Perancangan Simulasi *Risk Pooling* untuk Mendukung Proses Pembelajaran

Michael Gilbert Thimandalle[1], I Gede Agus Widyadana[2]

Abstract: This research was conducted to provide new innovations in the learning process activities of the Supply Chain Management course. By using simulation games, the learning process will run more interestingly so it is hoped that the knowledge taught can be absorbed by students optimally. The risk pooling material is one of many materials whose teaching will be more optimal with the existence of simulation games. This method compares the decentralized and centralized supply chain systems. In decentralized supply chain there are 3 stages and in centralized supply chain there are 2 stages. The simulation game was created using the AnyLogic application. The result of this research is in the form of a simulation game that describes a decentralized and centralized supply chain system.

Keywords: risk pooling, simulation games, AnyLogic

Pendahuluan

Manajemen rantai pasok merupakan salah satu mata kuliah yang diajarkan di Program Studi Teknik Industri. Manajemen rantai pasok ini sendiri merupakan kegiatan yang memproses segala kegiatan dan informasi yang berkaitan dengan pengadaan bahan baku, pengolahan bahan baku hingga menjadi barang jadi, dan pendistribusian produk akhir tersebut kepada konsumen (Martono [1]). Mata kuliah Manajemen Rantai Pasok merupakan mata kuliah yang sangat menarik dan sangat berguna bagi mahasiswa untuk mempersiapkan diri memasuki dunia pekerjaan. Maka dari itu diperlukan adanya inovasi-inovasi terbaru dalam pengajaran dari tiap materi agar lebih efektif untuk diserap oleh para mahasiswa. Hal ini tentu sangat dibutuhkan terlebih lagi di masa pembelajaran *daring* yang berlangsung hingga saat ini. Salah satu inovasinya adalah pengajaran dengan menggunakan *game* simulasi. Ada banyak contoh nyata *game* simulasi yang dapat membantu pembelajaran. Salah satu contohnya adalah *game* yang bernama *Dystopian City* (Padilla [2]). Hasil dari penelitian ini sendiri adalah berupa simulasi distribusi dengan metode *risk pooling* melalui aplikasi *anylogic*. Diharapkan dengan adanya simulasi ini, pengajaran mengenai metode *risk pooling* pada mata kuliah Manajemen Rantai Pasok untuk ajaran baru akan lebih menarik dan interaktif.

[1,2] Fakultas Teknologi Industri, Program Studi Teknik Industri, Universitas Kristen Petra. Jl. Siwalankerto 121-131, Surabaya 60236. Email: michaelthimandalle@gmail.com, gede@petra.ac.id

Metode Penelitian

Metode penelitian merupakan langkah-langkah atau tahapan yang akan dilakukan dalam sebuah penelitian. Penelitian ini dilakukan dengan melalui beberapa langkah-langkah. Berikut adalah langkah-langkah dalam metode penelitian.

Desain Skenario

Dalam pembuatan *game* simulasi ini diperlukan skenario sebagai langkah awal dalam pembuatannya. Skenario pertama menggambarkan kondisi awal dimana terdapat 3 titik distribusi yang memiliki gudang masing-masing. Hal ini tentu akan mengakibatkan tingginya jumlah *safety stock* yang harus disediakan di masing-masing. Semakin tinggi jumlah *safety stock* yang ada maka semakin tinggi pula *cost* yang harus dikeluarkan. Adapun skenario kedua yang menggambarkan penerapan metode *risk pooling* dimana hanya terdapat 1 gudang sebagai *Distribution Center* untuk memenuhi 3 titik distribusi tersebut (Nadeem [3]).

Tugas dari pemain adalah mengatur jumlah persediaan pada *Distribution Center* dan jumlah pengiriman untuk masing-masing titik distribusi dengan memerhatikan *demand* dan *lead time* tiap titik distribusi. Tujuan dari *game* simulasi ini adalah untuk meminimumkan *cost* yang dikeluarkan. Semakin rendah *cost* yang dikeluarkan maka akan semakin baik.

Studi Literatur

Studi literatur dilakukan untuk menambah ilmu dan wawasan yang diperlukan dalam melakukan

pembuatan *game* simulasi ini. Referensi yang digunakan berasal dari buku, jurnal, dan artikel-artikel di situs internet. Pencarian referensi tersebut haruslah relevan dan dapat dibuktikan kebenarannya sehingga diperlukan adanya penyaringan dalam penetapan referensi.

Pembangunan Konsep Simulasi

Konsep dalam simulasi sangatlah penting dikarenakan hal tersebut dapat dijadikan standar dalam pembangunan simulasi. Dengan adanya standar maka dapat mempermudah dalam pembuatan *game* simulasi nantinya. Konsep disini merupakan pengembangan dari skenario awal yang telah ditetapkan. Kriteria-kriteria yang awalnya hanya gambaran umum dikembangkan menjadi lebih spesifik.

Pembangunan Simulasi

Pembangunan simulasi merupakan tahapan untuk membuat *game* simulasi yang telah direncanakan. *Game* simulasi yang dibuat haruslah berpedoman pada konsep yang telah ditentukan. Hal ini bertujuan agar *game* simulasi yang diciptakan mampu menjawab permasalahan yang ada.

Verifikasi Simulasi

Tahapan verifikasi merupakan tahapan yang dilakukan setelah simulasi telah tercipta. Tahapan verifikasi adalah tahapan pemeriksaan apakah simulasi yang diciptakan mampu berjalan dengan baik tanpa adanya kesalahan. Tahapan ini dilakukan dengan mengumpulkan umpan balik dari pengguna untuk menguji ketepatan dari hasil pemrosesan yang dihasilkan oleh simulasi. Jika hasil pemrosesan yang dihasilkan tidak sesuai maka diperlukan perbaikan dan dilakukan pembangunan simulasi kembali.

Uji Coba Simulasi

Uji coba simulasi dilakukan untuk menguji apakah simulasi sudah memenuhi konsep yang telah ditentukan sebelumnya. *Game* simulasi diuji dari semua aspek berdasarkan kriteria yang ditetapkan. Aspek-aspek tersebut antara lain ketepatan dengan kondisi nyata dari permasalah yang berkaitan dengan metode *risk pooling*, kemudahan dalam penggunaan, menarik secara estetika, dan lain-lain. Hal tersebut dinilai melalui pengumpulan umpan balik dari pengguna.

Modifikasi Simulasi

Modifikasi dilakukan apabila dijumpai adanya ketidaksesuaian yang ditemukan oleh pengguna. Hal ini tentu diperlukan agar *game* simulasi yang diciptakan dapat menjawab permasalahan dengan baik. Jika tahapan modifikasi diperlukan, maka tindakan yang dilakukan adalah kembali melakukan pembangunan konsep simulasi. Hal tersebut dilakukan terus-menerus hingga mendapatkan *game* simulasi yang terbaik.

Kesimpulan dan Saran

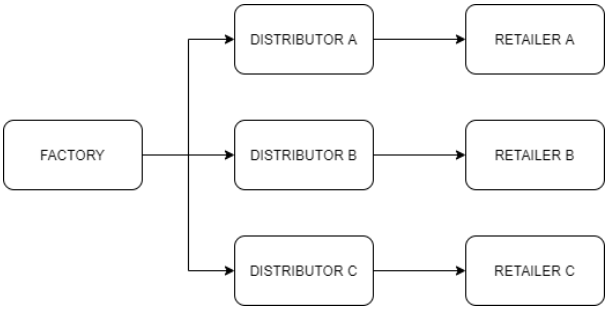
Kesimpulan yang diambil merupakan perwujudan akhir dari *game* simulasi itu sendiri. Kesimpulan tersebut berkaitan dengan kemampuan *game* simulasi dalam memenuhi kriteria-kriteria yang telah ditetapkan. Kesimpulan tersebut juga dapat dijadikan acuan dalam pemberian saran bagi penelitian serupa di kemudian hari.

Hasil dan Pembahasan

Pembangunan Konsep *Game* Simulasi

Pembangunan konsep game simulasi merupakan langkah penting yang harus dilakukan. Hal tersebut dikarenakan agar memudahkan dalam proses pembuatan simulasi. Pembangunan konsep diawali dengan penentuan komponen-komponen apa saja yang terdapat pada simulasi. Pembuatan kosnep ini terbagi menjadi dua konsep yaitu game simulasi decentralized dan centralized.

Konsep *Game* Simulasi *Decentralized*



Gambar 1. Konsep simulasi *decentralized*

Pada *game* simulasi *decentralized,* terdapat tujuh komponen utama yaitu *factory* , *retailer A, retailer B, retailer C, distributor A, distributor B,* dan *distributor C.* . Pemain diperbolehkan untuk menjadi *factory, distributor,* dan *retailer* maupun ke-tujuhnya sekaligus tergantung jumlah pemain.

Tujuan dari *game* ini adalah pemain dituntut untuk menjalankan suatu rantai pasok agar dapat berjalan dengan efektif dan efisien. Pemain haruslah mampu menjaga persediaan barang agar dapat terus memenuhi *demand* yang ada. Di lain sisi, pemain juga harus memperhatikan biaya yang dikeluarkan

untuk menyimpan persediaan (*inventory cost*) dan *backorder* yang terjadi akibat adanya kekurangan *stock* terhadap *demand* yang ada.

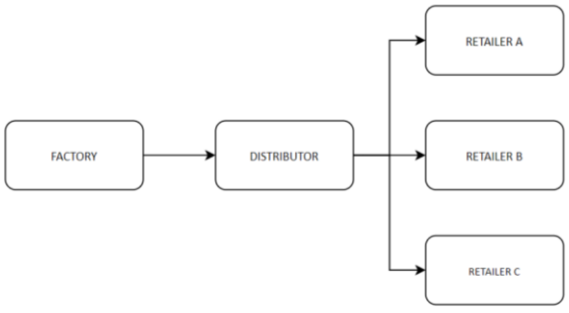
Game ini akan berjalan selama 100 hari. Di tiap harinya terdapat *demand* yang muncul pada setiap *retailer*. Maka dari itu lambat laun persediaan akan semakin berkurang. Untuk menanggulangi hal tersebut, baik *retailer*, *distributor,* dan *factory* akan menetapkan tingkat persediaan minimum dan maksimum guna menjaga tingkat persediaan.

Jika tingkat persediaan di *retailer* sudah mencapai minimum, maka *retailer* akan memesan kepada *distributor* sebesar selisih batas persediaan maksimum dikurangi jumlah persediaan saat itu. Hal tersebut juga berlaku pada *distributor* tetapi akan memesan kepada *factory*.

Factory akan mengirimkan ke *distributor* sejumlah permintaan yang diminta. Jika tingkat persediaan di *factory* sudah mencapai batas minimum, maka *factory* akan melakukan produksi sebesar selisih batas persediaan maksimum dikurangi jumlah persediaan.

Lead time pemesanan dari *retailer* ke *distributor* maupun dari *distributor* ke *factory* adalah selama satu minggu. Jika *retailer*, *distributor,* ataupun *factory* tidak mampu memenuhi permintaan maka dianggap sebagai *backorder* dan harus dipenuhi pada periode berikutnya. *factory* dan *retailer* dikenai biaya persediaan dan biaya *backorder* dengan tingkat biaya tertentu. Semakin efektif dan efisien rantai pasok berjalan, maka semakin baik.

Konsep *Game* Simulasi *Centralized*



Gambar 2. Konsep simulasi *centralized*

Pada *game* simulasi *centralized,* terdapat lima komponen utama yaitu *factory, distributor* , *retailer A, retailer B, retailer C*. Pemain diperbolehkan untuk menjadi *factory, distributor* dan *retailer* maupun kelimanya sekaligus tergantung jumlah pemain. Tujuan dari *game* ini adalah pemain dituntut untuk menjalankan suatu rantai pasok agar dapat berjalan dengan efektif dan efisien. Pemain haruslah mampu menjaga persediaan barang agar dapat terus memenuhi *demand* yang ada. Di lain sisi, pemain juga harus memperhatikan biaya yang dikeluarkan untuk menyimpan persediaan (*inventory cost*) dan *backorder* yang terjadi akibat adanya kekurangan *stock* terhadap *demand* yang ada.

Game ini akan berjalan selama 100 hari. Di tiap harinya terdapat *demand* yang muncul pada setiap *retailer*. Maka dari itu lambat laun persediaan akan semakin berkurang. Untuk menanggulangi hal tersebut, baik *retailer*, *distributor,* dan *factory* akan menetapkan tingkat persediaan minimum dan maksimum guna menjaga tingkat persediaan.

Jika tingkat persediaan di *retailer* sudah mencapai minimum, maka *retailer* akan memesan kepada *distributor* sebesar selisih batas persediaan maksimum dikurangi jumlah persediaan saat itu. Hal tersebut juga berlaku pada *distributor* tetapi akan memesan kepada *factory*. *Factory* akan mengirimkan ke *distributor* sejumlah permintaan yang diminta. Jika tingkat persediaan di *factory* sudah mencapai batas minimum, maka *factory* akan melakukan produksi sebesar selisih batas persediaan maksimum dikurangi jumlah persediaan.

Lead time pemesanan dari *retailer* ke *distributor* maupun dari *distributor* ke *factory* adalah selama satu minggu. Jika *retailer*, *distributor,* ataupun *factory* tidak mampu memenuhi permintaan maka dianggap sebagai *backorder* dan harus dipenuhi pada periode berikutnya. *factory* dan *retailer* dikenai biaya persediaan dan biaya *backorder* dengan tingkat biaya tertentu. Semakin efektif dan efisien rantai pasok berjalan, maka semakin baik.

Skenario *Game* Simulasi *Decentralized*

Game simulasi *decentralized* menggambarkan suatu rantai pasok yang terdiri dari tiga *retailer,* tiga *distributor*, dan *factory*. Oleh karena itu *game* simulasi ini dapat dimainkan maksimal tujuh pemain. Pemain dapat memilih *role* yang tersedia. Jika pemain kurang dari tujuh, maka *role* yang tidak dipilih pemain akan berjalan secara otomatis oleh sistem.

Tujuan akhir dari *game* simulasi ini adalah untuk mendapatkan biaya seminimum mungkin. Hal itu dapat tercapai berdasarkan strategi dari pemain. Sebelum memulai permainan, pemain diharuskan untuk menetapkan kondisi awal berjalannya rantai pasok. Seperti dijelaskan sebelumnya, terdapat tujuh *role* yang tersedia yaitu *"RETAILER A", "RETAILER B", "RETAILER C", "DISTRIBUTOR A",*

"DISTRIBUTOR B", "DISTRIBUTOR C", dan "*FACTORY*". Pemain dapat memilih *role-role* tersebut dengan mengklik *text* dari masing-masing *role.*

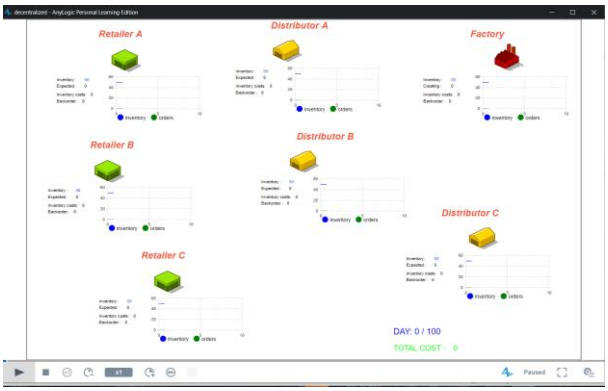
Gambar 3. Tampilan awal simulasi *decentralized*

Gambar 4. Tampilan awal simulasi *decentralized* ketika *"Retailer A"* dimainkan

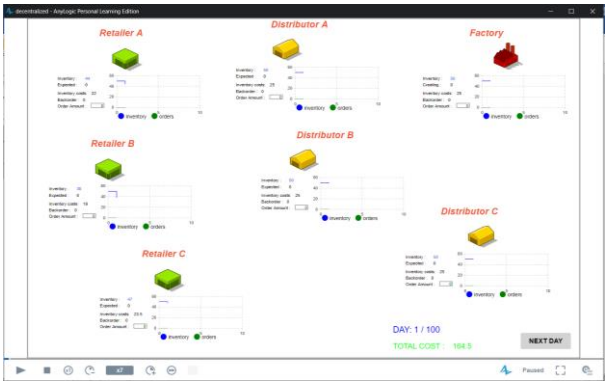
Jika pemain memainkan suatu *role* maka pemain hanya dapat menentukan jumlah inventori awal pada *role* tersebut. Hal ini juga berlaku jika pemain memilih *role* yang lainnya. Jika terdapat *role* yang tidak dipilih maka *role* tersebut akan berjalan otomatis melalui sistem.

Pemain dapat menentukan jumlah maksimal dan minimum dari inventori pada *role* yang tidak dimainkan. Hal ini berguna untuk membantu sistem dalam menjalankan permainan. Untuk pemain yang memilih *role* sebagai "*FACTORY"* maka pemain juga harus menentukan jumlah produksi per hari.

Selanjutnya terdapat biaya penyimpanan dan *backorder* yang dapat ditentukan pemain. Biaya penyimpanan muncul ketika terdapat sejumlah barang pada inventori di setiap *role. Backorder* terjadi ketika jumlah inventori tidak mampu memenuhi *demand* yang ada. Maka dari itu biaya *backorder* muncul ketika terjadi *backorder*. Jika semua kondisi telah ditetapkan maka *game* simulasi dapat dijalankan. *Game* simulasi akan berjalan dengan diawali masuknya *demand* pada *role retailer*. *Game* ini akan dijalankan dengan satuan hari selama seratus hari. Selanjutnya pemain yang memilih *role retailer* akan ditugaskan untuk menjaga tingkat inventorinya untuk menghindari terjadinya *backorder*.



Gambar 5. Tampilan simulasi *decentralized* dijalankan otomatis



Gambar 6. Tampilan simulasi *decentralized* dijalankan manual

Tingkat inventori dapat dilihat di bagian "*Inventory"*. Untuk melakukan pemesanan pada *factory* maka yang harus dilakukan adalah dengan mengisi jumlah order pada *box "Order Amount"*. Jumlah *order* yang sudah dipesan dapat dilihat pada bagian "*Expected"*. Hal yang berperlu diperhatikan dalam memainkan *game* simulasi ini adalah adanya biaya penyimpanan, biaya *backorder,* dan lamanya waktu pemesanan (*lead time*). Biaya penyimpanan dan *backorder* didapatkan dari perhitungan jumlah inventori yang ada maupun jumlah *backorder* dikalikan dengan biaya penyimpanan dan *backorder* yang telah ditentukan sebelumnya.

Pada *game* simulasi ini telah ditentukan *lead time* pemesanan yaitu selama empat belas hari atau dua minggu. Untuk berpindah hari maka yang harus dilakukan adalah dengan mengklik tombol dengan tulisan "*Next Day"*. Setelah berganti hari maka *order* pada *role retailer* akan diteruskan pada *role factory*. Pemain yang memilih *role factory* akan menentukan

jumlah produksi dengan memperhatikan jumlah produksi per hari yang telah ditentukan sebelumnya. Pada *role factory* juga berlaku biaya penyimpanan dan biaya *backorder*.

Jika simulasi berjalan dengan otomatis maka kegiatan *order* dan produksi akan berjalan berdasarkan strategi maksimum-minimum berdasarkan jumlah inventori maksimum dan minimum yang telah ditentukan sebelumnya. Strategi ini tentunya bertujuan agar simulasi dapat berjalan dengan efektif dan efisien. Simulasi ini akan berakhir pada hari ke-seratus. Diakhir akan didapatkan biaya total yang muncul selama simulasi berlangsung. Semakin minimum biaya total yang muncul maka akan semakin baik.

Skenario *Game* Simulasi *Decentralized*

Game simulasi *centralized* menggambarkan suatu rantai pasok yang terdiri dari tiga *retailer* , *distributor* dan *factory*. Oleh karena itu *game* simulasi ini dapat dimainkan maksimal lima pemain. Jika pemain kurang dari lima, maka *role* yang tidak dipilih pemain akan berjalan secara otomatis oleh sistem. Sistem disini berjalan berdasarkan rancangan yang telah dibuat melalui *AnyLogic*.

Gambar 7. Tampilan awal simulasi *centralized*

Gambar 8. Tampilan awal simulasi *centralized* ketika *"Retailer A"* dimainkan

Tujuan akhir dari *game* simulasi ini adalah untuk mendapatkan biaya seminimum mungkin. Hal itu dapat tercapai berdasarkan strategi dari pemain. Sebelum memulai permainan, pemain diharuskan untuk menetapkan kondisi awal berjalannya rantai pasok.

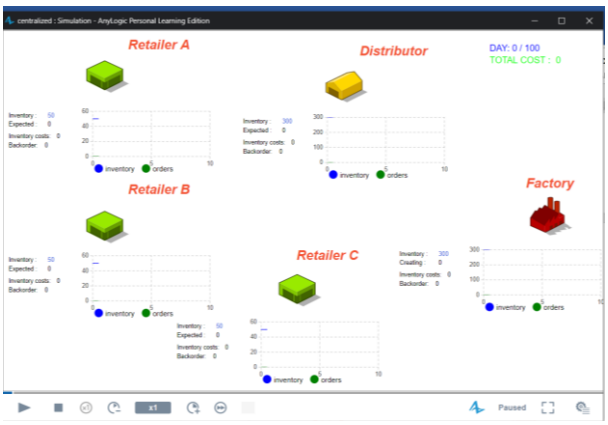
Seperti dijelaskan sebelumnya, terdapat lima *role* yang tersedia yaitu *"RETAILER A", "RETAILER B", "RETAILER C", "DISTRIBUTOR",* dan "*FACTORY*". Pemain dapat memilih *role-role* tersebut dengan mengklik *text* yang tertera dari masing-masing *role*.

Jika pemain memainkan suatu *role* maka pemain hanya dapat menentukan jumlah inventori awal pada *role* tersebut. Hal ini juga berlaku jika pemain memilih *role* yang lainnya.

Jika terdapat *role* yang tidak dipilih maka *role* tersebut akan berjalan otomatis melalui sistem. Pemain dapat menentukan jumlah maksimal dan minimum dari inventori pada *role* yang tidak dimainkan. Hal ini berguna untuk membantu sistem dalam menjalankan permainan. Untuk pemain yang memilih *role* sebagai "*FACTORY"* maka pemain juga harus menentukan jumlah produksi per hari.

Selanjutnya terdapat biaya penyimpanan dan *backorder* yang dapat ditentukan pemain. Biaya penyimpanan muncul ketika terdapat sejumlah barang pada inventori di setiap *role*. *Backorder* terjadi ketika jumlah inventori tidak mampu memenuhi *demand* yang ada. Selain itu, jika terdapat *backorder* maka *backorder* tersebut haruslah dipenuhi pada periode-periode selanjutnya.

Maka dari itu biaya *backorder* muncul ketika terjadi *backorder*. Jika semua kondisi telah ditetapkan maka *game* simulasi dapat dijalankan. *Game* simulasi dapat dijalankan dengan mengklik tombol "*run"* yang terdapat di pojok kanan bawah.



Gambar 9. Tampilan simulasi *centralized* dijalankan otomatis

Gambar 10. Tampilan simulasi *centralized* dijalankan manual

Game simulasi akan berjalan dengan diawali masuknya *order* pada *role retailer*. *Game* ini akan berlangsung dengan satuan hari selama seratus hari. Selanjutnya pemain yang memilih *role retailer* akan ditugaskan untuk menjaga tingkat inventorinya untuk menghindari terjadinya *backorder*. Tingkat inventori dapat dilihat di bagian "*Inventory*". Untuk melakukan pemesanan pada *factory* maka yang harus dilakukan adalah dengan mengisi jumlah order pada *box "Order Amount"*. Jumlah *order* yang sudah dipesan dapat dilihat pada bagian "*Expected*". Hal yang berperlu diperhatikan dalam memainkan *game* simulasi ini adalah adanya biaya penyimpanan, biaya *backorder,* dan lamanya waktu pemesanan (*lead time*). Biaya penyimpanan dan *backorder* didapatkan dari perhitungan jumlah inventori yang ada maupun jumlah *backorder* dikalikan dengan biaya penyimpanan dan *backorder* yang telah ditentukan sebelumnya.

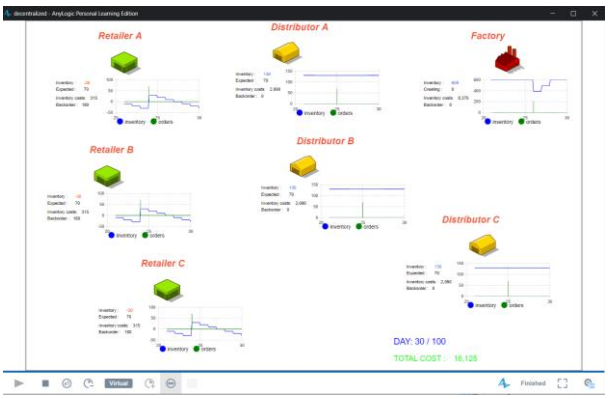
Pada *game* simulasi ini telah ditentukan *lead time* pemesanan yaitu selama tujuh hari/satu minggu. Untuk berpindah hari maka yang harus dilakukan adalah dengan mengklik tombol dengan tulisan "*Next Day*". Setelah berganti hari maka *order* pada *role retailer* akan diteruskan pada *role distributor*. Tugas yang dilakukan oleh pemain dengan *role distributor* cukup serupa dengan *role retailer*. Hal yang berbeda hanyalah *demand* yang berbeda. Selanjutnya *order* dari *role distributor* akan diteruskan pada *role factory*. Pemain yang memilih *role factory* akan menentukan jumlah produksi dengan memperhatikan jumlah produksi per hari yang telah ditentukan sebelumnya. Pada *role factory* juga berlaku biaya penyimpanan dan biaya *backorder*.

Jika simulasi berjalan dengan otomatis maka kegiatan *order* dan produksi akan berjalan berdasarkan strategi maksimum-minimum berdasarkan jumlah inventori maksimum dan minimum yang telah ditentukan sebelumnya. Strategi ini tentunya bertujuan agar simulasi dapat berjalan dengan efektif dan efisien. Simulasi ini akan berakhir pada hari ke-seratus.

Diakhir akan didapatkan biaya total yang muncul selama simulasi berlangsung. Semakin minimum biaya total yang muncul maka akan semakin baik.

Verifikasi Simulasi *Decentralized*

Pada verifikasi *decentralized*, simulasi dijalakan selama tiga puluh hari. Telah ditentukan biaya *inventory* sebesar 0.5, biaya *backorder* sebesar 1, jumlah *inventory* awal *retailer* sebesar 100, *inventory* maksimum *retailer* sebesar 100, dan *inventory* minimum *retailer* sebesar 70, jumlah *inventory* awal *distributor* sebesar 200, *inventory* maksimum *retailer* sebesar 200, dan *inventory* minimum *retailer* sebesar 70, jumlah *inventory* awal *factory* sebesar 600, *inventory* maksimum *factory* sebesar 600, *inventory* minimum *factory* sebesar 420, kapasitas produksi harian sebesar 100, dan *demand* harian sebesar 10. Parameter-parameter tersebut berlaku berdasarkan *role* dalam simulasi. Hasil yang didapat menunjukkan total biaya sebesar 16125 dan melalui penghitungan manual menghasilkan hasil yang sama. Oleh karena itu, simulasi dapat dinyatakan terverifikasi.



Gambar 11. Hasil simulasi *decentralized*

Tabel 1. Hasil penghitungan manual simulasi *decentralized*

Biaya Total Tiap *Role*	Jumlah
Biaya Total "*RETAILER A*"	495
Biaya Total "*RETAILER B*"	495
Biaya Total "*RETAILER C*"	495
Biaya Total "*DISTRIBUTOR A*"	2090
Biaya Total "*DISTRIBUTOR B*"	2090
Biaya Total "*DISTRIBUTOR C*"	2090
Biaya Total "*FACTORY*"	8370
Total Biaya	16125

Verifikasi Simulasi *Centralized*

Pada verifikasi *centralized*, simulasi dijalakan selama tiga puluh hari. telah biaya *inventory* sebesar 0.5, biaya *backorder* sebesar 1, jumlah *inventory* awal *retailer* sebesar 100, *inventory* maksimum *retailer* sebesar 100, dan *inventory* minimum *retailer* sebesar 70, jumlah *inventory* awal *distributor* sebesar 300, *inventory* maksimum *distributor* sebesar 500, dan *inventory* minimum *distributor* sebesar 210, jumlah *inventory* awal *factory* sebesar 300, *inventory* maksimum *factory* sebesar 500, *inventory* minimum *factory* sebesar 210, kapasitas produksi harian sebesar 100, dan *demand* harian sebesar 10. Parameter-parameter tersebut berlaku berdasarkan *role* dalam simulasi.

Gambar 12. Hasil simulasi *centralized*

Tabel 2. Hasil penghitungan manual simulasi *centralized*

Biaya Total Tiap *Role*	Jumlah
Biaya Total "*RETAILER A*"	495
Biaya Total "*RETAILER B*"	495
Biaya Total "*RETAILER C*"	495
Biaya Total "*DISTRIBUTOR*"	3350
Biaya Total "*FACTORY*"	5825
Total Biaya	10660

Setelah dilakukan penghitungan, ditemukan hasil yang sama antara hasil simulasi dengan hasil penghitungan manual. Hasil yang didapat menunjukkan total biaya sebesar 10660. Oleh karena itu, simulasi dapat dinyatakan terverifikasi. Jika terverikasi maka dapat dikatakan bahwa simulasi berjalan dengan benar. Contoh perhitungan manual terdapat di Lampiran 1.

Simpulan

Hasil dari penelitian ini adalah *game* simulasi yang menggambarkan sistem rantai pasok *decentralized* dan *centralized*. Untuk *game* simulasi *decentralized* terdapat tiga *retailer,* tiga *distributor* dan satu *factory* dengan *lead time order* selama tujuh hari atau satu minggu. Untuk *game* simulasi *decentralized* terdapat tiga *retailer* dan satu *factory* dengan *lead time order* selama tujuh hari atau satu minggu. Kedua *game* simulasi tersebut dapat dijadikan media pembelajaran dalam pengajaran materi *risk pooling*.

Dengan adanya pemahaman dari peserta didik dengan dibandingkannya kedua simulasi tersebut maka diharapkan materi *risk pooling* berhasil diajarkan. Hal ini dapat ditinjau dari teori *risk pooling* sendiri yang menyatakan bahwa rantai pasok *centralized* lebih baik dibanding rantai pasok *decentralized*. Dari perbandingan keduanya, *game* simulasi *centralized* menghasilkan biaya yang lebih murah dibandingkan *game* simulasi *decentralized*.

Dengan adanya penelitian ini diharapkan akan ada *game-game* simulasi dengan menggambarkan metode-metode yang lain sehingga proses pembelajaran akan lebih menarik dan mudah dipahami oleh mahasiswa/i. Metode-metode lain yang ingin disimulasikan alangkah lebih baiknya jika berasal dari mata kuliah diluar rantai pasok. Hal ini tentu membantu terciptanya kegiatan pembelajaran yang lebih menarik dan interaktif.

Pembuatan *game* simulasi juga dapat dibuat dengan menggunakan aplikasi-aplikasi yang lain. Disini peneliti menggunakan aplikasi *AnyLogic* dikarenakan aplikasi ini cukup memudahkan pembuatan simulasi. Meskipun dibutuhkan adaptasi dan pengenalan lebih mendalam mengenai Bahasa pemrogaman *java* yang sangat berperan penting dalam pembuatan simulasi.

Daftar Pustaka

1. Martono, R. V., *Dasar-Dasar Manajemen Rantai Pasok,* Bumi Aksara, Jakarta, 2019.
2. Nadeem, S. P., Risk Pooling, A Technique to Manage Risk in Supply Chain Management, *Logistics and Supply Chain Management Publication*, 2016, pp. 86-93.
3. Padilla, J. J., Using Simulation Games for Teaching and Learning Discrete-Event Simulation, *Proceedings of the 2016 Winter Simulation Conference*, 2016, pp. 3375-3384.

Lampiran

Lampiran 1. Perhitungan manual simulasi *centralized*

DAY	RETAILER A							
	DEMAND	INVENTORY	BACKORDER	EXPECTED	RECEIVED	INVENTORY COST	BACKORDER COST	TOTAL COST
0	10	0	0	0	0	0	0	0
1	10	-10	0	110	0	-5	0	-5
2	10	-20	0	10	0	-10	0	-10
3	10	-30	0	120	0	-15	0	-15
4	10	-40	0	30	0	-20	0	-20
5	10	-50	0	30	0	-25	0	-25
6	10	-60	0	30	0	-30	0	-30
7	10	-70	0	30	0	-35	0	-35
8	10	-80	0	30	0	-40	0	-40
9	10	-90	0	30	0	-45	0	-45
10	10	-70	0	70	30	-35	0	-35
11	10	-80	0	70	0	-40	0	-40
12	10	-90	0	70	0	-45	0	-45
13	10	-100	100	70	0	-50	100	50
14	10	-110	110	70	0	0	110	110
15	10	-120	120	70	0	0	120	120
16	10	-130	130	70	0	0	130	130
17	10	-70	0	70	70	-35	0	-35
18	10	-80	0	70	0	-40	0	-40
19	10	-90	0	70	0	-45	0	-45
20	10	-100	0	70	0	-50	0	-50
21	10	-110	10	70	0	0	10	10
22	10	-120	20	70	0	0	20	20
23	10	-130	30	70	0	0	30	30
24	10	-70	0	70	70	-35	0	-35
25	10	-80	0	70	0	-40	0	-40
26	10	-90	0	70	0	-45	0	-45
27	10	-100	0	70	0	-50	0	-50
28	10	-110	10	70	0	0	10	10
29	10	-120	20	70	0	0	20	20
30	10	-130	30	70	0	0	30	30
					TOTAL	-735	580	-155

FACTORY						
INVENTORY	BACKORDER	CREATING	CREATED	INVENTORY COST	BACKORDER COST	TOTAL COST
300	0	0	0	0	0	0
300	0	0	0	150	0	150
300	0	0	0	150	0	150
300	0	0	0	150	0	150
300	0	0	0	150	0	150
10	0	490	0	5	0	5
110	0	0	100	55	0	55
210	0	0	100	105	0	105
310	0	0	100	155	0	155
410	0	0	100	205	0	205
500	0	0	90	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
80	0	420	0	40	0	40
180	0	0	100	90	0	90
280	0	0	100	140	0	140
380	0	0	100	190	0	190
480	0	0	100	240	0	240
500	0	0	20	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
500	0	0	0	250	0	250
			TOTAL	5825	0	5825