,0.7]	5	6	11	9	16	21	23	25	19	30
Time [0.7,0.3]	3	6	10	11	16	21	22	23	23	30
Time [0.9,0.1]	3	6	9	12	15	19	22	25	24	30

Berdasarkan hasil dari Tabel 4.32, dapat kita simpulkan bahwa urutan alternatif terbaik secara keseluruhan adalah seperti yang ditampilkan pada Tabel 4.33.

Tabel 4.33 Peringkat tiap alternatif dari agregasi semua metode

Skenario	Rank									
	I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**IX**	**X**
Bobot Seimbang [0.5,0.5]	10	9	8	7	6	5	4	3	2	1
Cost [0.1,0.9]	10	8	7	6	5	9	3	4	1	2
Cost [0.3,0.7]	10	8	7	6	9	5	3	4	2	1
Time [0.7,0.3]	10	9	8	7	6	5	4	3	2	1
Time [0.9,0.1]	10	8	9	7	6	5	4	3	2	1

Hasil dari Tabel 4.33 bisa dijadikan standard untuk menilai akurasi dari tiap metode. Tabel 4.34 menunjukkan perbandingan antara masing-masing metode dengan peringkat keseluruhan untuk setiap skenario.

Tabel 4.34 Perbandingan metode dengan hasil keseluruhan

Scenario	Metode	Rank										
		I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**IX**	**X**	
Hasil Keseluruhan		10	9	8	7	6	5	4	3	2	1	
Bobot Seimbang [0.5,0.5]	SMAA-2	10	6	7	8	5	9	3	4	2	1	3/10
	MOORA	10	9	8	7	6	5	4	3	2	1	10/10
	COPRAS	10	9	8	7	6	5	4	3	2	1	10/10

Hasil Keseluruhan		10	8	7	6	5	9	3	4	1	2	
Cost [0.1,0.9]	SMAA-2	10	6	7	8	5	1	3	9	2	4	4/10
	MOORA	10	8	7	6	9	5	4	3	2	1	6/10
	COPRAS	10	8	7	6	9	5	4	3	2	1	6/10

Hasil Keseluruhan		10	8	7	6	9	5	3	4	2	1	
Cost [0.3,0.7]	SMAA-2	10	6	7	8	5	3	9	1	2	4	3/10
	MOORA	10	8	7	9	6	5	4	3	2	1	6/10
	COPRAS	10	8	9	7	6	5	4	3	2	1	5/10

Hasil Keseluruhan		10	9	8	7	6	5	4	3	2	1	
Time [0.7,0.3]	SMAA-2	10	6	7	8	5	9	3	4	2	1	3/10
	MOORA	10	9	8	7	6	5	4	3	2	1	10/10
	COPRAS	10	9	8	7	6	5	4	3	2	1	10/10

Hasil Keseluruhan		10	8	9	7	6	5	4	3	2	1	
Time [0.9,0.1]	SMAA-2	10	8	7	6	9	5	4	3	2	1	7/10
	MOORA	10	9	8	7	6	5	4	3	2	1	8/10
	COPRAS	10	9	8	7	6	5	4	3	2	1	8/10

Dari Tabel 4.34 tersebut dapat kita hitung akurasi dari masing-masing metode. Tabel 4.35 menunjukkan perhitungan akurasi dari masing-masing metode.

Tabel 4.35 Perhitungan akurasi

Method	Total Benar	Hasil Akurasi
SMAA-2	20/50	40%
MOORA	40/50	80%
COPRAS	39/50	78%

Dari hasil pada Tabel 4.26 telah ditunjukkan perbandingan antara masing-masing metode dengan hasil keseluruhan yang didapat pada Tabel 4.25. Dari hasil tersebut MOORA memiliki tingkat akurasi mencapai 80%. Sedangkan COPRAS memiliki tingkat akurasi sebesar 78%. SMAA-2 memiliki tingkat akurasi yang paling rendah dengan 40%.

Karena hasil akurasi ini merupakan pengaruh dari agregasi, maka metode yang memiliki pendekatan yang hampir sama pada mayoritas metode memiliki peluang lebih besar untuk mendapatkan akurasi yang lebih besar. Karena itu,

MOORA dan COPRAS yang memiliki pendekatan yang hampir sama masing-masing memiliki akurasi yang hampir mendekati, yaitu 80% dan 78%. Sedangkan SMAA-2 yang merupakan metode yang memiliki pendekatan berbeda untuk memberikan peringkat pada alternatif memiliki akurasi yang paling rendah.

Berdasarkan hasil sensitivitas dan akurasi, COPRAS dan MOORA seimbang dengan *trade-off* pada kedua hasil tersebut. COPRAS memiliki sensitivitas yang lebih rendah sehingga lebih stabil, namun berkurang akurasinya. Berkebalikan dengan COPRAS, MOORA memiliki akurasi yang lebih tinggi, namun sensitivitasnya lebih besar. SMAA-2 merupakan metode yang paling tinggi sensitivitasnya dan memiliki akurasi yang paling rendah.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Untuk melakukan simulasi *asynchronous* dengan *Agent Based Simulation*, penelitian ini menggunakan Bahasa pemrograman Python dengan *library* SPADE. Model simulasi yang dibuat adalah berdasarkan pada SOP yang didapat dari PT. TPS Surabaya. Sedangkan, untuk menjalankan simulasi, digunakan log yang didapat dari PT TPS Surabaya.
2. Untuk mengukur *Asynchronous Waiting Time* (AWT) ada dua cara. Cara pertama adalah dengan menggunakan Persamaan 19 apabila seuatu aktivitas telah selesai namun *agent* yang bertindak sebagai penerima masih mengerjakan aktivitas yang lain. Cara kedua adalah dengan menggunakan persamaan 20 apabila suatu *agent* masih belum menerima pesan yang dibutuhkan untuk mengerjakan suatu *task*.
3. Untuk melakukan paralelisasi, tiap jenis agent akan diberikan jumlah agent parallel yang sama. Selanjutnya, tugas yang diberikan pada tiap agent berdasarkan pada metode *Least Work Left* (LWL). Dengan melakukan paralelisasi pada agen, maka AWT tersebut dapat dikurangi.
4. Dengan AWT dan biaya yang didapatkan, kami menggunakan metode metode *Multiple Criteria Decision Making* (MCDM) untuk menentukan jumlah agen paralel yang paling optimal. Kriteria yang digunakan dalam metode MCDM tersebut adalah AWT dan biaya total dari tiap hasil paralelisasi. Metode yang digunakan antara lain adalah SMAA-2, MOORA, dan COPRAS.
5. Hasil dari optimasi menunjukkan bahwa simulasi menggunakan 10 agen paralel memiliki hasil simulasi terbaik. Sedangkan, metode yang paling baik untuk digunakan bergantung pada keinginan dari pengambil keputusan. Apabila pengambil keputusan memilih untuk mengambil metode yang paling stabil, maka COPRAS lebih baik digunakan daripada metode lain dengan *sensitivity coefficient* terendah. Sedangka apabila bergantung pada

akurasi metode, maka MOORA dengan 80% akurasi lebih unggul dibandingkan metode COPRAS dengan 78% akurasi dan SMAA-2 dengan 40% akurasi.

5.2 Saran

1. Peneliti menyarankan untuk menggunakan lebih banyak *log data* dibandingkan dengan 1100 *records* yang digunakan untuk penelitian ini. Karena itu, dibutuhkan mesin yang lebih kuat. Dengan lebih banyak *log data* yang disimulasikan, akurasi dari metode MCDM mungkin dapat ditingkatkan.
2. Peneliti menyarankan menggunakan *log data* dengan jumlah *records* yang lebih banyak, karena akan menampilkan hasil simulasi yang lebih mendekati kenyataan.

DAFTAR PUSTAKA

Büth, L., Broderius, N., Herrmann, C., & Thiede, S. (2017). Introducing Agent-Based Simulation of Manufacturing System to Industrial Discrete-Event Simulation Tools. *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, 3–8. https://doi.org/10.1109/INDIN.2017.8104934

El Kholy, W., Bentahar, J., El Menshawy, M., Qu, H., & Dssouli, R. (2014). Modeling and verifying choreographed multi-agent-based web service compositions regulated by commitment protocols. *Expert Systems with Applications*, *41*(16), 7478–7494. https://doi.org/10.1016/j.eswa.2014.05.046

Fajar, A., Sarno, R., & Fauzan, A. C. (2018). Comparison of Discrete Event Simulation and Agent Based Simulation for Evaluating the Performance of Port Container Terminal. *2018 International Conference on Information and Communications Technology (ICOIACT)*, 259–265. https://doi.org/10.1109/ICOIACT.2018.8350717

Fauzan, A. C., Sarno, R., & Yaqin, M. A. (2017). Performance Measurement Based on Coloured Petri Net Simulation of Scalable Business Processes. *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*. https://doi.org/10.1109/EECSI.2017.8239121

Hyytiä, E., Righter, R., Bilenne, O., & Wu, X. (2017). Dispatching fixed-sized jobs with multiple deadlines to parallel heterogeneous servers. *Performance Evaluation*, *114*, 32–44. https://doi.org/10.1016/j.peva.2017.04.003

Ivashkin, Y. A., & Nazoikin, E. A. (2009). Agent-based simulation model of educational process in the student group. *CSSim 2009 - 1st International Conference on Computational Intelligence, Modelling, and Simulation*, 132–137. https://doi.org/10.1109/CSSim.2009.29

Kaklauskas, A., Zavadskas, E. K., & Raslanas, S. (2005). Multivariant design and multiple criteria analysis of building refurbishments. *Energy and Buildings*, *37*(4), 361–372. https://doi.org/10.1016/j.enbuild.2004.07.005

Kaklauskas, A., Zavadskas, E. K., Raslanas, S., Ginevicius, R., Komka, A., & Malinauskas, P. (2006). Selection of low-e windows in retrofit of public buildings by applying multiple criteria method COPRAS: A Lithuanian case. *Energy and Buildings*, *38*(5), 454–462. https://doi.org/10.1016/j.enbuild.2005.08.005

Karande, P., & Chakraborty, S. (2012). Application of multi-objective optimization on the basis of ratio analysis (MOORA) method for materials selection. *JOURNAL OF MATERIALS&DESIGN*, *37*, 317–324. https://doi.org/10.1016/j.matdes.2012.01.013

Lahdelma, Risto; Salminen, P. (2001). Smaa-2 : Stochastic Multicriteria Analysis

for Group Decision Making. *Operational Research*, *49*(3), 444–454.

Lee, H., & Chang, C. (2018). Comparative analysis of MCDM methods for ranking renewable energy sources in Taiwan. *Renewable and Sustainable Energy Reviews*, *92*(May), 883–896. https://doi.org/10.1016/j.rser.2018.05.007

Milanovi, M. (2010). Modeling Service Choreographies with Rule-enhanced Business Processes. *2010 14th IEEE International Enterprise Distributed Object Computing Conference*, 194–203. https://doi.org/10.1109/EDOC.2010.18

Mohammed, A., Wang, Q., & Filip, M. (2017). Towards a cost-effective design of a meat supply chain: A multi-criteria optimization model. *ICAC 2017 - 2017 23rd IEEE International Conference on Automation and Computing: Addressing Global Challenges through Automation and Computing*, (September), 7–8. https://doi.org/10.23919/IConAC.2017.8082016

Parsa, S., Ebrahimifard, A., Amiri, M. J., & Arani, M. K. (2016). Towards a goal-driven method for web service choreography validation. *2016 2nd International Conference on Web Research, ICWR 2016*, 66–71. https://doi.org/10.1109/ICWR.2016.7498448

Peltz, C. (2003). Real-time web services orchestration and choreography. *CEUR Workshop Proceedings*. https://doi.org/10.1109/MC.2003.1236471

Rozinat, A., Mans, R. S., Song, M., & van der Aalst, W. M. P. (2008). Discovering colored Petri nets from event logs. *International Journal on Software Tools for Technology Transfer*, *10*(1). https://doi.org/10.1007/s10009-007-0051-0

Stanujkic, D., Đorđević, B., & Đorđević, M. (2013). COMPARATIVE ANALYSIS OF SOME PROMINENT MCDM METHODS : A CASE OF RANKING SERBIAN BANKS, *8*(2), 213–241. https://doi.org/10.5937/sjm8-3774

Tervonen, T., & Lahdelma, R. (2007). Implementing stochastic multicriteria acceptability analysis. *European Journal of Operational Research*, *178*(2), 500–513. https://doi.org/10.1016/j.ejor.2005.12.037

Wang, C., Ge, J., & Xu, X. (2009). Analysis of air traffic flow control through agent-based modeling and simulation. *Proceedings - 2009 International Conference on Computer Modeling and Simulation, ICCMS 2009*, 286–290. https://doi.org/10.1109/ICCMS.2009.38

Wang, J., Zhu, B., Wang, Y., & Huang, L. (2016). Mining organizational behaviors in collaborative logistics chain: An empirical study in a port. *2016 International Conference on Logistics, Informatics and Service Sciences (LISS)*, 1–5. https://doi.org/10.1109/LISS.2016.7854453

Yu, B., Cui, L., & Lu, X. (2016). Research on the Logistics Efficiency by and Simulation of Tianjin Port. *2016 International Conference on Logistics, Informatics and Service Sciences (LISS)*, 0–4. https://doi.org/ 10.1109/LISS.2016.7854440

Zouari, Z., & Khayech, K. (2011). Performance evaluation of port logistics: The case of the Sousse port. *2011 4th International Conference on Logistics, LOGISTIQUA'2011*,241–247. https://doi.org/10.1109/LOGISTIQUA.2011.5939297

[Halaman ini sengaja dikosongkan]

BIOGRAFI PENULIS

Aziz Fajar. Anak kedua dari tiga bersaudara, lahir pada tanggal 5 Oktober 1994 di Malang, Jawa Timur dari pasangan M. Farid dan Wiwik Azizah. Penulis menjalani pendidikan formal di TK Wahid Hasyim (1999), SDI Wahid Hasyim (2006), MtsN Malang 1 (2009), MAN 3 Malang (2012), S-1 Teknik Informatika di Universitas Islam Negeri Maulana Malik Ibrahim Malang (2017) dan menempuh pendidikan S-2 Teknik Informatika di Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2017. Penulis dapat dihubungi melalui telepon/whatapps di nomor 085755995700 atau email azizfajar510@gmail.com.