PERANCANGAN WEB SIMULASI SISTEM DINAMIS MENGGUNAKAN ASP DAN POWERSIM SDK

SKRIPSI

ROTUA JUNITA V. MANULLANG
04 04 077101

UNIVERSITAS INDONESIA
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INDUSTRI
DEPOK
JULI 2008

PERANCANGAN WEB SIMULASI SISTEM DINAMIS MENGGUNAKAN ASP DAN POWERSIM SDK

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

ROTUA JUNITA V. MANULLANG
04 04 077101

DEPARTEMEN TEKNIK INDUSTRI
FAKULTAS TEKNIK
UNIVERSITAS INDONESIA
DEPOK
JULI 2008

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	**: Rotua Junita V. Manullang**
NPM	**: 04 04 07701**
Tanda Tangan	**:**
Tanggal	**: 10 Juli 2008**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Rotua Junita V. Manullang
NPM	: 0404077101
Program Studi	: Teknik Industri
Judul Skripsi	: Perancangan Web Simulasi Sistem Dinamis Menggunakan ASP dan Powersim SDK

Telah berhasil dipertahankan di hadapan Dewan Penguji dan di terima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana pada Program Teknik Industri Fakultas Teknik Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: Armand Omar Moeis, S.T., M.Sc	(.....)
Penguji	: Ir. M. Dachyar, M.Sc.	(.....)
Penguji	: Ir. Yadrifil, M.Sc	(.....)

Ditetapkan di : Depok
Tanggal : 10 Juli 2008

UCAPAN TERIMA KASIH

Puji syukur penulis kembalikan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmatNya, penulis dapat menyelesaikan skripsi yang berjudul "**Perancangan Web Simulasi Sistem Dinamis Menggunakan Powersim SDK dan ASP**" ini. Penyusunan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Industri pada Fakultas Teknik Universitas Indonesia.

Penulis mengucapkan terima kasih kepada:

1. Mama dan Papa terkasih, Denny, Sopar, dan segenap keluarga besar atas setiap doa, kasih sayang, dan dukungan yang tiada habisnya kepada penulis.
2. Bapak Armand Omar Moeis, S.T, MSc., selaku dosen pembimbing skripsi untuk segala bantuan, dukungan, perhatian, dan pengarahan kepada penulis.
3. Bapak Ir. Akhmad Hidayatno, MBT selaku dosen pembimbing akademis atas segala bimbingan selama penulis menyelesaikan studi di Teknik Industri UI.
4. Bapak Ir. M. Dachyar, M.Sc dan Bapak Ir. Yadrifil, M.Sc atas kritik dan masukan yang membangun dalam seminar dan sidang skripsi.
5. Kak Laura, Mas Riyantoko, Kak Komar dan Gilang yang telah memberi banyak bimbingan dan masukan selama mengerjakan skripsi.
6. Teman-teman TIUI 2004 dan teman-teman skripsi 2008, Randy, Ade, Hendry, Hery, Asep, dan Eko yang memberikan kebersamaan sampai saat ini.
7. Ria, Frisca, Olan, Benny, Tirza, Loly, Nia, Tita, Togi, Budi, Satria, Norman, Ijul dan yang tidak bisa disebut satu persatu, atas setiap dukungannya.

Penulis menyadari bahwa skripsi ini jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan. Semoga skripsi ini dapat memberikan manfaat bagi setiap orang yang membacanya dan bagi pengembangan ilmu pengetahuan di masa yang akan datang.

Depok, 10 Juli 2008

Penulis

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI

KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Rotua J. V. Manullang
NPM/NIP : 0404077101
Program Studi : Teknik Industri
Departemen : Teknik Industri
Fakultas : Teknik
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Non-Eksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul:

"P**erancangan Web Simulasi Sistem Dinamis Menggunakan ASP dan Powersim SDK**"

beserta perangkat yang ada (bila diperlukan). Dengan Hak Bebas Royalti Non-Ekslusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelolanya dalam bentuk pangkalan data (*database*), mendistribusikannya, dan menampilkan/mempublikasikannya di Internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok

Pada tanggal: 10 Juli 2008

Yang menyatakan,

(Rotua J. V. Manullang)

ABSTRAK

Nama : Rotua Junita V. Manullang
Program Studi : Teknik Industri

Judul :Perancangan Web Simulasi Sistem Dinamis Menggunakan ASP dan Powersim SDK

Simulasi merupakan cara yang efektif untuk memodelkan sistem nyata. Pemodelan dilakukan dengan memberikan batasan-batasan sesuai dengan objektif yang ingin dicapai dari model. Simulasi dibedakan menjadi dua yaitu simulasi tradisional dan simulasi komputer. Simulasi tradisional melibatkan beberapa pemain yang berinteraksi langsung untuk memetakan sistem nyata melalui peran masing-masing. Seiring perkembangan teknologi, maka simulasi tradisional berkembang menjadi simulasi komputer yang memanfaatkan program simulasi untuk memodelkan sistem melalui model yang dibuat. Jaringan internet yang juga berkembang memungkinkan integrasi antara simulasi komputer dengan internet sehingga menghasilkan simulasi berbasis aplikasi web. Tujuan yang akan dicapai adalah simulasi dapat dijalankan oleh beberapa peserta yang berinteraksi melalui internet.

Penelitian ini merupakan penyempurnaan perancangan simulasi berbasis aplikasi web yang dilakukan dalam 4 tahap yaitu penjelasan kembali pembuatan model sistem dinamis, penyempurnaan *web interface* model, *hosting* web ke penyedia layanan internet, lalu verifikasi web. Model yang digunakan adalah *Beer Distribution Game*. Penyempurnaan *interface* dilakukan pada sisi administrator yaitu penyempurnaan grafik, sedangkan pada sisi pemain dilakukan penambahan fitur web. Konsep perancangan dasar dari web menggunakan konsep telah dibuat pada penelitian sebelumnya termasuk *database*, penggunaan *script* ASP dan Powersim SDK.

Hasil penelitian ini adalah *Beer Distribution Game Online* yang diakses dari internet dengan penyempurnaan *web interface* sebagai nilai tambah.

Kata kunci: Sistem Dinamis, Simulasi Berbasis Aplikasi Web, ASP, SDK.

ABSTRACT

Name : Rotua Junita V. Manullang
Study Program : Industrial Engineering
Judul : Designing Web Simulation Of System Dynamic Using ASP and Powersim SDK

Simulation represents an effective methodology to model a real system. Modeling is conducted with boundaries based on the desired objectives. Simulation can be conducted in two approaches; traditional simulation and computer simulation. Traditional simulation entangles some players directly interacts each other through each role. Along technological growth, hence traditional simulation develop into computer simulation that exploits simulation program to model a system. Internet network wich also expand, enable the integration between computer simulation and internet so that yield the web-based simulation. This research's target is simulation can be run by players which have interaction only by using internet.

This research represent the completion of web-based simulation design that is performed in four phases; re-explain the making of system dynamic model, development of web interface model, hosting web to the ISP so the simulation can be played online, then result's verification. The developed model is Beer Distribution Game. Interface development at the administrator's side that is graph completion while at the player's side that is additional web feature. Base conception of the web is developed from the previous research's concept included database, use of ASP script and Powersim SDK.

The result of this research is Beer Distribution Game Online with completion of web interface as an added value.

Keywords: System Dynamic, Web-based Simulation, ASP, SDK

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

1. PENDAHULUAN

1.1. Latar Belakang

Sistem jaringan internet yang terus mengalami peningkatan pesat dari sudut kemudahan akses informasi maupun sebagai alat komunikasi menjadi sangat bermanfaat bagi manusia di seluruh belahan dunia. Kehidupan manusia menjadi lebih terkomputerisasi dengan lebih baik sehingga membantu dalam penyelesaian pekerjaan dengan lebih cepat. Salah satu pemanfaatan jaringan internet adalah sebagai sumber jaringan pertukaran informasi dari seluruh dunia dalam berbagai bidang kehidupan. Jaringan internet juga saat ini bisa diakses dengan sangat mudah, tidak hanya menggunakan komputer namun juga alat yang lain seperti telepon seluler.

Perkembangan internet dari tahun ke tahun bermula dari keikutsertaan jutaan orang dari berbagai belahan dunia dalam pertukaran informasi di dalam 'dunia maya' tersebut. Metode yang digunakan dalam pengembangan internet ini adalah melalui web yang mulai diperkenalkan sekitar tahun 1994. Perkembangan yang makin meluas juga dipengaruhi oleh penggunaannya dalam transaksi berbagai macam bisnis di dunia. Selain itu, jaringan internet juga sangat bermanfaat bagi lingkungan pendidikan yaitu sarana ilmu pengetahuan dengan cakupan sangat luas, termasuk di dalamnya adalah dalam bentuk simulasi.

Definisi simulasi berdasarkan kamus Oxford America adalah metode untuk mereplikasi kondisi suatu situasi dengan menggunakan model untuk mempelajari, mengevaluasi atau melatih perilaku sistem. Menurut Schriber (1987), simulasi adalah pemodelan dari suatu proses atau sistem dimana model tersebut memberikan respon seperti yang terjadi pada sistem nyata ketika berada dalam situasi yang sama.[1]

Simulasi yang baik memiliki objektif yang jelas dan kuantitatif dan secara umum dapat diklasifikasikan menjadi beberapa tujuan antara lain untuk

[1] Schriber 1987, dikutip dari Charles Harrel, Biman K. Ghosh, dan Royce Bowden, *Simulation Using ProModel*, Mc Graw Hill, 2000, hal. 5.

menganalisa sistem, edukasi dan pelatihan, akuisisi dan penerimaan sistem, riset, dan hiburan.[2] Sistem dalam dunia nyata bersifat dinamis, kompleks dan mengalami perubahan secara kontinu berdasarkan perubahan waktu, dengan demikian, mensimulasikan sistem riil memerlukan penyederhanaan. Penyederhanaan sistem untuk disimulasikan sehingga tetap menghasilkan sistem yang dinamis dan memenuhi objektif memerlukan batasan-batasan yang harus didefinisikan dengan jelas sebelumnya.

Manfaat simulasi sudah dirasakan baik dalam dunia pendidikan maupun dalam dunia bisnis. Dalam dunia pendidikan, simulasi digunakan sebagai pembelajaran yang aplikatif karena dengan simulasi, siswa terlibat aktif dalam kondisi buatan untuk mengaplikasikan berbagai teori yang berkaitan dengan kondisi tersebut. Simulasi tersebut pada akhirnya akan sangat membantu dalam memberikan pemahaman yang lebih mudah dan menarik. Proses pembelajaran yang diperoleh dari simulasi dapat memperdalam pemahaman siswa tentang sistem dinamis dan berbagai teori mengenai manajemen. Perkembangan pemodelan dan simulasi juga diikuti dengan perkembangan perangkat lunak mengenai hal tersebut. Berbagai perusahaan mengembangkan perangkat lunak untuk menjalankan simulasi dengan menawarkan berbagai fitur yang memiliki keunggulan dan kelemahan masing-masing, beberapa diantaranya adalah Powersim, Promodel, Vensim, dan lain-lain. Namun, untuk menggunakannya untuk keperluan pribadi, seseorang memerlukan lisensi yang mahal sehingga biasanya dimiliki oleh institusi tertentu saja.

Mahalnya lisensi untuk perangkat lunak tersebut akhirnya menjadi pertimbangan untuk mengintegrasikan simulasi dengan jaringan internet. Integrasi tersebut berawal pada tahun 1995 sehingga memungkinkan terjadinya pelaksanaan model yang paralel dan terdistribusi serta penyimpanan data model yang terdistribusi (Fishwick, 1996).[3] Simulasi berbasis aplikasi web memberikan berbagai nilai tambah dan manfaat bagi riset dan pendidikan yaitu kepraktisan dan

[2] Richard E. Nance dan Robert G. Sargent, *Perspective on the Evolution of Simulation*, *Operation Research*, Vol. 50, No. 1, 50th *Anniversary Issue*, Januari-Februari, 2002, hal. 161.
[3] Paul A. Fishwick, *Web Based Simulation*, *Proceeding of the 1997 Winter Simulation Conference*, University of Florida, 1997, hal 100.

efisiensi baik dari segi waktu maupun biaya serta dapat diakses oleh siapa saja, kapan saja dan dimana saja. Kemudahan mengakses simulasi berbasis aplikasi web dikarenakan pengguna tidak harus membeli perangkat lunak dengan harga yang mahal namun bisa mengakses dengan menggunakan jaringan internet. Data dan perintah simulasi sudah disimpan di komputer *server* dengan menggunakan program simulator dan *script* pemograman tertentu. Pengguna hanya menggunakan komputer pribadi dan modem atau jaringan internet untuk melakukan simulasi yang ditawarkan. Dengan memanfaatkan kemudahan internet tersebut maka banyak pihak yang mengadakan kompetisi dalam bentuk permainan simulasi dengan aplikasi web, sebagai contoh adalah permainan simulasi L'0real E-Strat Challenge. Simulasi *online* ini juga dimanfaatkan oleh perusahaan-perusahaan untuk melakukan pelatihan manajerial sehingga para karyawannya dapat berlatih untuk mengambil keputusan dengan kondisi yang sudah diatur dalam simulasi tersebut sehingga terlihat akibat dan konsekuensi dari keputusan itu jika nantinya diterapkan dalam dunia nyata di kemudian hari. Dengan menggunakan simulasi terlebih dahulu, perusahaan tidak akan khawatir mendapatkan kerugian atau resiko yang berbahaya ketika mengambil keputusan karena telah diuji coba dalam simulasi.

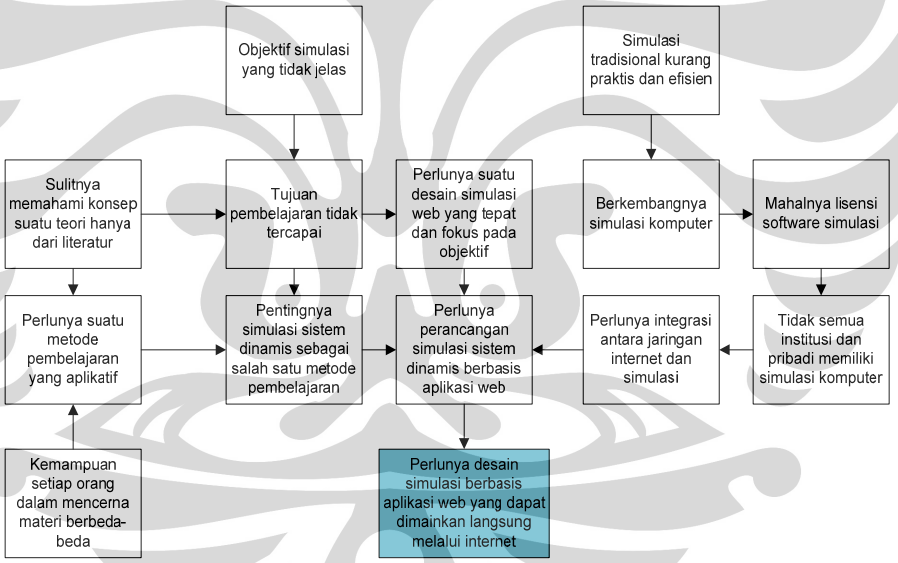
Pembuatan simulasi *online* membutuhkan beberapa komponen yaitu: program simulasi sebagai simulator, jaringan internet, *script* pemrograman untuk membuat halaman situs yang dinamis, interaktif dan atraktif, *database engine* sebagai bank data, dan *account administrator* untuk meng-*upload* simulasi ke web.

Desain simulasi web haruslah tepat sasaran, tidak berlebihan namun dapat dipelajari dengan mudah. Desain yang tepat dan memperhatikan ergonomi akan mengarahkan pengguna (*user*) untuk memperoleh objektif yang ingin disampaikan oleh simulasi tersebut dan juga tidak mudah merasa jenuh karena permainan simulasi yang cenderung kompleks dan tidak sederhana.

Dalam lingkungan Teknik Industri Universitas Indonesia, simulasi sistem dinamis dalam bentuk web telah dimulai pada tahun 2006 oleh saudara Nur Fathony, ST dalam penelitian tugas akhirnya yang berjudul *Perancangan Simulasi Berbasis Aplikasi Web Dengan Menggunakan Bahasa Pemrograman ASP*. Topik

ini juga disempurnakan lagi dengan menampilkan grafik-grafik yang menunjukkan perilaku variabel yang terdapat dalam permainan simulasi tersebut serta pembuatan modul pemrogramannya sebagai dokumentasi untuk alat pembelajaran. Bagian ini diteliti oleh saudari Laura Olivia Ramadhona dalam tugas akhirnya yang berjudul *Perancangan Simulasi Sistem Dinamis Berbasis Aplikasi Web Menggunakan Powersim SDK dan ASP* pada tahun 2007. Mengacu pada kedua penelitian tersebut maka perlu adanya pengembangan lebih lanjut untuk simulasi berbasis aplikasi web yang dapat dimainkan oleh beberapa pemain dan dapat diakses secara langsung melalui internet dengan *website* khusus ditambah dengan penyempurnaan fitur *web interface* pada sisi pemain.

1.2. Diagram Keterkaitan



Gambar 1.1 Diagram keterkaitan masalah

1.3. Pokok Permasalahan

Pokok permasalahan yang akan diselesaikan dalam penelitian ini adalah perlunya dibuat sebuah program simulasi berbasis aplikasi web yang dapat dijalankan oleh *multi-user* secara langsung melalui koneksi internet. Pada

akhirnya pengguna dapat mempelajari hubungan antar variabel sistem melalui variabel perilaku dari sistem yang dimodelkan tersebut dalam lingkup global.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah menghasilkan sebuah program simulasi berbasis aplikasi web yang dapat diakses oleh *multiplayer* secara langsung dari internet serta panduan permainan dan modul pembelajarannya. Jadi, hasil akhir yang dapat diakses langsung adalah berupa *web site* permainan.

1.5. Pokok Permasalahan

Pembuatan model simulasi permainan yang berbasis aplikasi web dan juga desain web akan meliputi perangkat lunak simulasi dan sistem jaringan internet yang menghubungkan komputer dengan server sehingga akan terjadi transfer data yang luas, dengan demikian diperlukan batasan-batasan yang jelas dari penelitian ini sehingga tetap terfokus pada tujuan yang telah ditetapkan. Batasan-batasan tersebut adalah:

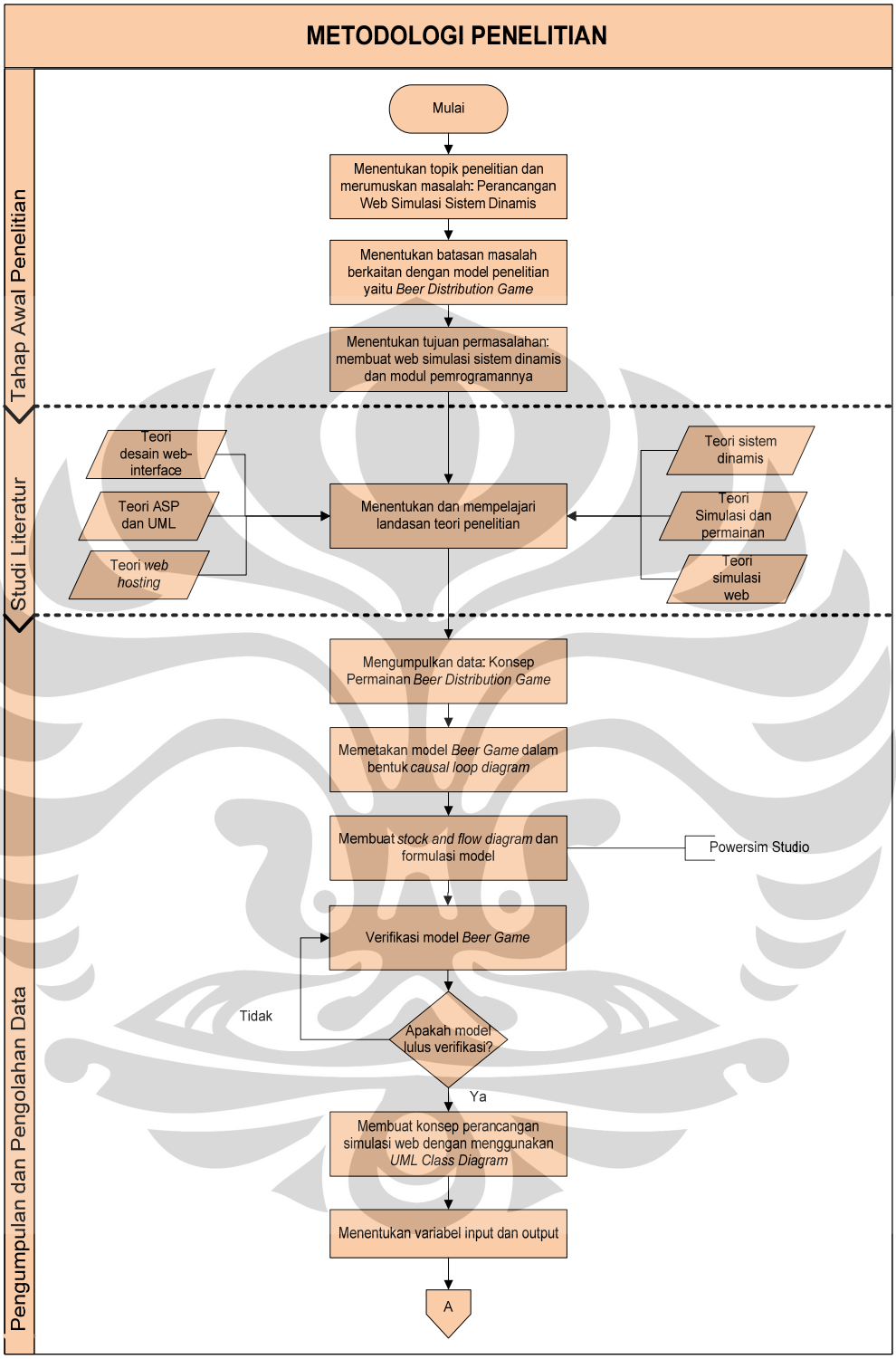
1. Model yang digunakan adalah model *Beer Distribution Game* yang diciptakan oleh MIT Sloan School of Management. Model ini telah digunakan pada penelitian selanjutnya dan tetap dipakai sebagai model simulasi penelitian ini dengan alasan yang sama yaitu model ini memiliki kompleksitas detil simulasi yang cukup tinggi namun kompleksitas *interface* yang rendah, sehingga kondisi ini membantu penulis dalam proses pembelajaran permainan simulasi produksi.
2. Konsep permainan mengacu kepada konsep permainan manual atau *board game*.
3. Perancangan *web interface* dan aplikasi pada jaringan internet menggunakan perangkat lunak Powersim SDK 2005, Ms. Access, dan ASP.
4. Pembuatan website simulasi dengan implementasi *web design* dan *web hosting*.
5. Pengguna aplikasi ini merupakan peserta permainan dan administrator.

1.6. Metodologi Penelitian

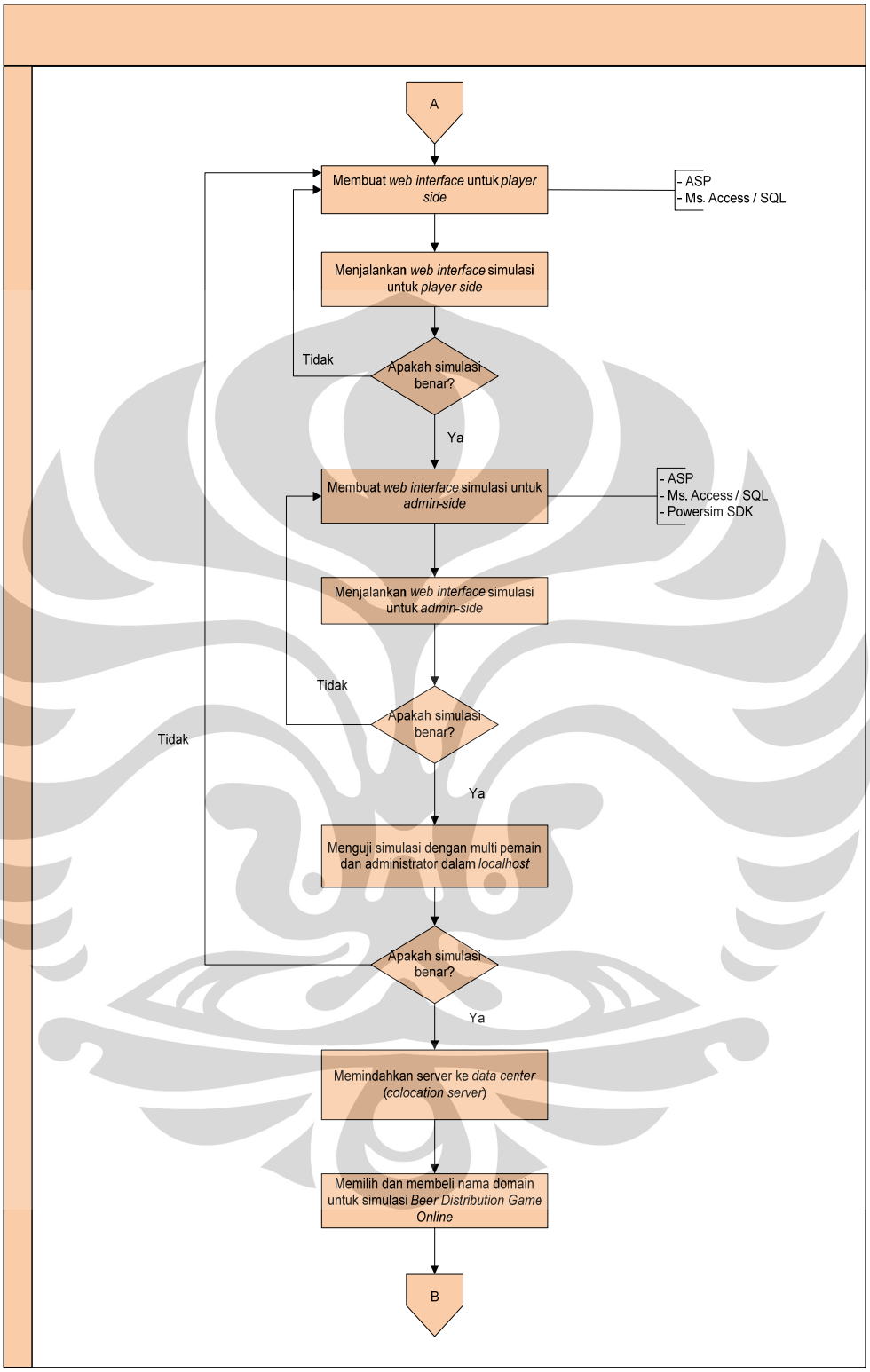
Metodologi yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Menentukan topik dan perumusan masalah yang akan diteliti.
2. Menentukan batasan masalah berkaitan dengan model simulasi permainan.
3. Menentukan tujuan yang ingin dicapai melalui penelitian ini yaitu menghasilkan *web site* dari program simulasi sistem dinamis berbasis aplikasi web berserta panduan permainan untuk pengguna.
4. Melakukan studi literatur untuk menentukan dasar teori yang akan digunakan dalam penelitian. Teori yang digunakan dalam penelitian ini adalah teori sistem dinamis, simulasi dan permainan, simulasi berbasis aplikasi web, desain *web interface*, ASP, UML dan tahapan *web hosting*.
5. Mengumpulkan data yang akan digunakan dalam penelitian melalui jurnal dan internet. Data yang dibutuhkan adalah konsep permainan *Beer Distribution Game* yang telah diimplementasikan pada penelitian sebelumnya. Pada penelitian ini, pengolahan data yang dilakukan merupakan tahap pembelajaran dan pengembangan konsep. Adapun langkah pengolahan data permainan yang adalah sebagai berikut:
   a. Membuat *causal loop diagram* untuk memetakan konsep permainan.
   b. Membuat *stock and flow diagram* dari model serta formulasi masing-masing variabelnya dengan menggunakan Powersim Studio 2005.
   c. Melakukan verfikasi terhadap model yang telah dibuat.
6. Membuat konsep perancangan simulasi web dengan beberapa pengembangan menggunakan UML Class Diagram.
7. Menentukan variabel input dan output yang akan ditampilkan pada *web interface*.
8. Memperbaiki *web interface* untuk sisi pemain (*player-side*) menggunakan ASP dan Ms Access/ SQL sebagai bank data.

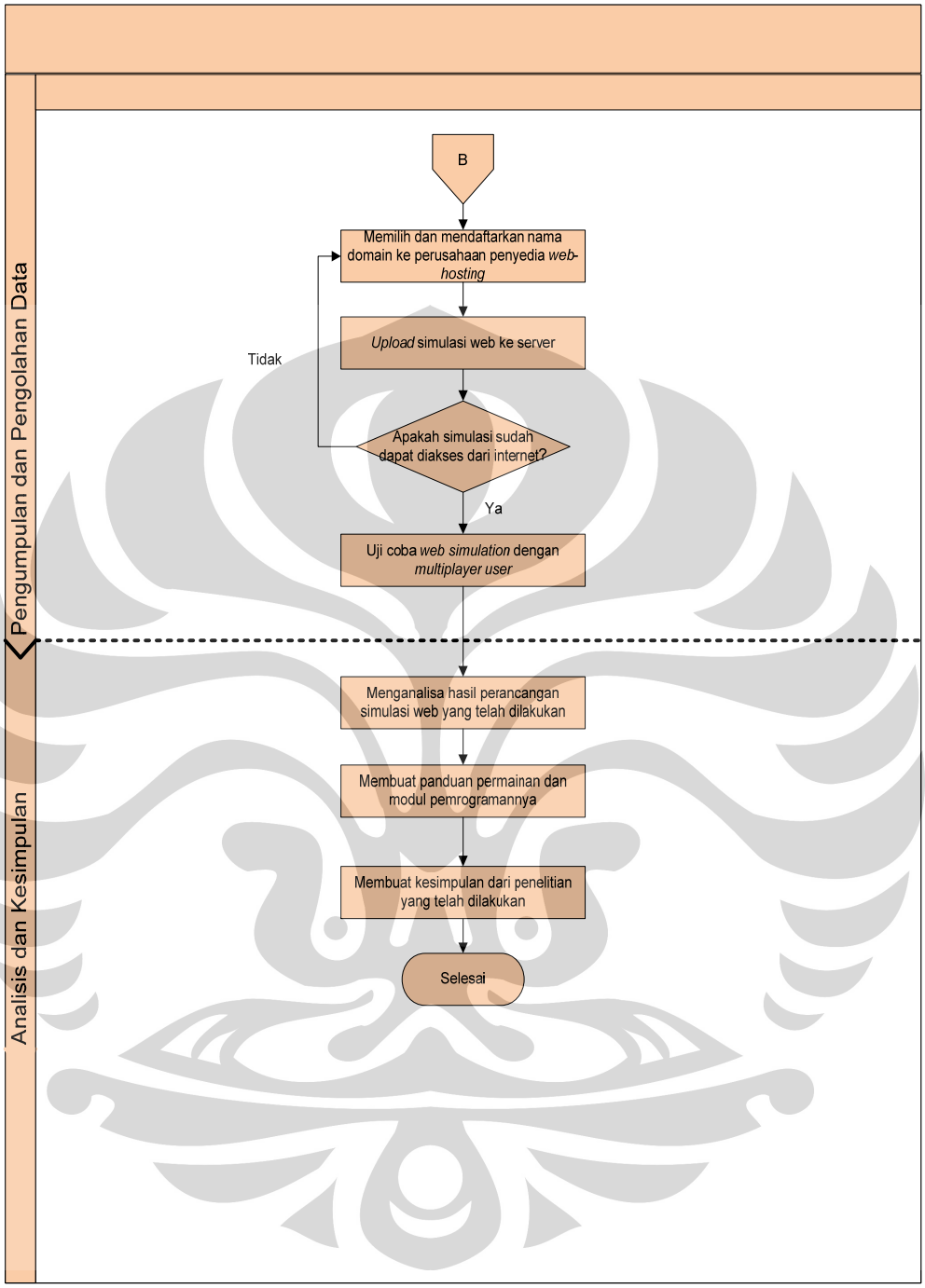
9. Melakukan pengujian terhadap *web interface* yang telah dibuat apakah dapat mengakses data simulasi dengan benar.
10. Memperbaiki *web interface* untuk sisi administrator permainan (*admin-side*) dengan menggunakan Powersim SDK, ASP dan Ms Access/SQL.
11. Melakukan verfikasi *web interface* yang telah dibuat apakah sudah mengakses data dan menjalankan simulasi dengan benar.
12. Melakukan verifikasi dari uji coba simulasi permainan dengan melibatkan dua sisi *interface* yang telah dibuat yaitu dari sisi pemain dan administrator sampai memperoleh simulasi yang benar.
13. Melakukan pemindahan server ke *data center* (proses *colocation server*).
14. Membuat nama domain untuk simulasi permainan dan melakukan pembelian domain dan memperoleh DNS sehingga simulasi dapat diakses langsung dari internet.
15. Mendaftarkan nama domain ke perusahaan *web hosting* sehingga diperoleh *account administrator* untuk meng-*upload* aplikasi ke internet.
16. Meng-*upload* simulasi yang telah dibuat ke server.
17. Mengakses permainan via internet untuk mengetahui apakah simulasi *online* sudah terdaftar dan dapat dimainkan.
18. Melakukan uji coba simulasi web dengan *multiplayer* untuk mengetahui apakah simulasi dapat dimainkan seperti ketika belum di-*publish* ke *website*.
19. Menganalisa hasil perancangan simulasi web yang telah dilakukan secara keseluruhan.
20. Membuat panduan permainan dan modul pemrograman simulasi web yang telah dibuat.
21. Membuat kesimpulan penelitian dari hasil analisis.



Gambar 1.2 Diagram alir metodologi penelitian



Gambar 1.2 Diagram alir metodologi penelitian (lanjutan)



Gambar 1.2 Diagram alir metodologi penelitian **(lanjutan)**

1.7. Sistematika Penulisan

Penelitian ini menggunakan sistematika penulisan sesuai dengan petunjuk penulisan skripsi yang telah ditetapkan. Penulisan skripsi untuk penelitian ini terdiri dari empat bab yang akan menuntun pada penjelasan yang berkelanjutan sampai pada kesimpulan penelitian ini.

Sebagai langkah awal, bagian pendahuluan memiliki arti yang penting dalam membangun konsep dan tujuan yang jelas ketika mengerjakan penelitian ini. Pendahuluan ditulis dalam bentuk penjelasan latar belakang penelitian, tujuan yang akan dicapai dari penelitian ini, batasan masalah, metodologi penelitian dan sistematika penulisan penelitian ini. Dalam bagian ini, pembaca akan dibukakan konsep awal mengenai penelitian ini.

Penelitian yang dilakukan membutuhkan dasar teori dalam mengerjakan proses atau tahapan-tahapan berkaitan dengan penelitian yang akan dilakukan. Bab landasan teori akan membantu penulis dalam melandasi penelitian untuk mengambil keputusan yang tepat mengenai masalah yang akan diteliti dan membantu dalam memberikan alternatif metode dan alat bantu yang akan digunakan. Penelitian ini menggunakan dasar teori mengenai sistem dinamis, simulasi dan permainan, teori *web based simulation*, desain *web interface*, *script* pemrograman ASP (*Active Server Pages)*, teori UML (*Unified Modelling Language*), dan tahapan *web-hosting*.

Bab selanjutnya merupakan bab yang akan membahas pembuatan model dan penyempurnaan fitur web. Bagian ini akan menjelaskan konsep model dan pembuatan model hingga pada proses verifikasi yang dilakukan dengan menggunakan perangkat lunak Powersim Studio dan Microsoft Excel. Hasil dari pengolahan yang telah dilakukan akan menjadi data input pada proses pembuatan *interface* dan pada tahap pengembangan *web interface* program simulasi sehingga menghasilkan *web interface* yang lebih baik. Pembuatan program ini menggunakan perangkat lunak Powersim SDK dan, *script* ASP, dan Microsoft Access. Pada bagian pengembangan *web interface* model, penulis akan menjelaskan tahap pengembangan *web interface* dari simulasi web baik dari sisi pemain maupun dari sisi administrator serta melakukan analisa terhadap hasil pengujian dari program simulasi web yang telah dibuat. Pengujian ini dilakukan

untuk memastikan program simulasi berjalan dalam batasan *localhost*. Setelah program berhasil dimainkan, selanjutnya akan dijelaskan juga bagaimana membuat web untuk simulasi ini sehingga dapat diakses langsung dari internet (*web-hosting*) serta uji coba yang akan dilakukan langsung oleh *user*.

Pada bab verifikasi web dan analisa, penulis akan menjelaskan tahap verifikasi *web-based simulation* yang sudah dapat diakses lewat internet. Bagian ini dilakukan dengan tujuan untuk mengetahui apakah program simulasi sudah berjalan sesuai dengan konsep awal model.

Hasil penelitian yang telah dilakukan kemudian dirangkum dalam kesimpulan dan juga saran yang dapat digunakan sebagai pengembangan pada penelitian selanjutnya.

2. DASAR TEORI

2.1. Sistem Dinamis

Kita hidup dalam lingkungan yang tersusun dengan kompleks, sistem buatan manusia dimana kita akan bergantung pada keamanan, kenyamanan dan penghidupan. Secara rutin, kita akan bergantung pada transportasi, pemeliharaaan kesehatan, sistem produksi dan distribusi yang memberikan kebutuhan barang dan jasa, serta kita juga akan sangat memperhatikan kualitas setiap barang dan jasa yang akan kita peroleh baik itu kenyamanan, efektivitas waktu, biaya dan pelayanan yang akan diberikan oleh sistem-sistem ini. Sistem-sistem ini saling berkaitan satu dengan yang lain karena di dalam sistem itu sendiri, juga tersusun atas komponen maupun subsistem yang saling berintegrasi dan bekerja sama dalam mencapai tujuannya (Blanchard 1991). Seperti kondisi yang digambarkan sebelumnya, sistem tersebut adalah sistem yang dinamis dan terus berubah menurut waktu dan dipengaruhi oleh faktor internal maupun eksternal. Sistem tersebut dapat dipelajari melalui pendekatan sistem dinamis untuk mengetahui perubahan perilaku dalam sistem baik itu oleh adanya hubungan sebab akibat karena *feedback* yang terjadi atau hubungan perilaku yang lain.

Pendekatan sistem dinamis pertama kali dikembangkan oleh Profesor Jay W. Forrester di MIT pada awal tahun 1960-an. Sejak saat itu, ia mulai menerapkan apa yang ia pelajari mengenai sistem selama ia bekerja sehingga menghasilkan permasalahan yang kompleks yaitu sistem dinamis.

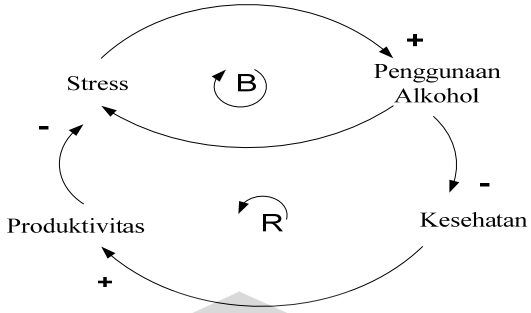
Menurut Vanderminden, sistem dinamis (SD) memiliki dua fungsi yaitu sebagai metodologi dan sebagai bidang studi. Sebagai metodologi, SD merupakan metode yang digunakan untuk memodelkan sturktur proses tersebut melalui penelitian terhadap aliran, akumulasi dan interaksi sumber daya berdasarkan waktu dalam *feedback loop* yang saling berhubungan secara dinamis. Jikalau dipandang sebagai bidang studi, SD merupakan ilmu pengetahuan yang mempelajari siklus, pertumbuhan dan kesinambungan perubahan dinamis dari

sistem swa-kendali.[4] Serupa halnya dengan *system thinking*, sistem dinamis juga membentuk diagram kausal yang menggambarkan hubungan sebab akibat antar variabel sistem yang memiliki *feedback* dan *delay*. *System thinking* merupakan cara pandang dan cara pikir kita dalam menghadapi suatu permasalahan secara global dan membuat gambaran sistem yang menunjukkan interaksi atau hubungan-hubungan yang terjadi di dalamnya. Pada saat kita dihadapkan dengan suatu masalah, kita dapat mengambil keputusan berdasarkan pertimbangan-pertimbangan yang menyebabkan masalah itu terjadi dan hubungan di antaranya. Dengan demikian, hal ini akan mencegah kita untuk cenderung terfokus pada inti permasalahan dan gejala yang terlihat di permukaan saja.

2.1.1. Hubungan Kausal

Hubungan aksi dan reaksi atau sebab dan akibat merupakan dasar pembentuk kejadian-kejadian di alam ini. Hubungan tersebut menunjukkan adanya pengaruh dari kejadian sebelumnya kepada kejadian berikutnya. Hubungan sebab akibat tersusun atas variabel-variabel yang saling berkaitan dan menunjukkan pengaruh yang satu terhadap yang lain. Contoh dari keterkaitan tersebut dapat dilihat di Gambar 2.1. Dalam contoh tersebut, variabel yang saling berkaitan masih sedikit namun dalam sistem yang sebenarnya, variabel yang terkait sangatlah banyak sehingga hubungan kausal yang terbentuk juga menjadi kompleks. Untuk melihat hubungan kausal dari suatu sistem, kita dapat menggunakan diagram kausal (*causal loop diagram*) yang juga menunjukkan *feedback loop* (lingkaran umpan balik) dalam sistem.

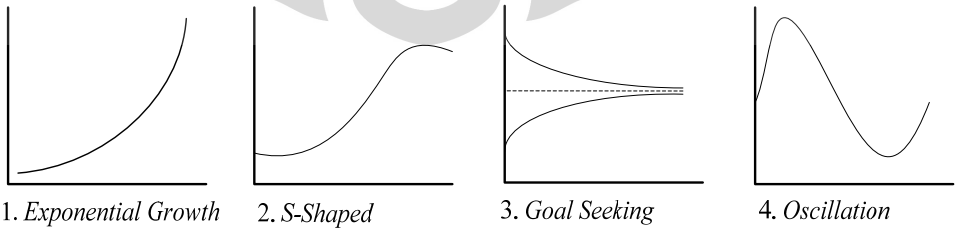
[4] Peter Vanderminden, dikutip dari http://systemdynamics.org/newsletter/2006Oct/NL10-06.htm



Gambar 2. 1 Contoh *causal loop* produktivitas

2.1.2. *Feedback* (Sebab Akibat)

Causal loop diagram (CLD) terdiri dari dua variabel atau lebih yang dihubungkan dalam lingkaran tertutup akan membentuk *feedback loop,* seperti yang tampak pada Gambar 2.1 di atas. *Feedback* yang terjadi dapat berupa *reinforcing* ataupun *balancing*. *Reinforcing feedback* terjadi ketika kondisi variabel yang saling terkait dalam suatu *loop* bergerak searah. Misalnya, ketika kesehatan menurun, maka produktivitas dalam bekerja juga akan menurun dan ketika produktivitas menurun maka akan meningkatkan stress dan akan memicu peningkatan penggunaan alkohol. *Balancing feedback* terjadi ketika kondisi variabel yang berkaitan saling berlawanan, misalnya yaitu ketika penggunaan alkohol meningkat maka tingkat kesehatan akan menurun. Kejadian atau disebut *event* itu akan membentuk suatu pola-pola tertentu dalam berbagai keadaan yang menunjukkan perilaku sistem yang ada. Ada 4 bentuk perilaku sistem, yaitu:



Gambar 2. 2 Bentuk Perilaku Sistem

2.1.3. *Delay*

Hubungan sebab akibat yang terjadi membutuhkan waktu antar variabelnya untuk berinteraksi. Waktu ini disebut *delay* yang akan menahan pengaruh dari suatu variabel terhadap variabel berikutnya dalam beberapa saat. Misalnya dalam kasus sederhana, untuk dapat naik jabatan, seorang karyawan tidak akan langsung bisa mencapainya jika tidak melalui usaha-usaha sehingga menyebabkan produktivitasnya naik. Hal ini tentu saja merupakan beberapa variabel-variabel yang akan menyebabkan waktu tunggu dari karyawan tersebut menuju jabatan yang diinginkannya. *Delay* penting untuk diperhatikan karena dapat membuat perilaku sistem menjadi tidak terprediksi dan akan mengacaukan usaha-usaha dalam mengontrol perilaku sisitem itu.

2.2. Simulasi

Kamus Oxford Amerika (1980) memberikan definisi simulasi, yaitu metode untuk mereplikasi kondisi suatu situasi dengan menggunakan sebuah model untuk kepentingan pembelajaran, percobaan maupun pelatihan. Simulasi dalam konteks sistem dinamis dapat didefinisikan sebagai imitasi dari sistem dinamis dengan menggunakan model komputer dengan tujuan untuk mengevaluasi dan meningkatkan performa sistem. Sedangkan menurut Schriber (1987), simulasi adalah pemodelan dari suatu proses atau sistem dimana model tersebut memberikan respon sepeti yang terjadi pada sistem nyata ketika berada dalam situasi yang sama.[5] Dengan kata lain, simulasi merupakan model yang merepresentasikan sistem nyata yang memiliki variabel yang kompleks dan terus berubah secara dinamis sehingga dapat digambarkan dan dimengerti secara lebih sederhana walaupun tidak akan dapat sama persis dengan sistem sebenarnya.

Simulasi berdasarkan konteks ini merupakan simulasi dengan menggunakan *software* komputer tertentu. Ketika menjalankan simulasi, kita dapat melakukan uji coba kondisi atau skenario yang dapat mewakili kemungkinan-kemungkinan yang terjadi pada sistem nyata dan mengamati

[5]Harrel, Charles, Biman K. Ghosh, dan Royce Bowden, *Simulation Using ProModel*, Mc Graw Hill, 2000, hal. 5.

akibatnya terhadap sistem. Dengan demikian, kita dapat mempelajari bagaimana mengoptimalkan keputusan yang akan diambil di sistem nyata dalam meningkatkan performa sistem tersebut.

Menurut Solberg (1988), kemampuan untuk melakukan *trial and error* untuk memperbaiki performa dari sistem manufaktur hampir tidak berguna di lingkungan karena perubahan-perubahan sangat cepat terjadi dibandingkan pengetahuan yang dapat dipelajari. Saat ini terdapat kebutuhan yang lebih besar untuk memprediksi metode formal berdasarkan sebab akibat yang diketahui. Hal inilah yang pada akhirnya menjadi alasan mengapa simulasi digunakan sebagai salah satu metode untuk memperbaiki atau meningkatkan performa suatu sistem. Terdapat beberapa karakteristik dari simulasi yang pada akhirnya membuatnya menjadi alat perencanaan dan pengambilan keputusan yang sangat kuat, yaitu:[6]

- Memetakan independensi dari sistem
- Menghitung variabilitas dari sistem
- Memiliki kemampuan untuk memodelkan banyak sistem
- Menunjukkan perilaku sepanjang waktu
- Biaya dan waktu yang dipakai sedikit dan gangguan yang terjadi lebih sedikit dibandingkan dengan sistem sebenarnya.
- Memberikan informasi dalam pengukuran performa ganda.
- Memberikan hasil yang mudah dimengerti dan dikomunikasikan.
- Menjalankan tekanan, atau bahkan waktu tunggu
- Memberikan perhatian pada desain.

[6] *Op.Cit.*Harrel, hal.7.

2.2.1. Objektif Simulasi

Seperti yang telah disebutkan sebelumnya bahwa simulasi akan memenuhi tujuan. Tujuan atau objektif tersebut harus jelas, kuantitatif, memiliki batasan waktu dan sumber daya sehingga simulasi tidak terlalu kompleks dan sulit dipahami. Menurut Nance, objektif simulasi dapat diklasifikasikan menjadi 5 kategori yaitu:[7]

1. Analisa sistem (*system analysis*): simulasi dilakukan untuk mempelajari atau meningkatkan performa sistem.
2. Edukasi dan pelatihan (*education and training*): simulasi dilakukan untuk mempelajari dan memahami konsep dan perilaku dari penerapan konsep yang ada.
3. Realisasi dan penerimaan sistem (*acquisition and acceptamce*): simulasi ditujukan untuk menjawab pertanyaan yang berhubungan dengan "Apakah sistem telah memenuhi syarat?" atau "Apakah sub sistem berkontribusi secara signifikan terhadap peningkatan performa sistem yang lebih besar?"
4. Penelitian (*Research*): simulasi dilakukan untuk melakukan tes terhadap komponen sistem atau membandingkan, membedakan, dan mengkategorikan perilakunya (*behaviour*).
5. Hiburan (*Entertainment*): simulasi model dilakukan untuk mendapatkan kesenangan atau hiburan.

Sedangkan menurut Harrel dan kawan-kawan, objektif simulasi secara umum dapat dikategorikan menjadi beberapa bagian yaitu:[8]

1. Analisa performa sistem

[7] Nance, Richard E. dan Robert G. Sargent, *Perspective on the Evolution of Simulation*, *Operation Research*, Vol. 50, No. 1, 50th *Anniversary Issue*, Januari-Februari, 2002, hal. 161.
[8] Op.Cit Harrel, hal. 83.

2. Analisa kapasitas/batasan: untuk menganalisa berapa kapasitas maksimum sistem produksi dan proses yang dapat dijalankan dalam sistem serta dimana terjadi *bottleneck*.

3. Perbandingan konfigurasi: untuk mengetahui sejauh mana suatu susunan/konfigurasi sistem dapat mencapai objektif dibandingkan dengan konfigurasi yang lain.

4. Optimisasi: untuk mengetahui apa keputusan variabel yang paling tepat untuk mencapai objektif simulasi.

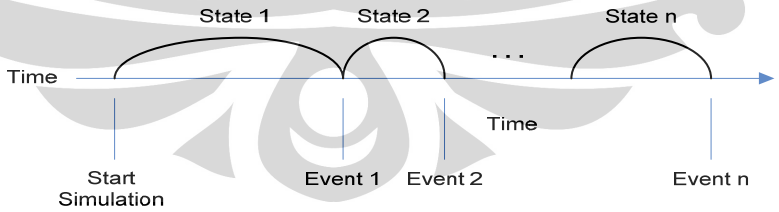
5. Analisa kepekaaan: untuk mengetahui keputusan pada variabel yang paling berpengaruh terhadap kinerja sistem dan bagaimana pengaruhnya.

6. Visualisasi

2.2.2. Jenis Simulasi

Simulasi dapat dibagi menurut representasi waktu dan kondisinya, simulasi dapat dibedakan menjadi 3 jenis, yaitu:[9]

1. *Discrete Event Simulation*: sebuah simulasi dimana perubahan kondisi sistem terjadi pada titik tertentu yaitu ketika terjadi *event* atau peristiwa.



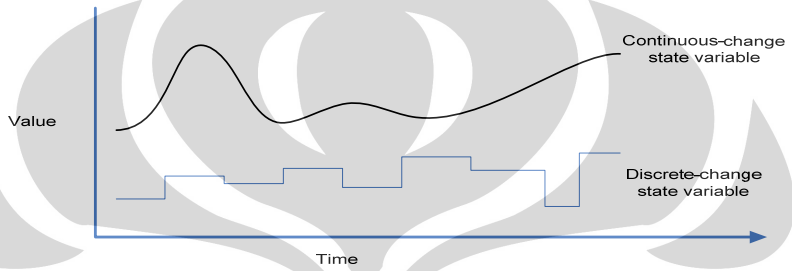
Gambar 2. 3 *Discrete event simulation*

(Sumber: Harrel et. al., *Simulation Using ProModel*, 2000)

[9] Ibid. Harel, hal.162.

2. *Continous Simulation*: pada simulasi ini, sistem berubah terhadap waktu.

3. *Hybrid Simulation*: merupakan kombinasi dari *discrete event simulaion* dan *continous simulation*, contohnya adalah pengantaran susu ke pabrik susu oleh mobil pengantar (*discrete event*) dan pada saat pengisian tanki susu dari mobil (*continous event*).



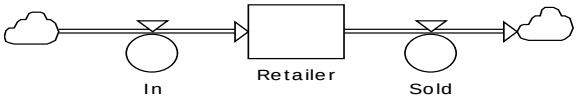
Gambar 2. 4 *Discrete event simulation* dan *continuous simulation*

(Sumber: Harrel et. al., *Simulation Using ProModel*, 2000)

2.2.3. Model dan Variabel

Simulasi dapat dijalankan jika terdapat model yang akan disimulasikan dengan benar. Model sistem dinamis tersebut terdiri dari variebel-variabel yang berbeda jenis, yaitu:

1. *Level/Stock* adalah variabel yang nilainya terus berubah berdasarkan waktu yang berjalan selama simulasi dijalankan. Variabel ini memiliki memori sehingga nilainya dapat bertambah sebanyak data yang masuk (*inflow*) dan berkurang sebanyak data yang keluar (*outflow*).



Gambar 2. 5 Contoh *level* pada sebuah model sederhana

Jika inflow diilustrasikan sebagai kran air, maka *level* adalah bak air yang menampung air. Namun agar air tidak melimpah dalam bak maka air tersebut dikelauarkan melalui saluran pembuangan air (*outflow*).

2. *Auxiliary* adalah variabel yang merupakan formulasi dan kombinasi dari informasi dan variabel lainnya yang terdapat dalam sistem. Nilai variabel ini terus berubah setiap waktu selama simulasi dijalankan dan variabel ini tidak memiliki memori.
3. *Constant* adalah variabel yang tidak akan berubah selama simulasi, namun dapat diubah juga oleh *user*.

Sebagai representasi dari sistem sebenarnya, sebuah model memerlukan batasan-batasan sehingga dapat memenuhi tujuan atau objektif yang telah ditetapkan. Oleh karena itu, dalam membuat sebuah model, kita diberikan metode umum yang akan membantu, yaitu:

1. Konseptualisasi

   Langkap pertama yang harus dilakukan sebelum membuat sebuah simulasi sistem adalah membuat konseptual model dari sebuah sistem.
2. Formulasi

   Langkah selanjutnya setelah membuat konsep model adalah membuat formulasi model tersebut dengan memasukkan data numerik dan menspesifikasi perilaku sistem.
3. Verifikasi dan validasi model

   Model yang telah dispesifikasi selanjutnya harus diverifikasi dan divalidasi dengan tujuan untuk mengetahui apakah model yang telah dibuat sudah dapat merepresentasikan sistem sebenarnya.

Adapun langkah-langkah dalam membuat model sistem dinamis adalah sebagai berikut:

1. Mengidentifikasi problem dan batasannya.
2. Mengidentifikasi variabel stok dan *flow* yang mengubah nilai stok.
3. Mengidentifikasi sumber informasi/variabel yang mempengaruhi *flow*.
4. Mengidentifikasi *feedback loop* utama dari sistem.

5. Menggambarkan *causal loop diagram* yang menghubungkan variabel stok, *flow*, dan informasi.
6. Membuat persamaan untuk menentukan *flow*.
7. Mengestimasi paramater dan kondisi awal sistem. Dapat dilakukan melalui metode statistik, pendapat ahli, data riset pasar atau sumber informasi relevan lainnya.
8. Melakukan verifikasi dan validasi terhadap model.

2.2.4. Verifikasi dan Validasi Model

Seperti yang sudah dijelaskan sebelumnya, pembuatan sebuah model bertujuan untuk merepresentasikan sistem sebenarnya. Namun, dengan tujuan mencegah kompleksitas yang tinggi terhadap model sistem, tetap diberikan batasan-batasan. Oleh karena itu, kesalahan dan perbedaan-perbedaan pasti akan terjadi ketika model dijalankan. Dalam rangka memperoleh model yang paling mendekati gambaran dan perilaku sistem yang sebenarnya, maka perlu dilakukan perbaikan-perbaikan untuk mengurangi kesalahan yang terjadi. Proses perbaikan ini disebut verifikasi dan validasi model.

Verfikasi adalah proses pengujian apakah model yang disimulasikan merefleksikan konsep model yang telah dibuat. Validasi model merupakan pengujian apakah model yang telah dibuat telah dapat merefleksikan sistem sebenarnya. Verifikasi dapat dilakukan dengan membandingkan hasil simulasi model dengan menggunakan data hasil perhitungan analitik.

2.3. Permainan

Permainan adalah sebuah bentuk karya seni atau aktivitas yang memiliki standar dan peraturan dimana perserta/pemain membuat keputusan dalam mengatur sumber daya yang dimiliki dan berkompetisi menghadapi pemain lainnya untuk mencapai tujuan permainan. Permainan selalu digunakan sejak lama sebagai salah satu metode dalam pengajaran dan pelatihan. Sebuah permainan memiliki 4 elemen penting yaitu tujuan permainan, peraturan permainan, tantangan dan interaksi antar pemain. Pada permainan tertentu, dibutuhkan unsur

kerjasama tim dalam mencapai tujuan permainan yang disebut permainan kolaborasi. Dalam permainan, para pemain dihadapkan pada situasi tertentu dan permasalahan yang harus diselesaikan. Dengan demikian, para pemain tidak hanya diberikan hiburan tetapi juga tetap mementingkan perkembangan pikiran mereka. Hal ini dapat dilihat dari kondisi para pemain yang harus menyusun strategi yang tepat sehingga dapat memenangkan permainan. Selama menjalani permainan, pemain harus memberikan waktu, energi dan komitmen mereka untuk menyelesaikannya, dan di saat yang sama mereka juga mendapatkan pengalaman dan nilai pembelajaran. Permainan yang membutuhkan daya pikir dan strategi seperti ini disebut juga *serious play*.

Permainan dapat dibedakan menjadi beberapa tipe berdasarkan media yang digunakan, yaitu:

1. Permainan lapangan (olahraga)

   Permainan olahraga adalah permainan yang sangat umum dan jenisnya bervariasi. Permainan ini bisa dilakukan di lapangan terbuka atau tertutup.

2. Permainan komputer atau *video games*

   Sebuah permainan komputer dikendalikan oleh mikroprosesor sehingga dapat menampilkan simulasi dari suatu aktivitas dimana untuk memainkannya para pemain harus menggunakan alat tertentu seperti *keyboard*, *mouse*, *joystick*, dan lain-lain. Pada permainan komputer yang lebih kompleks, para pemain dapat melakukan apa saja selama masih dalam batasan peraturan permainan dan area virtual, contohnya *Dinner Dash*.

3. *Board game*

   Board game adalah permainan yang dilakukan dengan menggunakan sebuah papan/*board* yang berisi peraturan permainan serta keterangan mengenai status, sumber daya, dan kemajuan yang dimiliki setiap para pemain. Biasanya untuk menjalankan atau menentukan giliran pemain digunakan dadu atau kartu. Contoh permainan yang menggunakan metode

ini adalah permainan simulasi rantai pasok seperti *Beergame* atau permainan manajemen produksi.

4. *Card game*

   Permainan ini menggunakan 1 dek kartu sebagai alat utamanya. Contoh permainan kartu antara lain *Uno, Rook, Bridge* dan lain-lain.

Berdasarkan jumlah peserta permainan, maka permainan dapat diklasifikasikan sebagai berikut:

1. *Single-player game*, yaitu permainan yang hanya bisa dimainkan oleh satu orang pemain. Pemain hanya berkompetisi menghadapi lingkungan permainan, lawan buatan (misalnya komputer), waktu atau kesempatan. Contoh permainan *single player adalah* permainan pada ponsel.
2. *Multi-player game*, yaitu permainan yang mengikutsertakan sekian banyak pemain (lebih dari satu orang) dan mereka berkompetisi menghadapi tantangan dan menyelesaikan misi yang diberikan untuk memenangkan permainan. Jenis permainan ini misalnya *multi- player game online* yang memberikan lebih banyak tantangan dan fitur tambahan seperti *chatting* antar pemain dan lain sebagainya, contohnya adalah permainan MMORPG (*Massively Multi-player Online Role-Playing Game*).

Desain permainan yang menarik dan tidak kaku juga tak kalah penting untuk dipertimbangkan dalam membangun sebuah permainan, meskipun apabila permainan tersebut ditujukan untuk edukasi. Tujuannya tentu saja untuk menarik minat para pemain sehingga mereka tidak cepat merasa jenuh selama bermain. Menurut Duke, terdapat 9 langkah dasar dalam mendesain sebuah permainan, yaitu:[10]

1) Membuat spesifikasi/kriteria untuk desain permainan
2) Membuat skema umum yang menggambarkan permasalahan
3) Memilih komponen permasalahan yang akan dimainkan

[10] Richard D. Duke, *A Paradigm For Game Design*, *Simulation & Games*, Vol. 11, No. 3, September 1980, hal. 365.

4) Merencanakan permainan dengan Komponen Sistem/Matriks elemen Permainan
5) Menggambarkan isi setiap sel matriks
6) Mencari daftar permainan sebagai ide untuk merepresentasikan setiap sel matriks
7) Membangun permainan
8) Mengevaluasi permainan (mengacu pada kriteria yang ditentukan pada langkah 1)
9) Menjalankan permainan dan memodifikasinya

2.4. Permainan Simulasi

Permainan simulasi merupakan metode yang telah lama digunakan dalam pengajaran dalam berbagai bidang, salah satunya adalah manajemen. Banyak universitas yang mengadopsi metode ini untuk meningkatkan baik kualitas maupun produktivitas pengajaran mereka dengan menggunakan metode ini. Metode permainan simulasi sangat menarik digunakan karena konsep yang telah dipelajari akan lebih mudah untuk dipahami ketika para peserta dapat berinteraksi langsung dalam proses pengambilan keputusan manajerial. Dengan demikian, peserta dapat memperdalam wawasan dan menambah pengalaman mengenai aplikasi ilmu pengetahuan yang terkandung dalam permainan simulasi tersebut. Salah satu kelebihan dari permainan simulasi adalah para peserta dapat melakukan eksperimen keputusan dan strategi tanpa perlu mengkhawatirkan resiko biaya ataupun bangkrut, karena semua hanya simulasi. Dari hal itulah, mereka dilatih untuk mengasah kemampuan dan pengetahuan yang telah mereka dapat secara konsep. Selain itu peserta permainan segera mendapat *feedback* dari setiap keputusan yang mereka ambil sehingga mereka semakin termotivasi untuk menemukan strategi terbaik dalam menghadapi permainan simulasi tersebut. Nilai-nilai positif yang diperoleh dari permainan simulasi sebagai metode pengajaran adalah sebagai berikut:[11]

[11]Enciso, Ricardo Zamora, *Simulation Games, A Learning Tool*, ISAGA 2001 *Proceedings Conference* (*International Simulation and Gaming Association*), hal. 6-7.

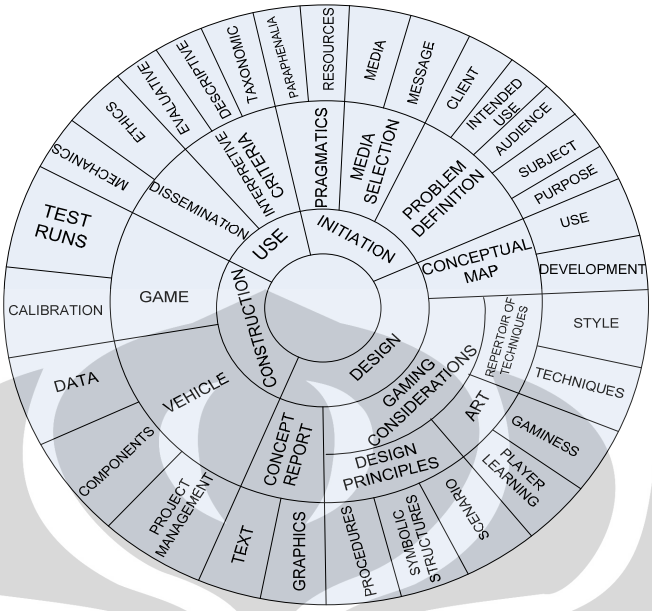
- Ketika siswa diminta untuk mempelajari sesuatu, mereka akan merasa terpaksa melakukannya sehingga ilmu yang diserap sedikit. Melalui permainan simulasi yang menyenangkan, siswa akan merasa lebih tertantang sehingga dengan suka rela belajar dari pengalaman yang didapat dari simulasi.
- Menurut Festinger (1957), langkah pertama dalam mengubah kebiasaan, keyakinan dan persepsi seseorang adalah dengan membiarkannya mengalami keraguan terhadap validitas sebuah mental model yang mereka miliki. Konflik dan kekacauan yang terjadi dalam permainan menimbulkan keraguan dalam pikiran para pemain akan pola pikir mereka selama ini sehingga mereka mau menerima dan melakukan perubahan.
- Pada permainan simulasi sistem dinamis, para pemain dipancing untuk membuat peta sebab akibat (*causal loop diagram*) dari sistem permainan. Di sini para pemain didorong untuk melihat permasalahan dengan sudut pandang yang luas yakni melihat gambaran globalnya sehingga mental model mereka semakin berkembang.
- Di dunia nyata, semua proses membutuhkan waktu yang tidak sedikit, sehingga kesempatan belajar dari pengalaman pun hanya sedikit. Dalam permainan simulasi, waktu dan ruang yang diperkecil semakin memperbesar peluang untuk belajar dari pengalaman selama bermain dan mempercepat proses pembelajaran tersebut.
- Seperti disebutkan sebelumnya, pada permainan simulasi pemain dapat mengembangkan kemampuannya dan melakukan berbagai percobaan dalam membuat keputusan, menyusun strategi, dan kebijakan baru tanpa resiko.
- Setelah bermain, semua peserta permainan berkumpul untuk berbagi pengalaman dan mendiskusikan mental model masing-masing.

Dengan berkembangnya teknologi dan program komputer untuk simulasi dan pemodelan, permainan simulasi komputer semakin diminati dan berkembang di berbagai kalangan. Terdapat 4 fase dalam mendesain sebuah permainan simulasi, yaitu:

1. *Initiation phase* – pada fase ini, para desainer menganalisa definisi masalah, pemilihan media permainan, dan tujuan permainan termasuk

pesan pokok dan tujuan komunikasi apa yang ingin disampaikan dari permainan yang akan dibuat. Batasan biaya dan waktu juga menjadi pertimbangan penting. Untuk itu, desainer perlu mengumpulkan data terkait baik melalui kuesioner, diskusi/wawancara maupun pelatihan, sehingga mereka dapat menghasilkan permainan simulasi layak dan berkualitas.

2. *Design phase* – fase berikutnya adalah fase desain dimana para desainer membuat peta konsep yang berisi gagasan yang akan dikembangkan untuk permainan. Konsep juga mencakup komponen dasar sebuah permainan yaitu skenario, prosedur dan struktur simbolis yang dibuat sebagai kerangka bentuk permainan dan aktivitas apa saja yang harus dilakukan untuk menyelesaikan permainan.
3. *Construction phase* – setelah desain selesai, maka permainan simulasi sudah bisa dibangun. Selama proses konstruksi, desainer membuat program permainan dan memasukkan data-data yang dibutuhkan. Setelah itu program dijalankan untuk diuji dan dikalibrasi. Apabila proses konstruksi permainan simulasi telah selesai, maka para operator dilatih untuk menggunakan permainan sehingga permainan dapat dioperasikan.
4. *Use phase* – pada fase ini desainer bertanggung jawab untuk memberikan kriteria pemahaman mengenai klasifikasi permainan kepada operator dan para pemain. Di dalamnya mencakup gambaran umum permainan, tujuan permainan serta prosedur evaluasi untuk mengetahui apakah peserta dapat meraih pesan dan tujuan yang disampaikan dari permainan simulasi. Selain itu desainer juga bertanggung jawab untuk memberikan mekanisme pengenalan dan penyebaran permainan kepada peserta termasuk pelatihan bagi operator.



Gambar 2. 6 Proses desain permainan

(Sumber: Richard D. Duke, *Game Design Process*, 1980)

2.4.1. Objektif Permainan Simulasi

Seperti halnya simulasi, permainan simulasi juga memiliki objektif atau tujuan. Selain digunakan dalam metode pembelajaran, menurut Rausch permainan simulasi juga memiliki 6 kategori tujuan sebagai berikut:[12]

1. Peramalan masa depan yaitu penggunaan simulasi untuk memperkirakan dampak dari suatu kebijakan.
2. Penelitian lain, misalnya pada bidang genetika untuk melihat kemungkinan perpaduan dua jenis gen berbeda.
3. Perencanaan, contohnya pada perencanaan kebijakan perusahaan, atau perencanaan jumlah produksi.
4. Desain, misalnya pada bidang seismologi, untuk melihat dampak dari gempa berkekuatan tertentu pada suatu daerah dengan karakteristik tertentu.

[12]Rausch, E. 1994, *Simulation and Games in Futuring and Other Uses*, hal. 5.

5. Alat bantu pembelajaran – contohnya pada penggunaan permainan *Bussines Game* seperti *Beergame* dan *Executive Decision Game*.
6. Hiburan – contohnya pada permainan PC seperti *SimCity, Diner Dash* dan sebagainya.

2.4.2. Jenis Permainan Simulasi

Permainan simulasi sebagian besar digunakan dalam bidang manajemen produksi. Oleh karena itu, untuk memudahkan dalam mencapai tujuan pembelajaran bagi akademisi atau pun perusahaan, maka permainan simulasi ini pun dibagi menjadi beberapa kategori dan contoh permainannya, sebagai berikut:[13]

1. Permainan dengan pengambil keputusan tunggal. Permainan ini dilakukan oleh satu orang yang biasanya dilakukan langsung berhadapan dengan komputer. Contoh dari permainan ini adalah *TRAIN-F Simulation Game, FMS Design Game, OPT_SIM dan DIC_XIM.*
2. Permainan dengan pusat keputusan (tim perencanaan). Permainan ini dilakukan dalam tim dengan keputusan terpusat. Contoh permainannya adalah: *LOGTIME, The Ruler Game for Production Group, Teamwork Game,* dan *Simulation Aided Planning of Work Structures*.
3. Permainan dengan pengaruh multifungsi. Permainan ini adalah permainan yang saling berintegrasi antarbagian dalam satu sistem. Semua bagian itu memiliki pengaruh yang signifikan dalam pengambilan keputusan manajerial. Contoh permainan ini adalah: *LEGO Truck Game, CIM game, The Logi Game* dan *Beer Distribution Game* termasuk dalam kategori ini.

2.5. *Web-based Simulation*

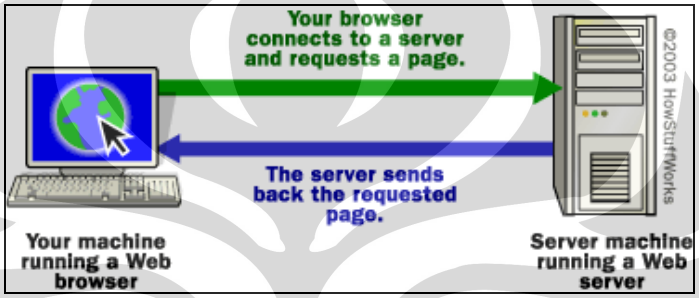
Jaringan internet (web) telah berkembang begitu pesat sejak tahun 1990-an dan memberi pengaruh kuat terhadap aplikasi simulasi, animasi dan visualisasi. Melalui jaringan internet, kita akan memperoleh integrasi yang lebih mudah dan lebih cepat antara simulasi, animasi dan teknik serta perangkat visualisasi.

[13] Riis. O. Jens, *Simulation Games and Learning in Production Management,* Chapman & Hall,1995. hal.7-10.

Simulasi berbasis aplikasi web merupakan *tool* rekayasa baru yang memiliki pengaruh yang sangat besar terhadap ilmu rekayasa. Penggunaan teknologi web dalam studi simulasi sendiri telah dimulai sejak tahun 1995 dengan menggunakan program CGI (*Common Gateway Interface*) dan penelitian ini kemudian dikembangkan dan dilanjutkan dengan menggunakan program Java. Simulasi berbasis aplikasi web pada dasarnya adalah penggabungan konsep simulasi tradisional dan simulasi komputer. Pada simulasi tradisional, peserta simulasi saling berinteraksi secara langsung di suatu ruangan dan menggunakan media tertentu sebagai alat simulasi, misalnya pada permainan simulasi *Beer Game* dimana media yang digunakan adalah papan permainan, koin, dan kartu untuk pemesanan barang. Konsep kedua yaitu simulasi komputer adalah bentuk simulasi dari suatu model yang dijalankan oleh program atau *software* simulasi sehingga simulasi dapat dilakukan oleh satu orang saja. Dengan menggabungkan kedua konsep ini, maka peserta simulasi tidak lagi harus bertatap muka untuk ikut serta dalam permainan simulasi, melainkan terhubung secara virtual melalui jaringan internet. Kelebihan yang didapat dengan mengaplikasikan simulasi dalam bentuk web antara lain:

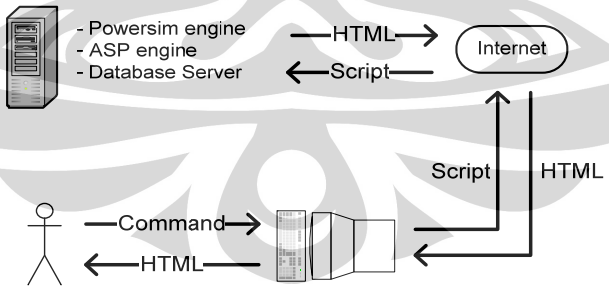
1. Kemampuan akses yang luas, dimana *user* dapat mengakses banyak *platform* tanpa harus mengkompilasi kembali data tersebut maupun menginstal *hardware* dan software-nya.
2. Adanya kontrol akses yang dapat mencegah terjadinya perubahan dan duplikasi yang tidak diharapkan terhadap model aslinya dengan sistem *password*.
3. Pemeliharaan yang efisien; memungkinkan dilakukannya modifikasi dan mudah mendistribusikannya, serta dapat mengurangi potensi *error* ketika melakukan *update* dan distribusi model melalui server.
4. Integrasi yang lebih baik; menampilkan *interface* secara instan menggunakan browser web yang ada, hanya membutuhkan browser web untuk menampilkan program, dan memungkinkan terjadinya komunikasi dan interaksi antar *user* melalui web.

Secara sederhana, di dalam jaringan internet terdapat 2 komponen utama yaitu *browser* dan *server*. *Browser* adalah komputer/perangkat yang digunakan oleh *user* untuk menarik data atau mengakses data melalui internet yang kemudian dikirim ke *server*. Setelah *server* mengolah permintaan dari *browser*, data yang diminta lalu dikirim melalui jaringan internet dan diteruskan hingga tampil di layar *browser*. Secara sederhana skemanya dapat dilihat pada gambar berikut.



Gambar 2. 7 Skema penarikan data melalui internet

Demikian pula halnya dengan konstruksi simulasi model berbasis web pada penelitian ini. Skemanya dapat dilihat pada gambar 2.8.



Gambar 2. 8 Konstruksi *web based simulation*

Ketika *user* berhadapan dengan *web interface* dari suatu simulasi model, *user* akan meng-*input* data dan melakukan berbagai perintah. *Input* data dan perintah

tersebut kemudian dikirim oleh *browser* ke *server* untuk disimpan dalam *database* dalam bentuk *script* ASP dengan fungsi SQL. Fungsi SDK (*Software Development Kit*) dari *simulator engine* yang ada di *server* akan menarik *input* dari *database*. Setelah menerima *input* dan perintah, *simulator engine* akan menjalankan simulasi pada model lalu menyimpan kembali *output* hasil simulasi ke dalam *database* dengan fungsi SDK dan SQL. *Server* kemudian akan mengirimkan *output* simulasi tersebut ke *browser* untuk ditampilkan di layar monitor dalam bentuk halaman HTML. Dari uraian singkat ini, dapat diketahui bahwa untuk membangun simulasi sistem dinamis berbasis web, diperlukan beberapa komponen penting yaitu:

1. Program simulasi (*simulation engine*) –berfungsi untuk menjalankan model dan mengolah *input*.
2. *Script* pemrograman – berfungsi untuk menerjemahkan hasil simulasi dalam bentuk halaman HTML.
3. *Database engine* – berfungsi sebagai bank data yang menyimpan dan memberikan semua data yang berhubungan dengan permainan simulasi.
4. *Image generator* – merupakan *script* pemrograman untuk membuat grafik dalam bentuk HTML di halan web dengan menggunakan data yang ditarik dari *database*.

2.6. *Web Interface*

Di saat *user* atau *browser* ingin mengakses data dari *server* melalui internet, data tersebut tidak langsung terhubung dengan komputer *user*. *User* akan memasuki satu halaman web yang menjadi penghubung dan jembatan antara *user* dan data di *server*. Halaman web ini disebut juga sebagai *web interface*. Karena berfungsi sebagai penghubung, maka *web interface* harus dirancang sedemikian rupa sehingga dapat memudahkan proses penarikan dan penyimpanan data dari dan ke *server*. Kriteria *web interface* yang baik adalah:

1. Memiliki proses navigasi yang simpel, singkat dan mudah dipahami. Proses navigasi terdiri dari rangkaian instruksi yang mengarahkan *user* untuk melakukan suatu aktivitas yang berkaitan dengan *web interface* tersebut.
2. Memberikan visualisasi yang tepat. Visualisasi yang baik haruslah mempertimbangkan kontras warna yang digunakan baik pada *background*, tulisan maupun gambar.
3. Menyediakan menu yang fungsional dan tepat guna. Tampilan menu dapat diperindah dengan fungsi *icon* maupun animasi, namun jangan berlebihan sehingga dapat mendestruksi konsentrasi *user* ketika mengakses data.
4. Memiliki jalur akses yang lancar dan tentu saja terkoneksi dengan baik terhadap *server*.

2.7. *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah bahasa standar yang digunakan untuk memvisualisasikan, merancang dan mendokumentasikan *blueprint* dari suatu model sistem piranti lunak sehingga mudah dipahami. UML dapat digunakan untuk merancang model untuk semua jenis aplikasi piranti lunak dimana aplikasi tersebut dapat dijalankan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun.

Tabel 2. 1 Konsepsi Dasar UML

Major Area	View	Diagrams	Main Concept
structural	static view	class diagram	class,association,generalization, dependency,realization,interface
	use case view	use case diagram	use case,actor,association,extend, include,use case generalization
	implementation view	component diagram	component,interface,dependency, Realization
	deployment view	deployment diagram	node,component,dependency, Location
dynamic	state machine view	statechart diagram	state,event,transition,action
	activity view	activity diagram	state,activity,completion transition, fork,join
	interaction view	sequence diagram	interaction,object,message, Activation
		collaboration diagram	collaboration,interaction,collaboration role,message
model management	model management view	class diagram	package,subsystem,model
extensibility	all	all	constraint,stereotype,tagged values

Sumber: Wahono, Pengantar Multi Agent Sistem

UML terdiri dari sejumlah elemen grafik yang menyatu membentuk diagram. Diagram-diagram tersebut digunakan untuk memetakan model dari suatu sistem. Beberapa jenis diagram UML adalah:

1. *Class diagram* – Diagram *class* menggambarkan keadaan (atribut/properti) suatu sistem serta memberikan fungsi dan metode untuk memanipulasi keadaan tersebut.
2. *Statechart diagram* – Diagram ini menggambarkan perubahan keadaan antar *state* suatu objek sistem sebagai akibat dari stimulan yang diterima. *Statechart* umumnya menggambarkan *class* tertentu.
3. *Use case diagram* – Diagram ini menggambarkan perilaku sistem dari sudut pandang pengguna atau dengan kata lain mendeskripsikan fungsionalitas yang diharapkan dari sebuah sistem. *Use case diagram* digunakan untuk menyusun kebutuhan sistem, mengkomunikasikan rancangan dengan klien, dan merancang skenario tes untuk semua fitur yang ada dalam sistem.

4. *Activity diagram* – Diagram *activity* menggambarkan aliran aktivitas yang terjadi di dalam sistem, mencakup bagaimana setiap aliran bermula, keputusan yang mungkin terjadi hingga bagaimana aliran tersebut berakhir, serta menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.
5. *Sequence diagram* – Diagram yang memiliki 2 dimensi yaitu dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Diagram ini digunakan untuk menggambarkan interaksi antar objek di dalam sistem maupun di sekitar sistem berupa pesan yang digambarkan setiap waktu.
6. *Component diagram* – Digram ini menggambarkan struktur dan hubungan antar komponen piranti lunak yang merupakan modul yang berisi kode program. Komponen biasanya terbentuk dari beberapa *class* atau komponen-komponen yang lebih kecil.
7. *Deployment diagram* – *Deployment diagram* digunakan untuk menggambarkan proses detail dari pemasangan komponen dalam infrastruktur sistem, di mana komponen akan diletakkan pada mesin, server atau piranti keras lainnya, bagaimana spesifikasi server yang dibutuhkan, bagaimana kemampuan jaringan pada lokasi tersebut, dan hal-hal yang berkaitan dengan fisik lainnya.

2.7.1. *Class Diagram*

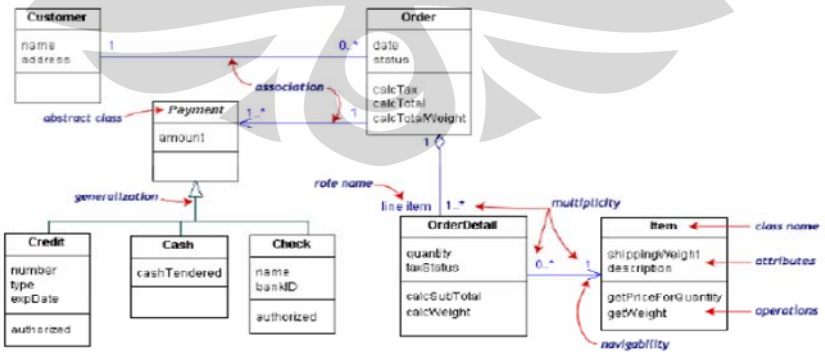
Semua hal yang ada di sekitar kita memiliki atribut/sifat dan perilaku yang berbea-beda. Perilaku ini dapat kita anggap sebagai rangkaian operasi. Dalam kehidupan sehari-hari, kita juga secara alami mengkategorikan setiap hal seperti kegiatan otomotif, barang perabotan, dan lain-lain. Kategori yang kita buat itu sama halnya dengan *class*. *Class* adalah sebuah kategori atau sekelompok hal yang memiliki atribut dan perilaku yang sama.

Diagram *class* berisi gambaran dari struktur dan deskripsi *class*, *package*, dan objek serta hubungan satu sama lain. 3 komponen utama dari *class* adalah yaitu nama, atribut dan metode/operasi. Atribut dan metode dapat memiliki sifat

private yaitu tidak dapat dipanggil dari luar *class* yang bersangkutan, *protected* yaitu hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya, dan *public* yaitu dapat dipanggil oleh siapa saja. *Class* juga bisa digunakan sebagai implementasi dari sebuah *interface* yang merupakan *class* abstrak yang hanya memiliki metode. Tujuannya adalah agar *interface* dapat mendukung resolusi metode pada saat *run-time*.

Seperti disebutkan sebelumnya, dalam *class diagram* terdapat beberapa jenis hubungan antar *class*, yaitu:

1. Asosiasi – Hubungan statis antar *class* yang umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain atau *class* yang harus mengetahui eksistensi *class* lain. Panah menunjukkan arah *query* antar *class*.
2. Agregasi – Hubungan yang menyatakan suatu *class* terdiri dari beberapa *class* lain.
3. Pewarisan – Hubungan hirarki antar *class*, artinya *class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metode dari *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis – Rangkaian pesan yang dikirim dari satu *class* ke *class* lain. Hubungan dinamis juga dapat digambarkan dengan menggunakan *sequence diagram*.



Gambar 2. 9 Contoh *class diagram*

2.8. *Script Active Server Pages* (ASP)

Web yang interaktif adalah web yang dapat berkomunikasi dengan pengunjungnya. Perkembangan jenis web sekarang ini menuju ke jenis web seperti ini. Web yang dinamis merupakan web yang dapat mengubah tampilannya dengan mudah sehingga selalu menyajikan informasi yang *up to date* bagi pengunjung. Web yang interaktif dan dinamis hanya mungkin dibuat dengan bantuan pemrograman web. ASP merupakan salah satu pemrograman web. Pemrograman web menggunakan basis pada proses interpreter (penerjemahan perintah) dan diberikan kebebasan untuk memilih dan menggunakan editor pengetikan kodenya. Pengetikan kode ini sering disebut *web scripting*. *Web scripting* adalah file teks yang berisi *script* (kode yang digunakan untuk pemrograman web) yang akan diterjemahkan untuk dideteksi kevalidannya setiap kali *script* itu diakses atau dipanggil sehingga pada akhir prosesnya akan dihasilkan output bertipe HTML yang ditampilkan melalui *web* browser. *Script* pemrograman web terdiri dari 2 jenis berdasarkan siapa yang mengolahnya, yaitu *client side* dan *server side*. *Client side* artinya *script* yang diberikan oleh user diterjemahkan dan diolah langsung oleh browser. Kelemahannya, *script* jenis ini tidak aman karena kode-kode *script* yang dibuat dapat dengan mudah dilihat oleh orang lain. Sedangkan *server side* artinya *script* pemrograman dikirim oleh browser kepada server untuk diterjemahkan dan diolah, lalu dikirim kembali kepada browser dalam bentuk HTML murni. *Script* jenis ini aman karena kode-kode *script* ASP yang telah disusun tidak dapat dilihat oleh orang lain.

ASP atau *Active Server Pages* adalah *script* pemrograman *server side* yang bersifat *Open Application Environment*, artinya dengan menggunakan ASP, pemrogram dapat menggabungkan kode-kode HTML, file teks, *script* pemrograman dan komponen ActiveX menjadi satu kesatuan di dalam aplikasi web yang telah dibangun. Dengan teknologi ASP, kita dapat membuka halaman *web site* yang dinamis, interaktif dan atraktif. Untuk menggunakan ASP ada beberapa persyaratan yang harus kita penuhi, yaitu: [14]

[14] Wahidin, ASP Untuk Orang Awam, Maxikom, Pambang, 2004, hal. 1.

1. Server Web

   Server web bertugas untuk mengolah *script* ASP sebelum ditampilkan di dalam browser. Bagi pengguna Microsoft Windows NT, Windows 2000 Professional dan Windows 2000 Server, Anda dapat menggunakan IIS (*Internet Information Service*) sebagai server webnya. Sedangkan bagi pengguna Windows dapat menginstal PWS (*Personal Web Server*).

2. Teks Editor

   ASP merupakan *script* pemrograman yang 100% teks, oleh sebab itu untuk menyusun script ASP Anda dapat menggunakan program teks editor apapun, seperti Notepad, Text Edit, Microsoft Script Editor dan lain-lain.

3. *Web Browser*

   Fungsi *web browser* adalah untuk menampilkan script ASP dalam format HTML. Anda dapat menggunakan *web server* manapun, seperti Internet Explorer, Mozilla, Opera, dan lain-lain.

Microsoft menawarkan ASP dengan sejumlah kelebihan berikut:[15]

1. Dengan ASP, para *web developer* lebih mudah membangun halaman web sehingga tampak lebih indah, lebih dinamis dan lebih interaktif.
2. Dapat dibangun transaksi yang aman via web, berbagai aplikasi berbasis web dan berbagai aplikasi server.
3. ASP dapat berinteraksi dengan database sehingga memudahkan untuk membangun suatu database yang berbasis web.

2.8.1. Penulisan *Script* ASP

ASP bukanlah bahasa pemrograman, karena ASP masih menggunakan teknologi *script programming* lain, yaitu *Vbscript* dan *Jscript*, pengembangan *Javascript* oleh *Microsoft*. Jika berdiri sendiri, kedua *script* ini berjenis *client side*,

[15] Newman, Frans, Pemrograman *Client/Server* dengan ASP,Elex Media Komputindo,Kelompok Gramedia, 2001, hal. 9.

namun ketika dimanfaatkan sebagai ASP *programming* maka *script* tersebut akan berbentuk *server side.* Yang menandakan sebuah *script* ASP adalah ekstensi .asp yang wajib ditambahkan pada setiap file ASP. Bentuk penulisan *script* ASP selalu diawali dengan tag pembuka <% dan ditutup dengan %>, contoh:

```
<HTML>
<%
...
script ASP
...
%>
</HTML>
```

atau

```
<SCRIPT LANGUAGE=”VBSCRIPT” RUNAT=”SERVER”>
…
script ASP
…
</SCRIPT>
```

2.8.2. *Object* ASP

Object merupakan sesuatu yang disusun secara hierarki. Seperti halnya pada *Vbscript,* ASP juga memiliki *object* bawaan yang dapat langsung digunakan. *Object* tersebut adalah *response object, request object, server object, session object, application object,* dan *error object.*

2.8.2.1. *Response Object*

Response object berfungsi untuk mengirimkan data-data (output) dari server untuk browser. Sintaksnya adalah:

Response.[collection|property|method] “pernyataan”

Tabel 2. 2 *Response Object Collection*

Collection	Description
Cookies	Mengatur nilai *cookie* yang menunjukkan kegiatan browser

Tabel 2. 3 *Response Object Property*

Property	Description
Buffer	Menentukan apakah halaman output akan ditampung oleh server atau tidak sebelum ditampilkan pada browser
CacheControl	Mengatur apakah *proxy server* dapat menerima output yang dihasilkan oleh ASP
Charset	Melampirkan nama karakter pada *content type*
ContentType	Mengatur *content type* HTTP untuk *response object*
Expires	Mengatur lamanya halaman dapat diterima oleh browser sebelum waktunya habis
ExpiresAbsolute	Mengatur tanggal dan waktu kapan halaman yang diterima browser diyatakan *expire*
IsClientConnected	Megidentifikasi ketika *client* terputus dari *server*
Pics	Melampirkan nilai pada label PICS
Status	Menentukan nilai status baris yang dikembalikan oleh server

Tabel 2. 4 *Response Object Collection*

Method	Description
AddHeader	Menambahkan subtittle HTTP yang baru dan nilai dari respon HTTP
AppendToLog	Menambahkan string pada akhir catatan masuk server
BinaryWrite	Menuliskan data output tanpa konversi karakter apapun
Clear	Menghapus output HTML yang ditampung
End	Menghentikan proses pada *script* dan menampilkan hasil yang telah diproses hingga saat itu
Flush	Mengirimkan output HTML yang ditampung dengan segera
Redirect	Mengantarkan *user* ke URL berbeda
Write	Menulis string tertentu pada output

2.8.2.2. *Request Object*

Fungsi *Request object* adalah untuk menangkap/mengambil nilai dari browser. Sintaksnya adalah:

Request.[collection|property|method] (“variabel”)

Tabel 2. 5 *Request Object*

Nama	Jenis	Keterangan
Cookies	Collection	Digunakan untuk mengambil nilai *cookie* yang dikirim oleh *browser.*
Form	Collection	Untuk mengambil nilai form yang dikirim dari *browser* menggunakan metode *post.*
QueryString	Collection	Untuk mengambil nilai dati HTTP query string menggunakan metode *get* .
ServerVariables	Collection	Digunakan untuk mengambil nilai dari variabel *server* .
TotalBytes	Property	Menghitung jumlah bytre yang dikirimkan oleh *browser* .

Untuk mendapatkan keterangan dari pengunjung web melalui variabel *server*, seperti alamat IP *client*, nama server, *port* yang digunakan, dan sebagainya digunakan Collection ServerVariabel pada *Request object*. Beberapa variabel tersebut dapat dilihat pada tabel berikut.

Tabel 2. 6 Variabel *Server*

Nama	Fungsi
CONTENT_LENGTH	Mendapatkan jumlah *byte* yang dikirim oleh *client*
LAST_MODIFIED	Waktu terakhir halaman web di *update*
REMOTE_ADDR	Alamat IP *client*
REMOTE_HOST	Alamat ipIP *server*
REMOTE_USER	Nama *client*
SERVER_NAME	Nama *server*
SERVER_PORT	Nomor *port* yang digunakan
SERVER_PROTOCOL	Nama dan versi protokol yang digunakan
SERVER_SOFTWARE	Nama dan versi *software* yang digunakan
HTTP_HOST	Nama komputer *host*
HTTP_USER-AGENT	Nama dan versi *software browser/client*

2.8.2.3. *Server Object*

Server object digunakan untuk mengakses *property* dan *method* pada *server*. Sintaksnya adalah:

Server.property|method

Tabel 2. 7 *Server Object Property*

Property	Description
ScriptTimeout	Mengatur atau mengembalikan waktu maksimum sebuah *script* dapat dijalankan sebelum diakhiri

Tabel 2. 8 *Server Object Method*

Method	Description
CreateObject	Membuat instan objek server
Execute	Menjalankan sebuah file ASP dari file ASP yang lain
GetLastError()	Mengembalikan *error object* yang menggambarkan kondisi eror yang terjadi
HTMLEncode	Menggunakan kode HTML pada string tertentu
MapPath	Memetakan sumber data
Transfer	Mengirim semua informasi yang dibuat pada satu file ASP ke file ASP lain.
URLEncode	Menggunakan kode URL pada string tertentu

2.8.2.4. *Session Object* dan A*pplication Object*

Application object bekerja pada lingkungan aplikasi web. Dengan menggunakan *application object*, kita dapat membuat variabel global yang dapat dimanfaatkan oleh seluruh *user*, misalnya menghitung jumlah *user* yang sedang mengakses halaman web. Sedangkan *session object* bertugas membuat *session* untuk masing-masing *user* serta dapat digunakan untuk mengatur hubungan antara halaman web yang satu dengan yang lainnya. Sebagai contoh, kita dapat mewajibkan setiap *user* untuk *login* terlebih dahulu dengan bantuan *session object* sebelum mengakses aplikasi web yang dianggap penting.

Tabel 2. 9 *Session Object* dan *Application Object*

Nama	Jenis	Keterangan
Lock	Method	Mengunci *user* lain mengadakan perubahan terhadap variabel objek *application*
Unlock	Method	Membebaskan penguncian terhadap variabel objek *application*
OnStart	Events	Kejadian sat pertama kali *user meminta layanan aplikasi*
OnEnd	Events	Kejadian saat *user* memutuskan/keluar dari aplikasi
Abandon	Property	Digunakan untuk membebaskan *user* dari sebuah *session*

Salah satu penggunaan *application object* dan *session object* adalah pada file global.asa. File ini terdapat di dalam *root* sebuah aplikasi web yang digunakan untuk mengatur *events* atau kejadian saat *application*/*session object* pertama kali diaktifkan atau saat *application*/*session object* dihentikan. File ini juga mencakup semua aplikasi web sehingga apabila terjadi pengaturan pada file ini, misalnya variabel, maka seluruh aplikasi juga akan ikut berubah. Sintaks penulisan *events* untuk *application object* dan *session object* di dalam file global.asa adalah:

```
<SCRIPT LANGUAGE=”VBScript” RUNAT=”Server”>
Sub Application_OnStart
        Application(“namaVariabel”)
End Sub
Sub Application_OnEnd
        Application(“namaVariabel”)
End Sub
Sub Session_OnStart
        Session(“namaVariabel”)
End Sub
Sub Session_OnEnd
        Session(“namaVariabel”)
End Sub
</SCRIPT>
```

2.8.2.5. *Error object*

Error object berfungsi untuk menampilkan informasi detil mengenai *error* yang terjadi dalam *script* pada halaman ASP. *Error object* muncul ketika *Server.GetLastError* dipanggil, jadi informasi *error* hanya dapat diakses dengan menggunakan metode *Server.GetLastError*.

Tabel 2. 10 *Error Object Property*

Property	Keterangan
ASPCode	Mengembalikan kode eror yang disebabkan oleh IIS
ASPDescription	Mengembalikan deskripsi detil dari eror yang terjadi (jika eror tersebut berhubungan dengan ASP)
Category	Mengembalikan sumber eror (apakah disebabkan oleh ASP?Oleh bahasa *script* ?Oleh *object* ?)
Column	Mengembalikan posisi kolom di dalam file yang menyebabkan terjadinya eror
Description	Mengembalikan deskripsi singkat dari eror
File	Mengembalikan nama file ASP yang menyebabkan eror
Line	Mengembalikan nomor baris tempat ditemukannya eror
Number	Mengembalikan kode standar eror COM untuk eror yang terjadi
Source	Mengembalikan kode sumber pada baris tempat terjadinya eror

2.9. *Structured Query Language* (SQL)

SQL adalah sebuah bahasa pemrograman komputer dengan standar ANSI (*American National Standards Institue*) yang ditujukan untuk mengakses dan memanipulasi sistem *database*. *Script* SQL digunakan untuk menarik dan memperbarui data pada sebuah *database*. SQL bekerja pada program *database* seperti *Microsoft Access*, DB2, *Informix*, *Microsoft SQL Server*, *Oracle*, *Sybase*, dan lain-lain. Bahasa SQL memiliki banyak versi yang berbeda-beda, namun untuk memenuhi standar ANSI maka bahasa tersebut harus memiliki kata kunci utama yang berfungsi sama, seperti SELECT, UPDATE, DELETE, INSERT, WHERE, dan lainnya.

2.9.1. Tabel *Database* SQL

Sebuah *database* biasanya memiliki satu tabel atau lebih, dimana setiap tabel memiliki identitas nama dan data. Berikut ini adalah contoh dari tabel yang berisi data karyawan yang diberi nama *Employee*.

Tabel 2. 11 Tabel data karyawan

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

2.9.2. *Query* SQL

Dengan SQL kita dapat membuat *query* dari *database* dan menarik data yang kita inginkan. Contohnya, jika kita hanya ingin melihat data nama belakang karyawan, maka kita membuat sintaks berikut:

SELECT LastName FROM Employee

sehingga tabel yang ditampilkan berisi data mengenai nama belakang karyawan saja, seperti berikut ini:

LastName
Hansen
Svendson
Pettersen

2.9.3. SQL *Data Manipulation Language* (DML)

Script SQL dapat mengeksekusi *query* dan seperti telah disebutkan sebelumnya, bahasa SQL juga memiliki sintaks untuk memperbarui data, memasukkan data, maupun menghapus data. Inilah yang disebut sebagai *Data Manipulation Language* (DML). Berikut ini adalah beberapa perintah *query* yang membentuk DML:

- SELECT – memilih data dari table *database*
- UPDATE – memperbarui data dala table *database*
- DELETE – menghapus data dari tabel *database*
- INSERT INTO – memasukkan data baru ke dalam tabel *database*

2.9.4. SQL *Data Definition Language* (DDL)

DDL adalah bagian dari SQL yang berisi perintah untuk membuat atau menghapus table *database*. Kita juga dapat mendefinisikan indeks, hubungan antar tabel dan menentukan batasan antar tabel dengan menggunakan DDL. Pernyataan DDL antara lain:

- CREATE TABLE – membuat tabel *database* baru
- ALTER TABLE – mengubah tabel *database*
- DROP TABLE – menghapus tabel *database*
- CREATE INDEX – membuat indeks (*search key*)
- DROP INDEX – menghapus indeks

2.10. *Web Hosting Service*

Web hosting service adalah salah satu jenis dari pelayanan *hosting* internet yang memberikan pelayanan atau jasa jaringan internet kepada individu atau pun organisasi via *world wide web. Web host* adalah perusahaan yang memberikan ruang pada server mereka untuk digunakan oleh klien dengan tujuan memberikan koneksi internet, biasanya disebut *data center*. *Web host* juga dapat memberikan ruang pada *data center* dan konektivitas internet untuk server yang sebelumnya tidak dilokasikan pada *data center*, pelayanan ini disebut *colocation*.

Terdapat beberapa jenis jasa *hosting web* yaitu sebagai berikut:

1. *Free web hosting service*: jasa ini gratis, terkadang didukung oleh iklan dari jasa *web hosting* tanpa pembayaran.
2. *Shared web hosting service*: suatu *web site* diletakkan pada server yang sama seperti *site* yang lain, biasanya diurutkan dari beberapa ratus atau ribu *web*. Secara spesifik, semua domain akan berbagi beberapa server, seperti RAM dan CPU pada komputer. *Web site* yang dibagi akan disewakan oleh *reseller*.
3. *Reseller web hosting*: menjadikan klien mereka sebagai *web host*. *Reseller* dapat berfungsi untuk domain individu, di bawah gabungan tipe *hosting*, bergantung kepada siapa mereka mengabungkan diri dengan pemberi jasa.
4. *Virtual dedicated server*: membagi server menjadi *virtual server*, dimana tiap pengguna merasa bahwa mereka yang memiliki server tersebut, tetapi sebenarnya mereka sedang berbagi satu server dengan banyak pengguna lainnya. Pengguna mendapatkan akses utama pada ruang virtual mereka. Hal ini sering dikenal dengan sebutan *virtual private server* atau VPS.
5. *Dedicated hosting service*: pengguna mendapat *web server* mereka dan mendapatkan kontrol penuh terhadap server.
6. *Managed hosting service*: pengguna mendapatkan *web server* mereka tetapi bukan merupakan kontrol penuh, walaupun mereka dapat mengatur data mereka via FTP atau alat *remote* lainnya.
7. *Colocation web hosting service*: sama halnya dengan poin sebelumnya, tetapi pengguna memiliki *colo server*. Perusahaan penyedia layanan memberikan ruang dimana server dapat dibawa dan diatur. Jenis ini merupakan yang paling baik dan cukup mahal dari semua layanan *web hosting*. Pada banyak kasus, penyedia *colocation* akan memberikan sedikit *support* secara langsung pada mesin klien. Mereka lebih fokus pada teknis elektrik, akses internet dan fasilitas penyimpanan pada server. Untuk layanan ini, klien akan mendapatkan dukungan dari administrator *data center* pada *site* untuk memperbarui data atau mengubahnya.
8. *Clustered hosting*: memiliki *multiple server hosting* yang secara daya tampung sama dengan sumber utilisasi yang lebih baik.

9. *Grid Hosting*: bentuk *hosting* ini didistribusikan ketika pembagian server dilakukan seperti jaringan listrik dan dikirim ke beberapa titik.
10. *Home server*: biasanya mesin tunggal ditempatkan di tempat pribadi yang dapat digunakan untuk menyediakan satu web atau lebih dari tingkat koneksi *broadband*.

2.10.1. *Colocation Server*

Colocation server merupakan salah satu jenis layanan di *data center* dimana beberapa pelanggan melokasikan jaringan, server, dan perlengkapan penyimpanan mereka pada penyedia layanan jaringan telekomunikasi atau penyedia layanan internet dengan biaya dan kompleksitas yang minimum. *Colocation* menjadi semakin dikenal karena organisasi melihat manfaat dari segi waktu, biaya, fasilitas internet dan pemeliharaan yang diberikan lebih menguntungkan dibandingkan dengan layanan *hosting* pada umumnya. Penyedia layanan ini akan memberikan layanan kapasitas *bandwith* yang lebih besar, dan kecepatan akses tinggi yang dapat terus dipertimbangkan untuk ditingkatkan. Selain faktor-faktor tersebut, faktor yang tidak kalah penting juga adalah layanan ini memberikan jaminan keamanan dan pemeliharaan jaringan telekomunikasi kepada pihak pelanggan sehingga individu atau organisasi dapat lebih fokus pada bisnis mereka.

Gambar 2. 10 Contoh *Colocation Server* pada *Data Center*

3. PEMBUATAN MODEL DAN PENYEMPURNAAN FITUR WEB

Dalam membuat sebuah simulasi web, yang harus dibuat terlebih dahulu adalah model yang akan dimainkan. Dengan demikian yang menjadi data dari penelitian ini adalah model simulasi permainan *Beer Distribution Game.* Seperti yang telah disebutkan sebelumnya bahwa penelitian ini merupakan penelitian lanjutan dari penelitian yang telah dilakukan oleh Saudari Laura Olivia R. pada tahun 2007, maka sumber data untuk pembuatan model berasal dari model yang telah dibuat sebelumnya yang kemudian akan dilakukan penyempurnaan pada beberapa fitur sampai menghasilkan *output* dalam bentuk *web site* permainan. Berikut ini merupakan penjelasan dari proses pembuatan model yang sudah dilakukan sebelumnya yang terbagi atas dua bagian besar yaitu pembuatan model dan konsep pembuatan *web interface* untuk *local web-based simulation.*

3.1. Pembuatan Model Simulasi

Tahap selanjutnya dari penelitian ini merupakan pembuatan model simulasi. Pembuatan model simulasi dilakukan berdasarkan metode yang telah disebutkan pada dasar teori.

3.1.1. Konseptualisasi

Hal pertama yang harus dilakukan sebelum membuat sebuah model dari suatu sistem adalah membuat konsep dari sistem, tahap ini disebut konseptualisasi sistem. Tahap ini bertujuan untuk membatasi ruang lingkup dari model yang akan dibuat sehinga model benar-benar merepresentasikan permasalahan yang ada. Adapun model yang digunakan dalam penelitian ini adalah model yang sama dengan model penelitian sebelumnya yaitu model *Beer Distribution Game* (*Beer Game*). Model ini dikenal dengan baik sebagai contoh dari struktur rantai pasok (*supply chain*). Model ini tetap digunakan sebagai model penelitian dengan alasan tingkat kompleksitas detail yang tinggi namun *interface* yang minimum, serta kompleksitas dinamis yang medium, sehingga dari segi pembelajaran, model ini

dapat meningkatkan proses pembelajaran dari penulis selama penelitian ini. Sebelum memulai membuat model, hal pertama yang harus dipahami adalah konsep rantai pasok mulai dari hulu hingga hilir kemudian baru memetakannya dalam *causal loop diagram* yang akan dispesifikkan lagi ke *stock and flow diagram*.

3.1.1.1. Identifikasi Problem dan Batasan Model

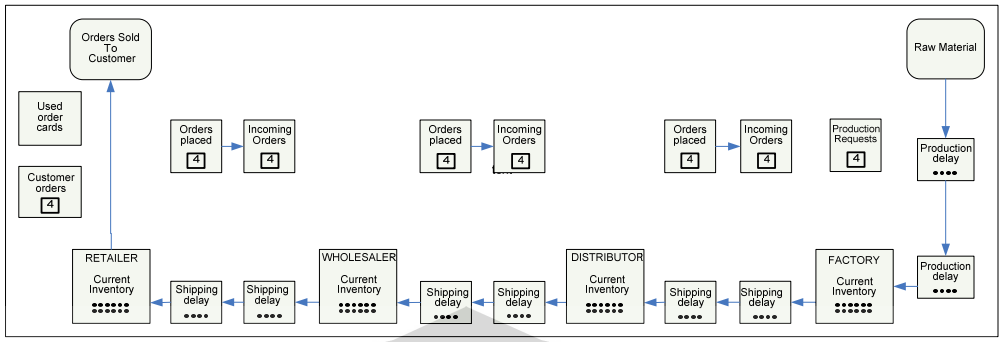
Beer Game merupakan permainan simulasi produksi yang berasal dari Kelompok Sistem Dinamis dari MIT *Sloan School of Management* yang dibuat dengan tujuan untuk mengajarkan sekaligus memberikan pengalaman mengenai pendekatan dinamis dalam menyelesaikan permasalahan manajemen. Permainan ini diciptakan oleh Jay Forrester dan rekan-rekannya sesama profesor pada awal tahun 1960-an.

Dalam permainan ini, setiap pemain dalam satu tim mengatur persediaan tunggal mereka selama permainan berlangsung melalui pengambilan keputusan terhadap tingkat pemesanan pada setiap periode. Setiap peserta terdiri dari 4 sampai 8 pemain dengan posisi sebagai: *Retailer, Wholesaler, Distributor,* dan *Factory*. Setiap sektor akan memesan barang dari sektor di atasnya (*upstream position*) dan menyuplai barang pada sektor di bawahnya (*downstream position*). Satu periode permainan adalah satu minggu. Jika barang tidak mencukupi untuk disuplai maka tidak ada kasus kehilangan penjualan, oleh karena itu kondisi *backlog* berlaku pada permainan ini.

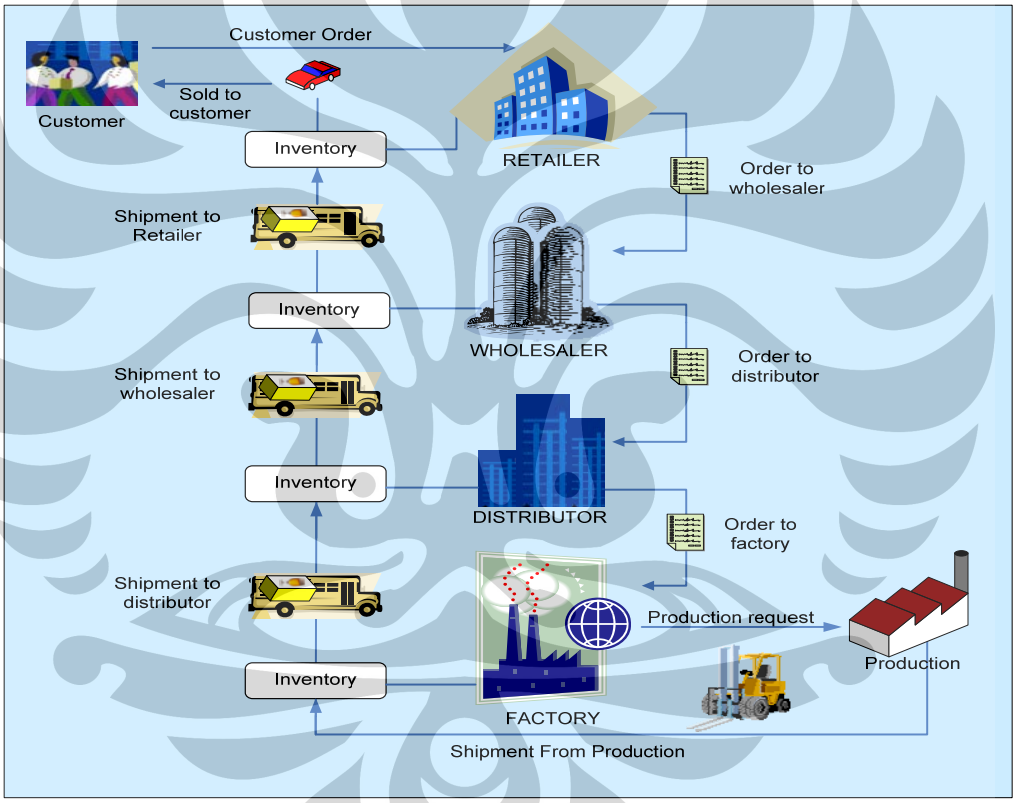
Aliran material dan *order* yang terjadi dapat mulai dijelaskan dari sektor *retailer*. *Retailer* menerima order *customer* dari administrator permainan dan *factory* memberikan perintah produksi pada gudang bahan baku. *Delay* penerimaan produk dan *delay* pengiriman produk antara 2 sektor masing-masing adalah 2 periode, kecuali pada sektor *factory*. Untuk *factory*, *delay* antara pemberian *production request* dan penerimaan produk adalah 3 periode. Pengiriman order yang tidak membutuhkan *delay* adalah pengiriman barang dari *retailer* terhadap order dari *customer*, jadi permintaan yang diterima di periode 1 akan dipenuhi pada periode 1. Pada periode pertama jumlah order dan inventori

setiap mata rantai masing-masing adalah sebanyak 4 unit dan 12 unit, sedangkan produk yang sedang yang berada dalam perjalanan masing-masing sebanyak 4 unit. Tujuan dari permainan ini adalah agar setiap kelompok dapat memenuhi permintaan *customer* dengan total biaya inventori dan *back order* yang minimum. Biaya inventori adalah sebesar $0.5/unit/periode dan biaya *backlog* sebesar $1/unit/periode. *Backlog* dihitung sebanyak order yang tidak dapat dipenuhi dan harus segera dipenuhi pada periode-periode berikutnya.

Selama permainan, setiap pemain tidak boleh berkomunikasi satu sama lain untuk merepresentasikan kondisi sebenarnya dari sistem yang dimainkan. Setiap pemain memperoleh informasi yang berhubungan dengan kondisi barang dan sektor yang dimainkannya (status inventori, *backlog*, dan produk yang sedang dikirim) tetapi memiliki informasi yang terbatas terhadap informasi global (informasi sektor lainnya). Pada akhir permainan, setiap kelompok akan menghitung total biaya yang dikeluarkan dan menggambarkan grafik inventori dan order setiap periode selama permainan berlangsung. Dari grafik itulah, fenomena rantai pasok dapat dilihat dan dianalisa. Fenomena ini merupakan kondisi dimana fluktuasi permintaan maupun inventori akan semakin besar seiring dengan jauhnya sektor dari ujung mata rantai pasok yaitu *customer*. Fluktuasi ini disebut dengan *bullwhip effect*. Salah satu yang menjadi pengaruh terjadinya efek ini terjadi adalah *lead time* dari aliran material dan informasi dari tiap sektor menuju ke *customer*. Alur dan skema permainan ini dapat dilihat pada gambar 3.1 dan 3.2 berikut ini:



Gambar 3. 1 Papan *beer game*[16]



Gambar 3. 2 Skema rantai pasok[17]

[16] Jacobs, F. Robert, *Playing The Beer Distribution Game Over The Internet, Production and Operation Management*, Vol. 9, No. 1, *Spring*, 2000

[17] Olivia, R, Laura, Perancangan Simulasi Sistem Dinamis Berbasis Aplikasi Web Menggunakan Powersim SDK dan ASP, Skripsi, Universitas Indonesia, 2007, hal. 41.

3.1.1.2. Identifikasi Variabel Awal

Konsep dan batasan model yang telah didefinisikan sebelumnya merupakan tahap awal untuk mengetahui variabel-variabel yang terdapat pada model. Identifikasi yang telah dilakukan sebelumnya dibagi menjadi dua tahap. Tahap pertama merupakan identifikasi variabel awal yang bertujuan memetakan konsep model dalam bentuk *Causal Loop Diagram* (CLD), sedangkan tahap kedua merupakan tahap pengembangan dari tahap pertama yang bertujuan untuk memetakan konsep model lebih detail dalam bentuk *Stock and Flow Diagram* (SFD) berdasarkan CLD yang telah terlebih dahulu dibuat. Variabel tahap awal yang membentuk subsistem setiap sektor adalah sebagai berikut:

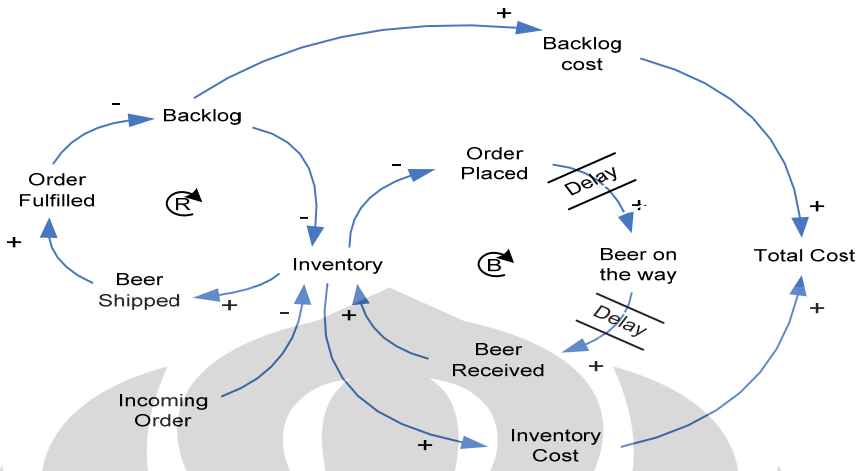
Tabel 3. 1 Identifikasi Variabel Awal

No	Nama Variabel
1	Incoming Order
2	Inventory
3	Order placed
4	Beer received
5	Beer on the way
6	Order fulfilled
7	Backlog
8	Beer shipped
9	Inventory Cost
10	Backlog Cost
11	Total Cost

Sumber: Olivia, R, Laura, 2007, hal. 43

3.1.1.3. Memetakan Konsep Model

Variabel yang telah ditentukan sebelumnya kemudian digunakan untuk memetakan konsep model dalam bentuk CLD. CLD dari variabel-variabel awal pada tiap sektor adalah:



Gambar 3. 3 *Causal loop diagram model beer game*
(Sumber: Olivia, R, Laura, 2007, hal. 44)

Adapun penjelasan dari diagram di atas adalah sebagai berikut. Setiap periode sektor x akan menerima order dari sektor di bawahnya. Order yang diterima akan mengurangi jumlah inventori yang dimiliki sektor x. Untuk menjaga agar inventori tidak kosong maka sektor x akan memberikan order pada sektor di atasnya sehingga setelah *delay* 2 periode sektor x akan menerima barang dan menambah jumlah inventorinya. Order yang diterima akan dikirim sejumlah yang diminta, namun ketika jumlah inventori tidak mencukupi maka jumlah barang yang dikirim tidak dapat memenuhi keseluruhan order sehingga terjadi *backlog*. *Backlog* harus dipenuhi pada periode berikutnya dan dikirim bersama sejumlah barang yang diorder pada periode tersebut.

3.1.1.4. Identifikasi Variabel SFD

Variabel yang telah dipetakan dalam CLD masih merupakan variabel awal yang secara umum terdapat pada tiap sektor. Oleh karena itu untuk merepresentasikan keseluruhan sistem sesuai dengan batasan-batasan yang telah dijelaskan sebelumnya, harus membuat *Stock and Flow Diagram*. SFD ini akan menunjukkan tidak hanya variabel pembentuk CLD saja tetapi juga variabel *flow*

rate yang menunjukkan besar input dan output pada variabel level per satuan waktu. Oleh karena itu variabel tiap sektor harus ditentukan secara keseluruhan.

Pada model penelitian ini yang menjadi variabel level adalah order yang diterima, inventori, *backlog*, barang yang sedang dalam pengiriman dan biaya karena kelima variabel ini akan bertambah maupun berkurang setiap periode dan berperan sebagai stok. Untuk mengetahui total biaya kita memerlukan variabel harga, baik inventori maupun *backlog*. Tahap terakhir, variabel lainnya yang perlu diidentifikasi adalah variabel *flow rate*. Hasil identifikasi variabel yang sudah lengkap dapat dilihat pada tabel 3.2 berikut.

Tabel 3. 2 Daftar Variabel Model

No.	Nama	Jenis	Keterangan
1	Prod_Req	Constant	Permintaan produksi
2	ProdRate_1	Auxilliary	*Flow rate* produksi
3	Prod_Del1	Level	*Production delay*
4	ProdRate_2	Auxilliary	*Flow rate* produksi
5	Prod_Del2	Level	*Production delay*
6	Prod_RecRate	Auxilliary	*Flow rate* barang masuk ke inventori *factory*
7	Fact_Inv	Level	Jumah inventori *factory*
8	D_OrderPlaced	Constant	Keputusan order Distributor
9	D_OrdInRate	Auxilliary	*In flow rate* order distributor
10	F_IncOrder	Level	Order diterima oleh *factory*
11	D_OrdOutRate	Auxilliary	*Out flow rate* order distributor
12	F_ShipRate1	Auxilliary	*Flow rate* barang diantar kepada Distributor
13	F_Backlog	Auxilliary	Jumlah order yang tidak terpenuhi di periode ini
14	F_TotBacklog	Level	Total *backlog* periode sebelumnya
15	F_ComplBacklog	Auxilliary	Jumlah *backlog* yang telah terpenuhi di periode ini
16	F_InvCost	Auxilliary	Biaya inventori
17	F_InvCost per period	Auxilliary	Biaya inventori per periode

Sumber: Olivia, R, Laura, 2007, hal. 43

Tabel 3.2 Daftar Variabel Model (lanjutan)

No.	Nama	Jenis	Keterangan
18	F_Backlog_cost	Auxilliary	Biaya *backlog*
19	F_Backlog_Cost per period	Auxilliary	Biaya *backlog* per periode
20	F_TotCost_rate	Auxilliary	*Flow rate* total biaya *factory*
21	F_TotCost	Level	Total biaya *factory*
22	F_ShipDel1	Level	Barang sedang dalam perjalanan
23	F_ShipRate2	Auxilliary	*Flow rate* pengiriman barang
24	F_ShipDel2	Level	Barang sedang dalam perjalanan
25	D_RecRate	Auxilliary	*Flow rate* pengiriman barang
26	Dist_Inv	Level	Jumlah inventori Distributor
27	W_OrderPlaced	Constant	Keputusan order *Wholesaler*
28	W_OrdInRate	Auxilliary	*In flow rate* order Wholesaler
29	D_IncOrder	Level	Order diterima oleh distributor
30	W_OrdOutRate	Auxilliary	*Out flow rate* order Wholesaler
31	D_ShipRate1	Auxilliary	*Flow rate* barang diantar kepada *Wholesaler*
32	D_Backlog	Auxilliary	Jumlah order yang tidak terpenuhi di periode ini
33	D_TotBacklog	Level	Total *backlog* periode sebelumnya
34	D_ComplBacklog	Auxilliary	Jumlah *backlog* yang telah terpenuhi di periode ini
35	D_InvCost	Auxilliary	Biaya inventori
36	D_InvCost per period	Auxilliary	Biaya inventori per periode
37	D_Backlog_cost	Auxilliary	Biaya *backlog*
38	D_Backlog_Cost per period	Auxilliary	Biaya *backlog* per periode
39	D_TotCost_rate	Auxilliary	*Flow rate* total biaya Distributor
40	D_TotCost	Level	Total biaya Distributor
41	D_ShipDel1	Level	Barang sedang dalam perjalanan
42	D_ShipRate2	Auxilliary	*Flow rate* pengiriman barang
43	D_ShipDel2	Level	Barang sedang dalam perjalanan
44	W_RecRate	Auxilliary	*Flow rate* pengiriman barang
45	Wh_Inv	Level	Jumlah inventori *Wholesaler*
46	R_OrderPlaced	Constant	Keputusan order Retailer
47	R_OrdInRate	Auxilliary	*In flow rate* order Retailer
48	W_IncOrder	Level	Order diterima oleh Wholesaler
49	R_OrdOutRate	Auxilliary	*Out flow rate* order Retailer
50	W_ShipRate1	Auxilliary	*Flow rate* barang diantar kepada *Retailer*

Sumber: Olivia, R, Laura, 2007, hal. 46

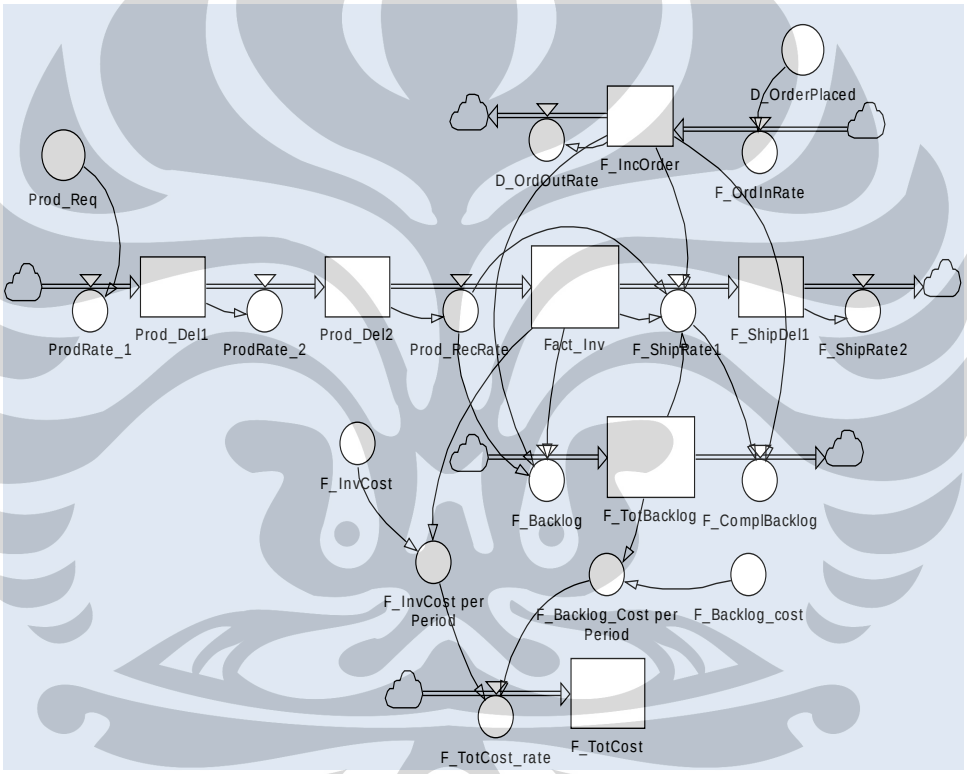
Tabel 3.2 Daftar Variabel Model (lanjutan)

No.	Nama	Jenis	Keterangan
50	W_ShipRate1	Auxilliary	*Flow rate* barang diantar kepada *Retailer*
51	W_Backlog	Auxilliary	Jumlah order yang tidak terpenuhi di periode ini
52	W_TotBacklog	Level	Total *backlog* periode sebelumnya
53	W_ComplBacklog	Auxilliary	Jumlah *backlog* yang telah terpenuhi di periode ini
54	W_InvCost	Auxilliary	Biaya inventori
55	W_InvCost per period	Auxilliary	Biaya inventori per periode
56	W_Backlog_cost	Auxilliary	Biaya *backlog*
57	W_Backlog_Cost per period	Auxilliary	Biaya *backlog* per periode
58	W_TotCost_rate	Auxilliary	*Flow rate* total biaya *Wholesaler*
59	W_TotCost	Level	Total biaya *Wholesaler*
60	W_ShipDel1	Level	Barang sedang dalam perjalanan
61	W_ShipRate2	Auxilliary	*Flow rate* pengiriman barang
62	W_ShipDel2	Level	Barang sedang dalam perjalanan
63	R_RecRate	Auxilliary	*Flow rate* pengiriman barang
64	Ret_Inv	Level	Jumlah inventori *Retailer*
65	Customer Order	Constant	Permintaan *customer*
66	Ret_Sold	Auxilliary	*Flow rate* barang diantar kepada *customer*
67	R_Backlog	Auxilliary	Jumlah order yang tidak terpenuhi di periode ini
68	Ret_TotBacklog	Level	Total *backlog* periode sebelumnya
69	R_TotBacklog	Auxilliary	Jumlah *backlog* yang telah terpenuhi di periode ini
70	R_InvCost	Auxilliary	Biaya inventori
72	R_Backlog_cost	Auxilliary	Biaya *backlog*
73	R_Backlog_Cost per period	Auxilliary	Biaya *backlog* per periode
74	R_TotCost_rate	Auxilliary	*Flow rate* total biaya *Retailer*
75	R_TotCost	Level	Total biaya *Retailer*

Sumber: Olivia, R, Laura, 2007, hal. 47

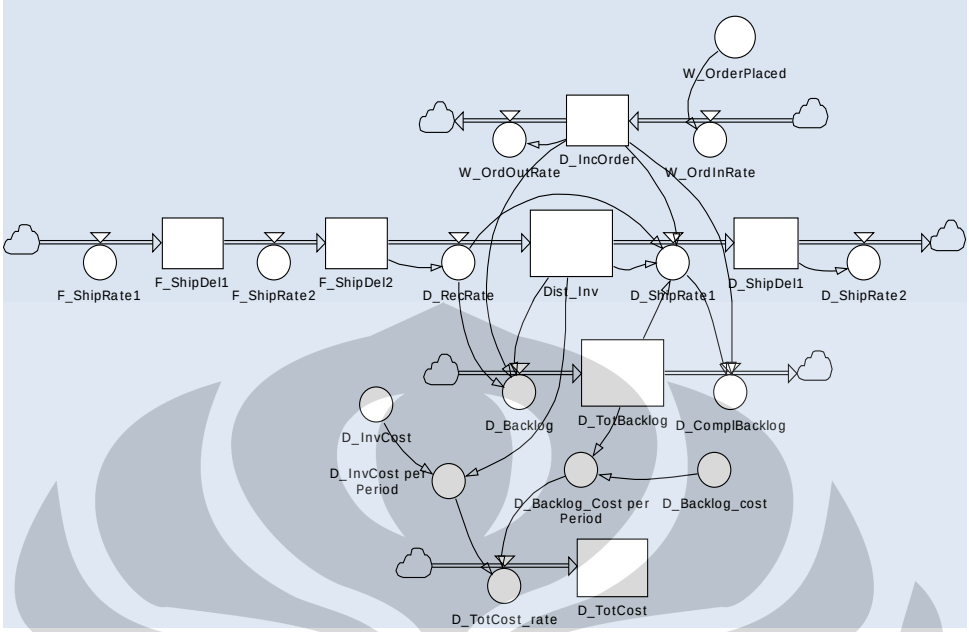
3.1.1.5. Membuat *Stock and Flow Diagram*

Sistem rantai pasok yang direpresentasikan oleh permainan *Beer Game* pada dasarnya terdiri dari 4 subsistem yang kurang lebih sama, yaitu *factory*, *distributor*, *wholesaler*, dan *retailer*. Oleh karena itu untuk memudahkan pembuatan s*tock and flow diagram* keseluruhan dari model permainan ini, Sdri. Laura Olivia membuat diagram untuk model subsistemnya terlebih dahulu baru kemudian menggabungkan keempat diagram submodel tersebut. Berikut ini adalah s*tock and flow diagram* dari masing-masing submodel:



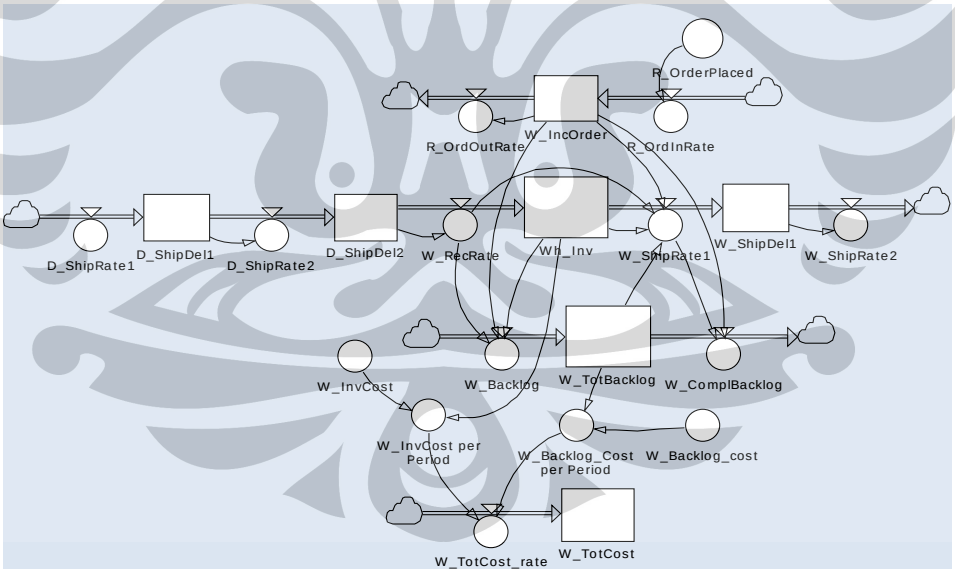
Gambar 3. 4 *Stock and flow diagram* subsistem: *factory*

(Sumber: Olivia, R, Laura, 2007, hal. 47)



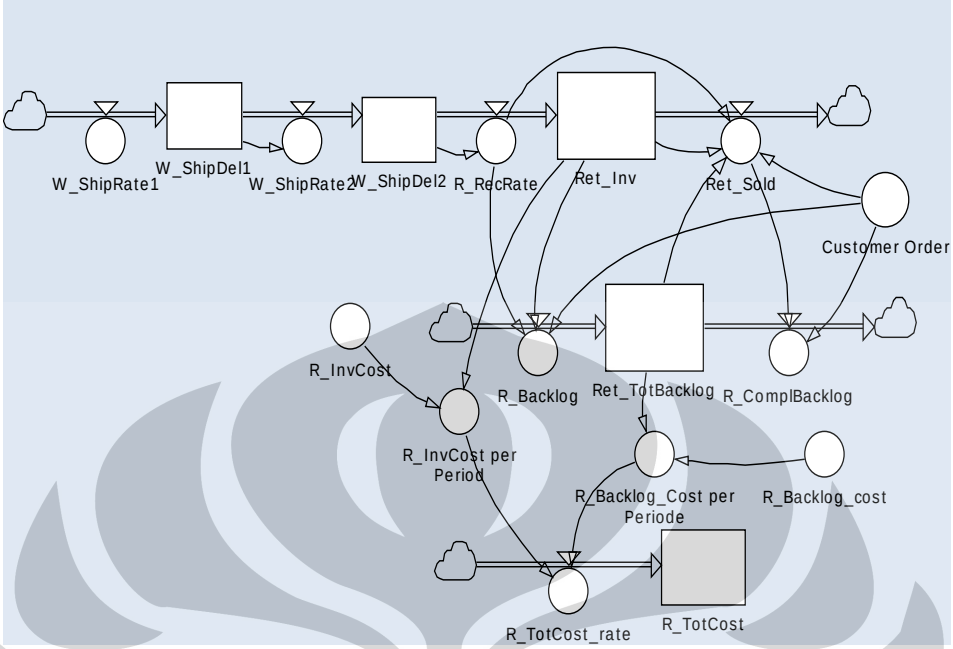
Gambar 3. 5 *Stock and flow diagram* subsistem: *distributor*

(Sumber: Olivia, R, Laura, 2007, hal. 48)



Gambar 3. 6 *Stock and flow diagram* subsistem: *wholesaler*

(Sumber: Olivia, R, Laura, 2007, hal. 48)



Gambar 3. 7 *Stock and flow diagram* subsistem: *retailer*
(Sumber: Olivia, R, Laura, 2007, hal. 47)

3.1.2. Formulasi

Setelah melakukan konseptualisasi terhadap model yang akan dibuat, maka tahap selanjutnya merupakan tahap formulasi setiap variabel-variabel. Variabel-variabel tersebut dispesifikasi dengan menggunakan data numerik. Formulasi dilakukan melalui *software* simulator yaitu Powersim 2005.

Menurut penelitian sebelumnya dan konsep yang sudah ada, *delay* antara pemberian order pada *supplier* dan order diterima oleh *supplier* adalah 2 minggu, lalu *delay* antara barang dikirim oleh *supplier* dan barang diterima oleh sektor dibawahnya juga 2 minggu. Jadi, barang yang dipesan pada periode 1 akan diterima pada periode ke-5 sebagai inventori, kecuali pada *factory* dimana barang yang dipesan pada periode 1 akan diterima pada periode ke-4. Selain itu inventori pada periode t adalah inventori pada periode sebelumnya ditambah kiriman barang dari *supplier* yang berada pada *shipping delay* kedua, seperti persamaan berikut:

$$Inventory_t = (RecRate_t * 1 \text{ <<wk>>}) + Inventory_{(t-1)}$$

Jumlah barang yang dikirim tidak selalu sesuai dengan order yang diterima akan tetapi tergantung jumlah inventori dan *backlog*. Apabila jumlah order lebih besar dari pada inventori maka akan terjadi *backlog* dan barang yang dikirim adalah sebanyak inventori yang dimiliki. Berikut persamaannya:

$$Backlog_t = IF(IncOrder_t > (Inventory_{(t-1)} + (RecRate_t * 1 <<wk>>)), IncOrder_t - (Inventory_{(t-1)} + (RecRate_t * 1 <<wk>>)), 0<<unit>>) /1<<wk>>$$

Karena pada permainan ini terdapat peraturan bahwa tidak ada order yang diabaikan, dimana *backlog* harus dipenuhi sesegera mungkin, maka pada pengiriman barang di periode berikutnya order akan ditambahkan dengan total *backlog*, dengan demikian diperoleh persamaan untuk jumlah barang yang dikirim sebagai berikut:

$$Ship\_rate_t = MIN(Inventory_{(t-1)} + (RecRate_t * 1 <<wk>>), TotBacklog_t + IncOrder_t)/ 1<<wk>>$$

dan *backlog* yang telah dipenuhi akan mengurangi total *backlog* sebanyak:

$$ComplBacklog_t = IF(((Ship\_rate_t*1<<wk>>)-IncOrder_t)>0<<unit>>, ((Ship\_rate_t*1<<wk>>)–IncOrder_t),0<<unit>>)/1<<wk>>$$

Biaya *backlog* per periode adalah total *backlog* dikali biaya *backlog* per unit, yaitu $1.0/unit, sedangkan biaya inventori adalah total inventori dikali biaya inventori per unit, yaitu $0.5/unit.

$$Backlog\_Cost\ per\ period = Backlog\_cost * TotBacklog_t$$

$$InvCost\ per\ period = InvCost * Inventory_t$$

$$TotCost\_rate_t = (InvCost\ per\ Period + Backlog\_Cost\ per\ Period)/1<<wk>>$$

Persamaan-persamaan tersebut kemudian dimasukkan ke dalam setiap variabel model yang telah dibuat. Berikut ini adalah persamaan dari masing-masing variabel serta data yang menunjukkan kondisi awal pada model *Beer Game*:

Tabel 3. 3 Formulasi Variabel Model *Beer Game*

No.	Nama	Formulasi	Satuan
1	Prod_Req	4	unit
2	ProdRate_1	Prod_Req/1<<wk>>	unit/wk
3	Prod_Del1	4	unit
4	ProdRate_2	Prod_Del1/1<<wk>>	unit/wk
5	Prod_Del2	4	unit
6	Prod_RecRate	Prod_Del2/1<<wk>>	unit/wk
7	Fact_Inv	12	unit
8	D_OrderPlaced	4	unit
9	D_OrdInRate	D_OrderPlaced/1<<wk>>	unit/wk
10	F_IncOrder	4	unit
11	D_OrdOutRate	F_IncOrder/1<<wk>>	unit/wk
12	F_ShipRate1	MIN(Fact_Inv+(Prod_RecRate*1<<wk>>), F_TotBacklog+F_IncOrder)/1<<wk>>	unit/wk
13	F_Backlog	IF(F_IncOrder>(Fact_Inv+(Prod_RecRate* 1<<wk>>)),F_IncOrder-(Fact_Inv+ (Prod_RecRate* 1<<wk>>)), 0<<unit>>)/1<<wk>>	unit/wk
14	F_TotBacklog	0	unit
15	F_ComplBacklog	IF(((F_ShipRate1*1<<wk>>)-F_IncOrder)> 0<<unit>>, ((F_ShipRate1*1<<wk>>)- F_IncOrder),0<<unit>>)/ 1<<wk>>	unit/wk
16	F_InvCost	0.5	$/unit
17	F_InvCost per period	F_InvCost*Fact_Inv	$
18	F_Backlog_cost	1	$/unit
19	F_Backlog_Cost per period	F_Backlog_cost*F_TotBacklog	$
20	F_TotCost_rate	('F_InvCost per Period'+'F_Backlog_Cost per Period')/1<<wk>>	$/wk
21	F_TotCost	0	$
22	F_ShipDel1	4	unit
23	F_ShipRate2	F_ShipDel1/1<<wk>>	unit/wk
24	F_ShipDel2	4	unit
25	D_RecRate	F_ShipDel2/1<<wk>>	unit/wk
26	Dist_Inv	12	unit
27	W_OrderPlaced	4	unit
28	W_OrdInRate	W_OrderPlaced/1<<wk>>	unit/wk
29	D_IncOrder	4	unit
30	W_OrdOutRate	D_IncOrder/1<<wk>>	unit/wk
31	D_ShipRate1	MIN(Dist_Inv+(D_RecRate*1<<wk>>), D_TotBacklog+ D_IncOrder)/1<<wk>>	unit/wk
32	D_Backlog	IF(D_IncOrder>(Dist_Inv+(D_RecRate* 1<<wk>>)),D_IncOrder-(Dist_Inv+(D_RecRate* 1<<wk>>)),0<<unit>>)/1<<wk>>	unit/wk
33	D_TotBacklog	0	unit
34	D_ComplBacklog	IF(((D_ShipRate1*1<<wk>>)-D_IncOrder)> 0<<unit>>,((D_ShipRate1*1<<wk>>)- D_IncOrder),0<<unit>>)/1<<wk>>	unit/wk
35	D_InvCost	0.5	$/unit

Sumber: Olivia, R., Laura, 2007, hal.50-51

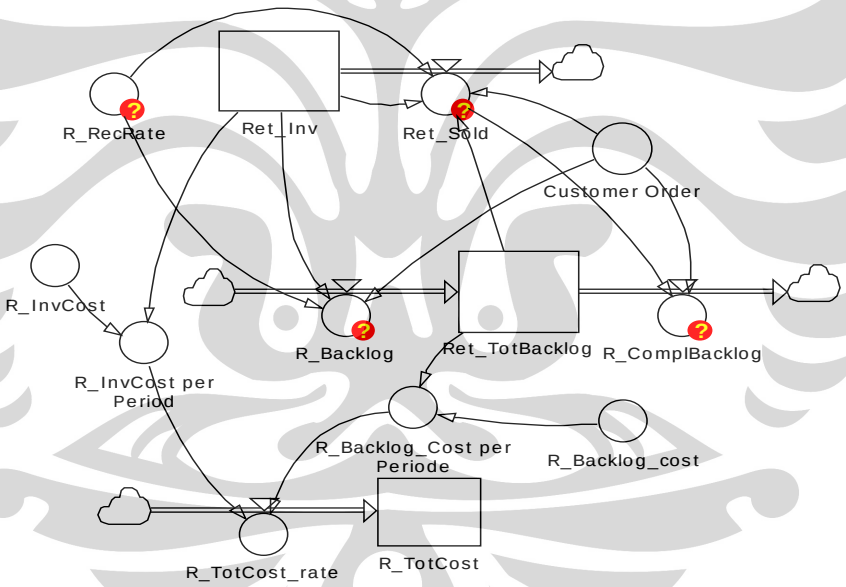
Tabel 3.3 Formulasi Variabel Model *Beer Game* (Lanjutan)

No.	Nama	Formulasi	Satuan
36	D_InvCost per period	Dist_Inv*D_InvCost	$
37	D_Backlog_cost	1	$/unit
38	D_Backlog_Cost per period	D_Backlog_cost*D_TotBacklog	$
39	D_TotCost_rate	('D_Backlog_Cost per Period'+'D_InvCost per Period')/1<<wk>>	$/wk
40	D_TotCost	0	$
41	D_ShipDel1	4	unit
42	D_ShipRate2	D_ShipDel1/1<<wk>>	unit/wk
43	D_ShipDel2	4	unit
44	W_RecRate	D_ShipDel2/1<<wk>>	unit/wk
45	Wh_Inv	12	unit
46	R_OrderPlaced	4	unit
47	R_OrdInRate	R_OrderPlaced/1<<wk>>	unit/wk
48	W_IncOrder	4	unit
49	R_OrdOutRate	W_IncOrder/1<<wk>>	unit/wk
50	W_ShipRate1	MIN(Wh_Inv+(W_RecRate*1<<wk>>), W_TotBacklog+W_IncOrder)/1<<wk>>	unit/wk
51	W_Backlog	IF(W_IncOrder>(Wh_Inv+(W_RecRate* 1<<wk>>)),W_IncOrder-(Wh_Inv+ (W_RecRate* 1<<wk>>)), 0<<unit>>)/1<<wk>>	unit/wk
52	W_TotBacklog	0	unit
53	W_ComplBacklog	IF(((W_ShipRate1*1<<wk>>)-W_IncOrder)> 0<<unit>>,((W_ShipRate1*1<<wk>>)- W_IncOrder),0<<unit>>)/1<<wk>>	unit/wk
54	W_InvCost	0.5	$/unit
55	W_InvCost per period	Wh_Inv*W_InvCost	$
56	W_Backlog_cost	1	$/unit
57	W_Backlog_Cost per period	W_Backlog_cost*W_TotBacklog	$
58	W_TotCost_rate	('W_InvCost per Period'+'W_Backlog_Cost per Period')/1<<wk>>	$/wk
59	W_TotCost	0	$
60	W_ShipDel1	4	unit
61	W_ShipRate2	W_ShipDel1/1<<wk>>	unit/wk
62	W_ShipDel2	4	unit
63	R_RecRate	W_ShipDel2/1<<wk>>	unit/wk
64	Ret_Inv	12	unit
65	Customer Order	4	unit
66	Ret_Sold	MIN(Ret_Inv+(R_RecRate*1<<wk>>), Ret_TotBacklog+ 'Customer Order')/1<<wk>>	unit/wk
67	R_Backlog	IF('Customer Order'>(Ret_Inv+(R_RecRate* 1<<wk>>)), 'Customer Order'-(Ret_Inv+ (R_RecRate*1<<wk>>)), 0<<unit>>)/1<<wk>>	unit/wk
68	Ret_TotBacklog	0	unit
69	R_TotBacklog	IF(((Ret_Sold*1<<wk>>)-'Customer Order')>0<<unit>>,((Ret_Sold*1<<wk>>)- 'Customer Order'),0<<unit>>)/1<<wk>>	unit/wk
70	R_InvCost	0.5	$/unit
71	R_InvCost per period	Ret_Inv*R_InvCost	$
72	R_Backlog_cost	1	$/unit
73	R_Backlog_Cost per period	R_Backlog_cost*Ret_TotBacklog	$
74	R_TotCost_rate	('R_Backlog_Cost per Periode'+'R_InvCost per Period')/1<<wk>>	$/wk
75	R_TotCost	0	$

Sumber: Olivia, R., Laura, 2007, hal. 52

3.1.3. Verifikasi Model

Berdasarkan teori yang telah disebutkan sebelumnya, tahap verifikasi merupakan tahap yang bertujuan untuk menguji apakah model yang telah dibuat telah sesuai dengan konsep awal sehingga simulasi dapat dijalankan tanpa kesalahan.Verifikasi ini telah dilakukan pada penelitian sebelumnya dengan menggunakan dua metode yaitu dengan melihat indikator kesalahan pada formulasi model yang menggunakan *software* Powersim Studio dan metode pembandingan *output* simulasi dengan data *output* pada permainan manualnya. Metode pertama dilakukan mengalami kemudahan karena indikator kesalahan sudah dapat langsung ditunjukkan dengan tanda tanya merah oleh *software* Powersim Studio tanpa harus menjalankan model.



Gambar 3. 8 Indikator *error* pada model
(Sumber: Olivia, R, Laura, 2007, hal. 53)

Setelah lulus tahap verifikasi melalui *software*, formulasi model juga diverifikasi dengan membandingkan data model dengan konsepnya. Pembandingan ini dilakukan melalui proses perhitungan analisis sesuai dengan konsep awal dengan menggunakan *software Microsoft Excel* dengan data hasil

simulasi, jika keduanya menghasilkan nilai yang sama maka model ini telah lulus verifikasi. Berikut ini merupakan data hasil perhitungan analisis dengan *MS. Excel*:

Tabel 3. 4 Data Permainan Pada Sektor *Factory*

Periode	Production Request	Shipping Delay 1	Shipping Delay 2	D_Order Placed	F_Incoming Order	Factory Inventory	Factory Backlog	F_Total Backlog	Shipping Delay 1
0	4	4	4	4	4	12	0	0	4
1	7	4	4	5	4	12	0	0	4
2	4	7	4	3	5	12	0	0	4
3	11	4	7	12	3	11	0	0	5
4	11	11	4	12	12	15	0	0	3
5	11	11	11	12	12	7	0	0	12
6	11	11	11	12	12	6	0	0	12

Sumber: Olivia, R., Laura, 2007, hal. 54

Tabel 3. 5 Data Permainan Pada Sektor *Distributor*

Periode	Shipping Delay 2	W_Order Placed	D_Incoming Order	Distributor Inventory	Distributor Backlog	D_Total Backlog	Shipping Delay 1
0	4	4	4	12	0	0	4
1	4	9	4	12	0	0	4
2	4	4	9	12	0	0	4
3	4	13	4	7	0	0	9
4	5	13	13	7	1	0	4
5	3	13	13	0	10	1	12
6	12	13	13	0	1	11	3

Sumber: Olivia, R., Laura, 2007, hal. 54

Tabel 3. 6 Data Permainan Pada Sektor *Wholesaler*

Periode	Shipping Delay 2	R_Order Placed	W_Incoming Order	Wholesaler Inventory	Wholesaler Backlog	W_Total Backlog	Shipping Delay 1
0	4	4	4	12	0	0	4
1	4	3	4	12	0	0	4
2	4	7	3	12	0	0	4
3	4	14	7	13	0	0	3
4	9	14	14	10	0	0	7
5	4	14	14	5	5	0	14
6	12	14	14	0	2	5	9

Sumber: Olivia, R., Laura, 2007, hal. 54

Tabel 3. 7 Data Permainan Pada Sektor *Retailer*

Periode	Shipping Delay 2	Cust_Order Placed	Retailer Inventory	Retailer Backlog	R_Total Backlog	Ret_Sold
0	4	4	12	0	0	4
1	4	5	12	0	0	5
2	4	9	11	0	0	9
3	4	15	6	5	0	10
4	3	15	0	12	5	3
5	7	15	0	8	17	7
6	14	15	0	1	25	14

Sumber: Olivia, R., Laura, 2007, hal. 54

Data yang dibandingkan dengan hasil perhitungan selama 6 periode di atas adalah data *output* simulasi yang dimainkan selama 6 periode. Data ini merupakan data historis dari penelitian sebelumnya.

Tabel 3. 8 Data Hasil Simulasi *Factory*

Time	Prod_Req (unit)	Prod_Del1 (unit)	Prod_Del2 (unit)	D_OrderPlaced (unit)	F_IncOrder (unit)	Fact_Inv (unit)	F_Backlog (unit/wk)	F_TotBacklog (unit)	F_ShipDel1 (unit)
Dec 30, 2006	4.00	4.00	4.00	4.00	4.00	12.00	0.00	0.00	4.00
Jan 07, 2007	7.00	4.00	4.00	5.00	4.00	12.00	0.00	0.00	4.00
Jan 14, 2007	4.00	7.00	4.00	3.00	5.00	12.00	0.00	0.00	4.00
Jan 21, 2007	11.00	4.00	7.00	12.00	3.00	11.00	0.00	0.00	5.00
Jan 28, 2007	11.00	11.00	4.00	12.00	12.00	15.00	0.00	0.00	3.00
Feb 05, 2007	11.00	11.00	11.00	12.00	12.00	7.00	0.00	0.00	12.00
Feb 12, 2007	11.00	11.00	11.00	12.00	12.00	6.00	0.00	0.00	12.00
Feb 19, 2007									

Sumber: Olivia, R., Laura, 2007, hal. 55

Tabel 3. 9 Data Hasil Simulasi *Distributor*

F_ShipDel2 (unit)	W_OrderPlaced (unit)	D_IncOrder (unit)	Dist_Inv (unit)	D_Backlog (unit/wk)	D_TotBacklog (unit)	D_ShipDel1 (unit)
4.00	4.00	4.00	12.00	0.00	0.00	4.00
4.00	9.00	4.00	12.00	0.00	0.00	4.00
4.00	4.00	9.00	12.00	0.00	0.00	4.00
4.00	13.00	4.00	7.00	0.00	0.00	9.00
5.00	13.00	13.00	7.00	1.00	0.00	4.00
3.00	13.00	13.00	0.00	10.00	1.00	12.00
12.00	13.00	13.00	0.00	1.00	11.00	3.00

Sumber: Olivia, R., Laura, 2007, hal. 55

Tabel 3. 10 Data Hasil Simulasi *Wholesaler*

D_ShipDel2 (unit)	R_OrderPlaced (unit)	W_IncOrder (unit)	Wh_Inv (unit)	W_Backlog (unit/wk)	W_TotBacklog (unit)	W_ShipDel1 (unit)
4.00	4.00	4.00	12.00	0.00	0.00	4.00
4.00	3.00	4.00	12.00	0.00	0.00	4.00
4.00	7.00	3.00	12.00	0.00	0.00	4.00
4.00	14.00	7.00	13.00	0.00	0.00	3.00
9.00	14.00	14.00	10.00	0.00	0.00	7.00
4.00	14.00	14.00	5.00	5.00	0.00	14.00
12.00	14.00	14.00	0.00	2.00	5.00	9.00

Sumber: Olivia, R., Laura, 2007, hal. 55

Tabel 3. 11 Data Hasil Simulasi *Retailer*

W_ShipDel2 (unit)	Customer Order (unit)	Ret_Inv (unit)	R_Backlog (unit/wk)	Ret_TotBacklog (unit)	Ret_Sold (unit/wk)
4.00	4.00	12.00	0.00	0.00	4.00
4.00	5.00	12.00	0.00	0.00	5.00
4.00	9.00	11.00	0.00	0.00	9.00
4.00	15.00	6.00	5.00	0.00	10.00
3.00	15.00	0.00	12.00	5.00	3.00
7.00	15.00	0.00	8.00	17.00	7.00
14.00	15.00	0.00	1.00	25.00	14.00

Sumber: Olivia, R., Laura, 2007, hal. 55

Data yang sama ternyata diperoleh dari kedua metode. Agar verifikasi semakin jelas, maka tidak hanya dilakukan perbandingan terhadap nilai *output* tetapi juga perbandingan perilaku model dan permainan manualnya. Dalam pengujian tersebut, model dijalankan dalam 12 periode untuk melihat grafik

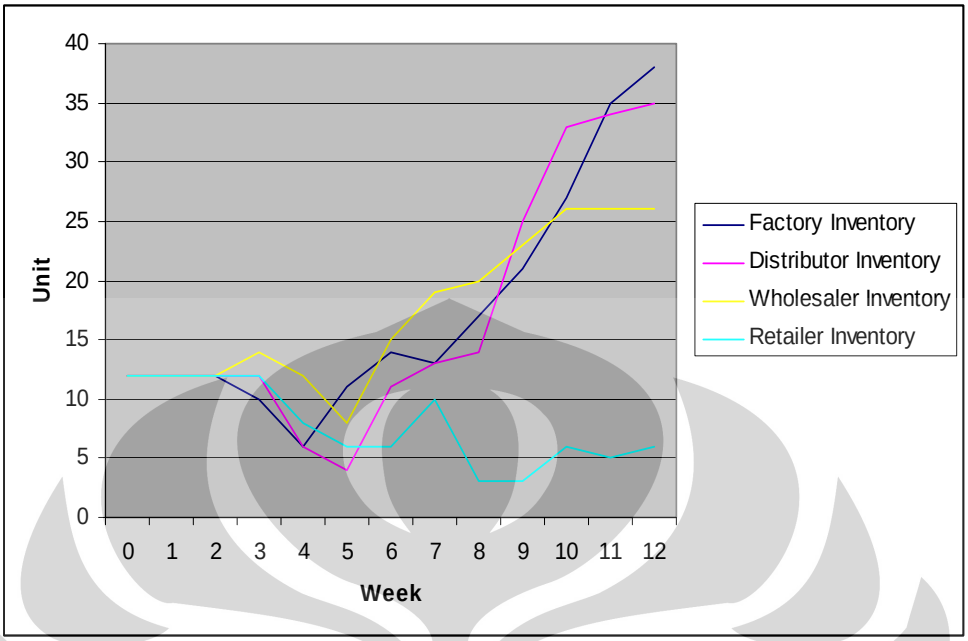
inventori dan order setiap sektor. Berikut ini merupakan perbandingan data yang diperoleh:

Tabel 3. 12 Keputusan Order

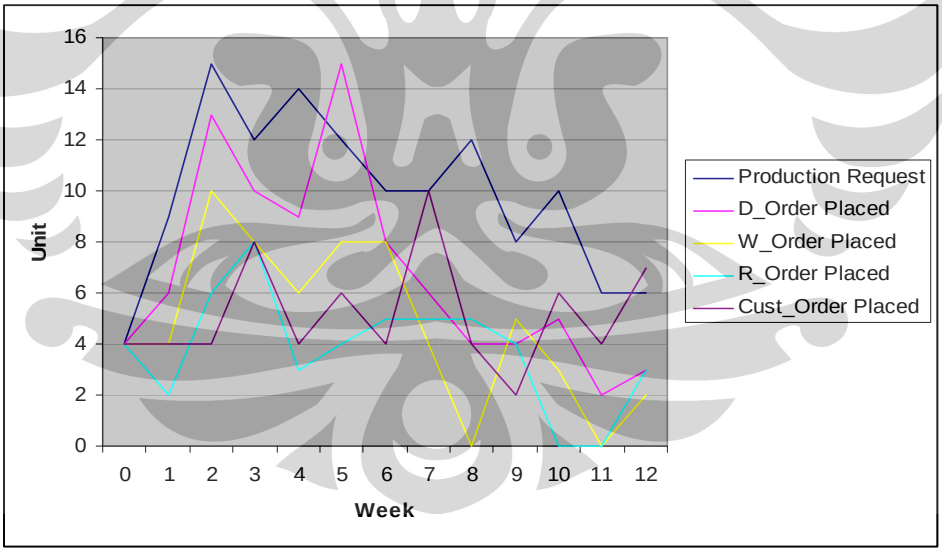
Periode	Factory	Distributor	Wholesaler	Retailer	Customer
0	4	4	4	4	4
1	9	6	4	2	4
2	15	13	10	6	4
3	12	10	8	8	8
4	14	9	6	3	4
5	12	15	8	4	6
6	10	8	8	5	4
7	10	6	4	5	10
8	12	4	0	5	4
9	8	4	5	4	2
10	10	5	3	0	6
11	6	2	0	0	4
12	6	3	2	3	7

Sumber: Olivia, R., Laura, 2007, hal. 56

Dari perhitungan analisis terhadap input tersebut diperoleh grafik inventori dan order yang dapat dilihat pada gambar berikut:



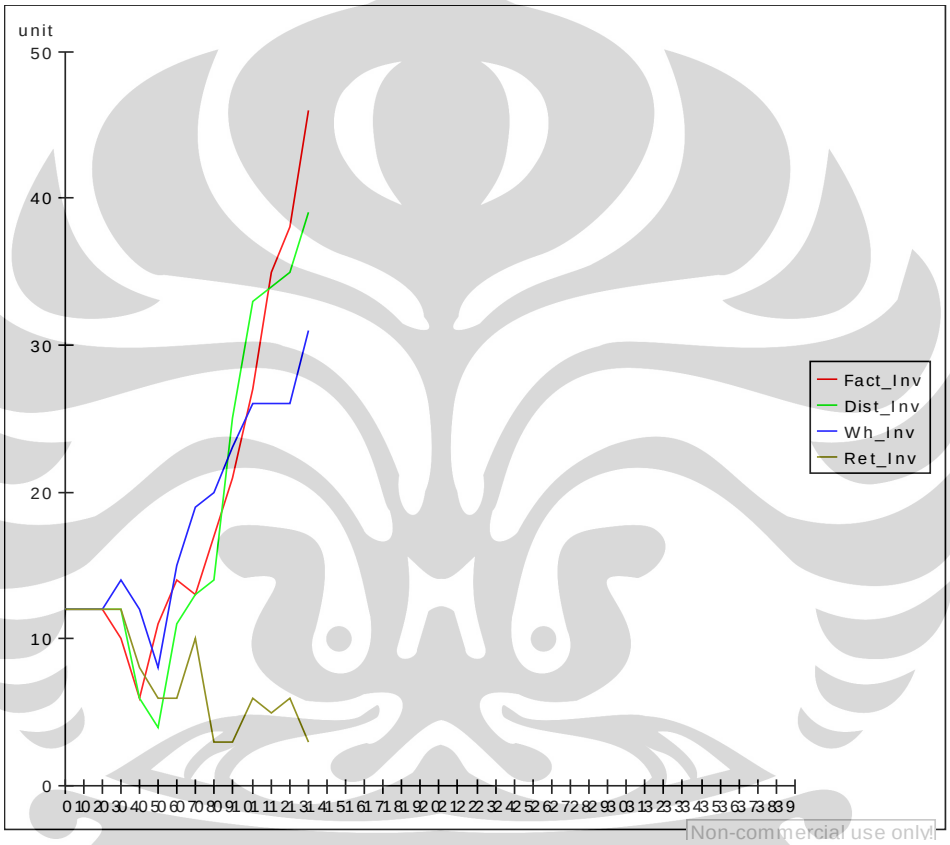
Gambar 3. 9 Grafik inventori setiap sektor dari perhitungan analisis (Sumber: Olivia, R., Laura, 2007, hal. 56)



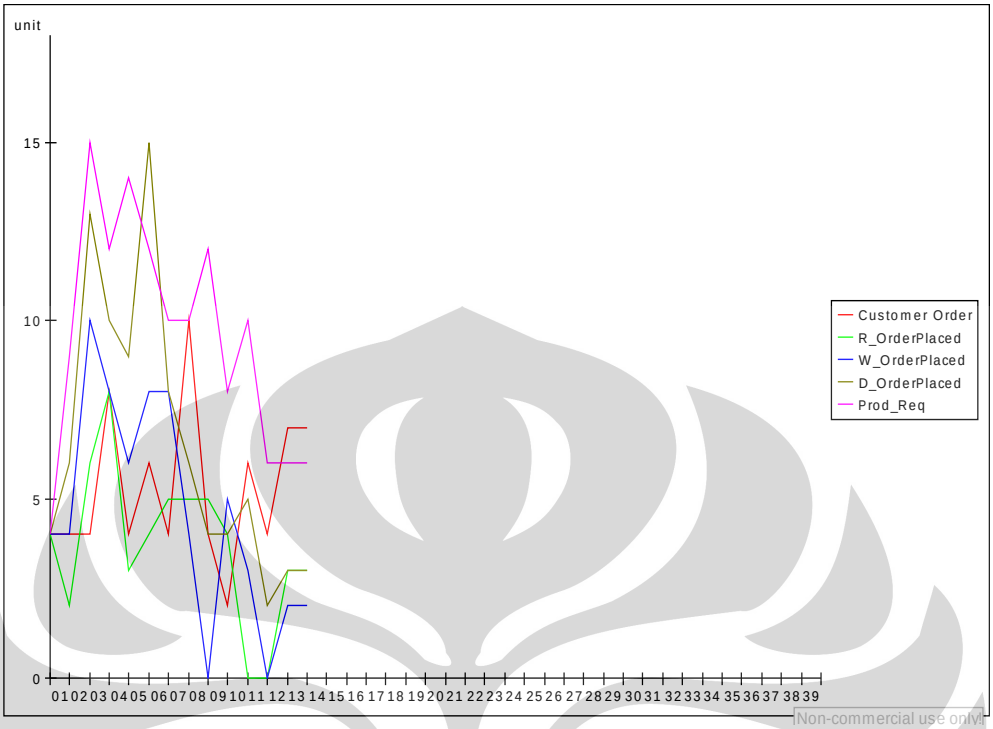
Gambar 3. 10 Grafik order setiap sektor dari perhitungan analisis (Sumber: Olivia, R., Laura, 2007, hal. 56)

Menurut konsep awal *Beer Game*, grafik-grafik tersebut menunjukkan fenomena yang terjadi pada model rantai pasok ini, yakni *Bullwhip Effect*. Efek

ini akan menunjukkan adanya fluktuasi *behavior* yang semakin besar seiring jauhnya sektor dari ujung rantai, yaitu *customer*. Berdasarkan data acuan ini kita menguji model yang telah dibuat untuk melihat apakah model juga memberikan *behavior* yang sama dengan sistem nyatanya. Setelah model disimulasikan selama 12 periode dengan input yang ditetapkan, maka diperoleh grafik sebagai berikut:



Gambar 3. 11 Grafik inventori setiap sektor dari simulasi model (Sumber: Olivia, R., Laura, 2007, hal. 58)



Gambar 3. 12 Grafik order setiap sektor dari simulasi model
(Sumber: Olivia, R., Laura, 2007, hal. 59)

Grafik tersebut menunjukkan *behavior* inventori dan permintaan dari setiap sektor pada model yang dibuat. Terlihat bahwa semakin jauh sektor dari mata rantai (*customer*), semakin besar pula nilai inventori dan permintaan serta fluktuasi yang terjadi. Ini menunjukkan terjadinya *Bullwhip Effect* pada model yang dijalankan. Dari hasil perbandingan data *output* dan grafik perilaku antara model dan perhitungan analitis dari permainan *Beer Game* diperoleh hasil yang sama, sehingga model sehingga model dapat dinyatakan lulus verifikasi karena model telah dapat merepresentasikan keadaan yang sesuai konsep awal. Karena konsep model yang digunakan adalah konsep dari sebuah permainan simulasi, maka pada penelitian ini tidak dilakukan uji validasi model yang membandingkan model dengan keadaan sistem di dunia nyata.

3.2. Pembuatan *Web Interface* Model

Seperti yang telah disebutkan sebelumnya bahwa inti penelitian yang telah dilakukan sebelumnya oleh Sdri Laura Olivia, S.T. merupakan pembuatan model simulasi, perancangan *web-interface* untuk multi pengguna dan pembuatan grafik simulasi. Setelah memaparkan proses pembuatan model di bagian sebelumnya, penulis akan memaparkan proses pembuatan *web interface* dan grafik yang dihasilkan oleh simulasi web yang sudah dapat dimainkan. Tahap ini merupakan tahap pembuatan *web interface* sebagai penghubung antara model simulasi dengan jaringan web. Tahap pembuatan *web interface* yang akan dipaparkan disini merupakan tahap pembuatan untuk simulasi multi pengguna dan tidak memaparkan simulasi *single user* dengan pertimbangan bahwa *output* penelitian sebelumnya yaitu simulasi web untuk *localhost* telah dapat dimainkan.

Proses pembuatan *web interface* yang telah dilakukan menggunakan bahasa pemrograman ASP (*Active Server Pages*) dan Powersim SDK yang berfungsi menghubungkan model agar dapat menerjemahkan *input* dari *browser* dalam bahasa ASP sehingga dapat diterima dan diolah oleh Powersim Studio 2005. Pengolahan data dengan *simulator engine* tersebut dilakukan untuk menjalankan model. Semua data simulasi web akan disimpan dalam *database* di server dengan menggunakan *Ms. Access* sebagai *database engine.*

3.1.4. Konsep Perancangan *Web Based Simulation*

Output yang telah dihasilkan oleh penelitian sebelumnya merupakan suatu program kecil (*applet*) berupa permainan simulasi yang perlu dibuat konsep perancangannya atau konstruksi program tersebut terlebih dahulu. Salah satu cara dalam mendokumentasikan *blueprint* dari suatu *applet* adalah dengan membuat diagram kelas UML (*class diagram*).

Diagram kelas berfungsi untuk memberikan gambaran dari struktur dan deskripsi kelas, *package*, dan objek serta hubungannya satu sama lain. Tiga komponen utama dari kelas adalah nama, atribut dan metode/operasi. Pada model *Beer Game* pada penelitian ini, diagram kelas digunakan untuk menggambarkan kelas-kelas apa saja yang membangun program ini serta atribut dan metode apa

saja yang dilakukan oleh masing-masing kelas agar model dapat dijalankan melalui web.

Seperti telah disebutkan pada bab sebelumnya, simulasi web yang dibuat dalam penelitian ini adalah simulasi yang dapat dimainkan oleh multi pengguna (*multi-user*) yang artinya simulasi dimainkan oleh lebih dari satu orang pemain yang bermain secara bersamaan. Agar jalannya simulasi dapat terkoordinasi dan terintegrasi maka perlu adanya operator atau administrator permainan yang bertugas mengatur dan menjalankan permainan simulasi. Pembagian peran antara pemain dan administrator akan dijelaskan dalam uraian berikut.

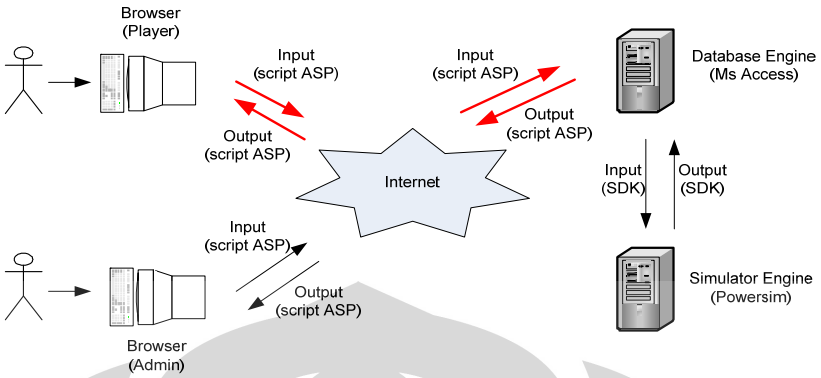
Berdasarkan konsep, peran yang dapat dimainkan dalam simulasi model *Beer Game* ada 5 yaitu *factory*, *distributor*, *wholesaler*, *retailer* dan *customer*. Peran *customer* hanya memberikan order kepada *retailer*, sedangkan keempat sektor lainnya harus mengirimkan barang sesuai permintaan yang diterima kepada sektor di bawahnya dan memesan barang sesuai kebutuhan kepada sektor di atasnya dengan mempertimbangkan inventori, *backlog* dan jumlah barang yang sedang berada dalam perjalanan (*shipping delay*). Dari konsep ini maka penulis membagi peran-peran tersebut untuk 2 sisi pengguna yaitu sisi pemain (*player*) dan sisi administrator. Karena tugas *customer* tidak banyak, maka posisi ini akan dimainkan oleh administrator dan 4 peran lainnya akan dimainkan oleh 4 orang pemain.

Kedua sisi pengguna ini mengakses model dari jalur yang berbeda dan memiliki kontrol akses berupa kata sandi atau *password*. Prinsip kerjanya dapat dijelaskan dalam uraian berikut ini. Seperti halnya permainan simulasi bisnis *on-line* L'Oreal, pemain dapat memainkan permainan *Beer Game* ini dari komputer di mana saja selama masih terhubung dengan internet dan untuk itu waktu permainan harus ditentukan oleh administrator agar semua pemain dapat bermain secara bersamaan pada waktu yang sama. Apabila hanya terdapat satu orang pemain maka simulasi tidak dapat dijalankan, untuk itu pemain yang akan mengikuti permainan ini akan diundang oleh administrator permainan. Ketika pemain hendak bergabung, pemain harus mengkonfirmasikan kembali keikutsertaannya pada administrator permainan untuk mendapatkan *account* dan

password serta peran dalam permainan. Dengan *account* tersebut maka pemain dapat memasuki permainan simulasi *Beer Game* pada halaman web pemain (*player-side*).

Pada halaman web tersebut pemain dapat memasukkan nilai *input* order sesuai sektornya dan mengubah data personalnya. Akan tetapi pemain tidak memiliki wewenang untuk menjalankan simulasi. Semua aktivitas yang dilakukan oleh pemain adalah menarik dan menyimpan data pada *database* yang terdapat di server. Jadi dengan kata lain, pemain hanya berhubungan dengan *database engine*.

Berbeda dengan pemain, administrator akan memasuki permainan simulasi *Beer Game* pada halaman yang berbeda namun tetap disertai dengan *account* khusus untuk mencegah model diubah-ubah dan diduplikasi oleh pengguna lainnya sehingga dapat mengacaukan model. Administrator tidak hanya berhubungan dengan *database engine* melainkan juga berhubungan dengan *simulator engine*. Setelah masuk ke halaman web admin (*admin-side*), administrator akan menarik *input* pemain dari *database* dan memasukkan *input* tersebut ke dalam model untuk dijalankan oleh Powersim, *output* simulasi kemudian kembali disimpan ke dalam *database* sehingga dapat ditarik oleh pemain, demikian seterusnya hingga periode simulasi dinyatakan berakhir oleh administrator. Wewenang administrator sangat luas baik terhadap jalannya permainan simulasi maupun terhadap pengaturan permainan dan pemain. Dari penjelasan ini dapat digambarkan bahwa skema dari simulasi web yang akan dibuat secara umum adalah sebagai berikut:



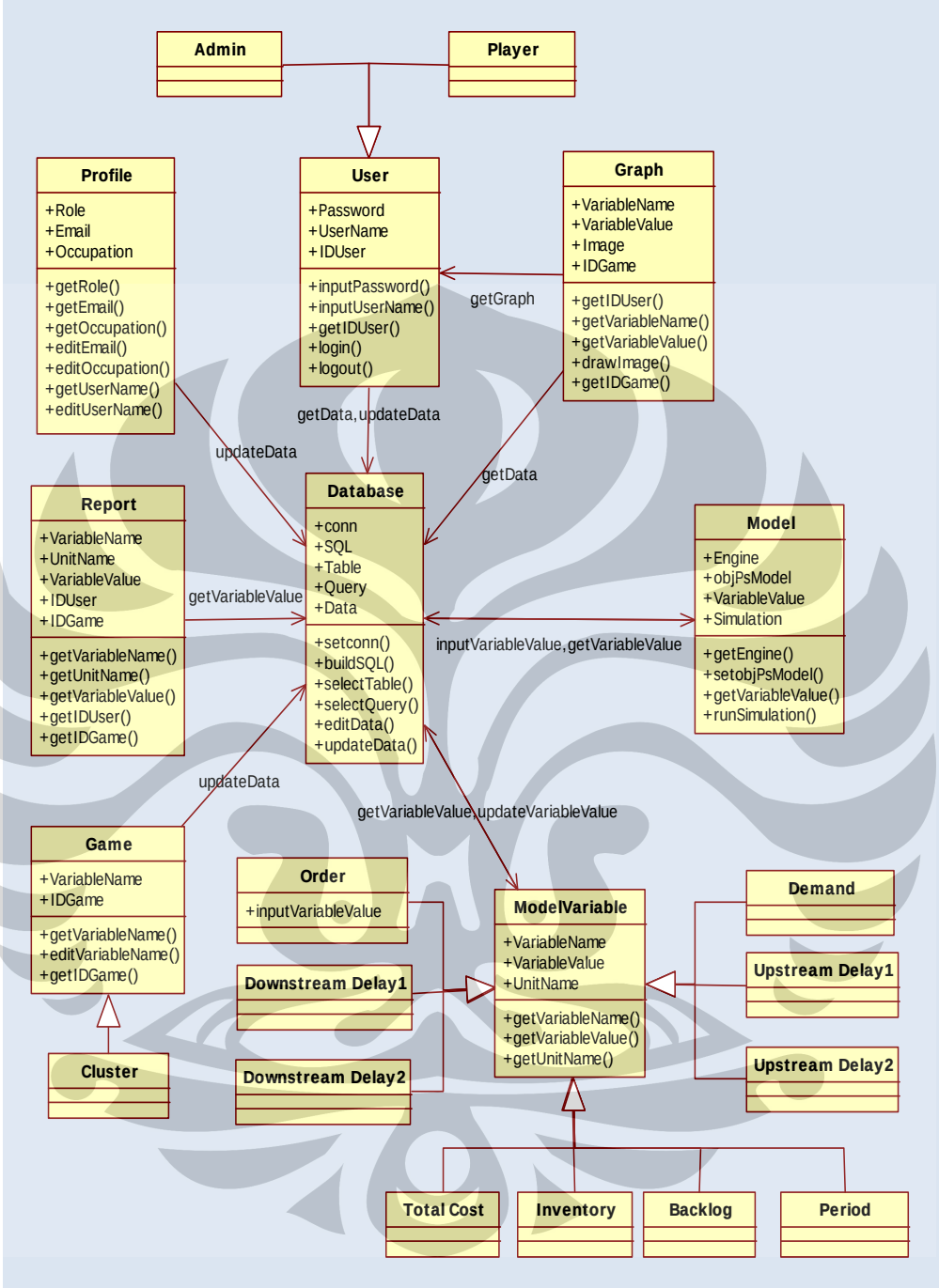
Gambar 3. 13 Skema umum simulasi web

(Sumber: Olivia, R., Laura, 2007, hal. 63)

Untuk kelancaran jalannya prosedur permainan simulasi, harus ada koordinasi yang jelas antara pemain dan administrator permainan. Di setiap awal periode, administrator akan memasukkan *input customer order* untuk sektor *retailer*. Ketika periode dimulai, pemain memasukkan *input* order untuk periode saat itu dengan pertimbangan yang dilakukan berdasarkan informasi yang diperoleh mengenai sektornya. Administrator akan memberikan tolerasi waktu tertentu (misalnya 5 menit) bagi pemain untuk memasukkan dan mengubah nilai *input*. Setelah waktu habis, maka administrator akan mengunci permainan selama 3 menit sehingga pemain tidak dapat lagi mengubah nilai order yang telah dimasukkannya. Selama permainan dikunci, administrator akan menjalankan simulasi untuk periode saat itu berdasarkan *input* dari pemain. *Output* simulasi kemudian secara otomatis akan tersimpan pada tabel laporan (*report*) pada *database*. Setelah menjalankan simulasi selama 1 periode, administrator kembali memasukkan *input* order untuk periode yang baru dan membuka permainan sehingga data laporan periode sebelumnya dan order baru bagi *retailer* dapat diterima oleh pemain dan pemain dapat memasukkan *input* untuk periode baru tersebut. Demikian seterusnya hingga permainan mencapai periode akhir yang telah disepakati oleh administrator dan pemain (misalnya 12 periode).

Seperti halnya pada permainan manual, *Beer Game* ini juga dirancang agar dapat dimainkan oleh beberapa *cluster* (kelompok) sekaligus, dimana satu *cluster* terdiri dari 4 pemain (*factory*, *distributor*, *wholesaler* dan *retailer*) dan setiap *cluster* akan menerima *input* administrator (*customer order*) yang sama.

Berdasarkan skema tersebut maka dibuatlah konsep perancangan simulasi web dengan menggunakan *Class Diagram* seperti yang terlihat pada gambar 3.14 yang untuk selanjutnya digunakan sebagai arahan dalam membuat *web interface* untuk model *Beer Game*.



Gambar 3. 14 *Class diagram* untuk *web interface* model *Beer Game*

(Sumber: Olivia, R., Laura, 2007, hal. 65)

3.1.5. Identifikasi Variabel

3.1.5.1. Variabel yang dikoneksikan

Pada simulasi berbasis aplikasi web, pengguna (*user*) tidak akan berhadapan langsung dengan model. Model simulasi akan berada di server sehingga untuk mengakses dan menjalankan model tersebut diperlukan halaman web yang terhubung dengan model. Halaman web ini harus dapat menampilkan informasi dan petunjuk yang jelas yang dapat mengarahkan pemain untuk mensimulasikan model melalui web tersebut. Oleh karena itu, sebelumnya harus ditentukan variabel apa saja yang perlu dikoneksikan dengan web dan disimpan dalam *database* karena tidak semua variabel perlu diketahui oleh pemain. Dari keseluruhan 75 variabel yang ada, semuanya akan dikoneksikan oleh Powersim SDK, namun hanya 28 variabel yang akan disimpan dalam *database* dan dapat diakses oleh pemain dan administrator sebagai variabel yang akan menjadi *input* dan informasi pada setiap awal periode serta variabel laporan dan grafik pada akhir periode. Berikut ini adalah daftar variabel yang akan disimpan ke dalam *database*:

Tabel 3. 13 Daftar Variabel Yang Akan Dikoneksikan

No	Nama Variabel
1	Prod_Req
2	Prod_Del1
3	Prod_Del2
4	Fact_Inv
5	D_OrderPlaced
6	F_IncOrder
7	F_TotBacklog
8	F_TotCost
9	F_ShipDel1
10	F_ShipDel2
11	Dist_Inv
12	W_OrderPlaced
13	D_IncOrder
14	D_Backlog

Sumber: Olivia, R., Laura, 2007, hal. 66

Tabel 3.13 Daftar Variabel Yang Akan Dikoneksikan (Lanjutan)

No	Nama Variabel
15	D_TotCost
16	D_ShipDel1
17	D_ShipDel2
18	Wh_Inv
19	R_OrderPlaced
20	W_IncOrder
21	W_TotBacklog
22	W_TotCost
23	W_ShipDel1
24	W_ShipDel2
25	Ret_Inv
26	Customer Order
27	Ret_TotBacklog
28	R_TotCost

Sumber: Olivia, R., Laura, 2007, hal. 66

3.1.5.2.Variabel Input, Informasi dan Laporan

Sesuai dengan konsep permainan bahwa di setiap periode, pemain akan memasukkan *input* berupa order yang ditujukan pada sektor di atasnya, maka pada model ini yang menjadi variabel *input* adalah order dari setiap sektor yaitu:

Tabel 3. 14 Daftar Variabel *Input*

No	Nama Variabel
1	Prod_Req
2	D_OrderPlaced
3	W_OrderPlaced
4	R_OrderPlaced
5	Customer Order

Sumber: Olivia, R., Laura, 2007, hal. 67

Nilai variabel *input* ini dapat diubah oleh pemain. Ketika administrator memasukkan nilai *input* dan menjalankan model dengan menekan tombol *run* maka nilai *input* tersebut akan dikirim ke model yang terdapat di server untuk disimulasikan. Hasil dari simulasi tersebut dikirim kembali dan diterima pemain

di awal periode berupa informasi mengenai segala sesuatu yang berada dalam sektornya antara lain barang yang sedang dalam perjalanan (*shipping delay* 1 dan 2), baik dari sektor di atasnya maupun menuju sektor di bawahnya, jumlah inventori, order periode sebelumnya, *demand* yang diterima, dan total *backlog* yang belum terpenuhi. Variabel-variabel ini akan ditampilkan sesuai dengan sektornya masing-masing. Berikut ini adalah daftar variabel informasi:

Tabel 3. 15 Daftar Variabel Informasi

No	Nama Variabel
1	Prod_Req
2	Prod_Del1
3	Prod_Del2
4	Fact_Inv
5	D_OrderPlaced
6	F_IncOrder
7	F_TotBacklog
8	F_ShipDel1
9	F_ShipDel2
10	Dist_Inv
11	W_OrderPlaced
12	D_IncOrder
13	D_TotBacklog
14	D_ShipDel1
15	D_ShipDel2
16	Wh_Inv
17	R_OrderPlaced
18	W_IncOrder
19	W_TotBacklog
20	W_ShipDel1
21	W_ShipDel2
22	Ret_Inv
23	Customer Order
24	Ret_TotBacklog

Sumber: Olivia, R., Laura, 2007, hal. 67

Di akhir periode, pemain akan mendapat laporan hasil permainan selama periode yang telah dijalankan. Data yang akan ditampilkan pada laporan ini adalah order yang diberikan setiap periode, inventori, *backlog*, *demand* yang

diterima, dan total biaya hingga periode sebelumnya. Daftar variabel yang akan ditampilkan dalam laporan dapat dilihat pada tabel berikut:

Tabel 3. 16 Daftar Variabel Laporan

No	Nama Variabel
1	Prod_Req
2	Fact_Inv
3	D_OrderPlaced
4	F_IncOrder
5	F_TotBacklog
6	F_TotCost
7	Dist_Inv
8	W_OrderPlaced
9	D_IncOrder
10	D_TotBacklog
11	D_TotCost
12	Wh_Inv
13	R_OrderPlaced
14	D_IncOrder
15	W_TotBacklog
16	W_TotCost
17	Ret_Inv
18	Customer Order
19	R_TotBacklog
20	R_TotCost

Sumber: Olivia, R., Laura, 2007, hal. 68

Laporan ini nantinya akan ditampilkan dalam bentuk tabel HTML yang tidak dapat diubah oleh pemain, demikian pula dengan variabel informasi yang akan diterima pemain disetiap awal periode. Beberapa dari variabel laporan tersebut juga digunakan untuk menampilkan grafik biaya, order dan inventori setiap sektor maupun *cluster*. Variabel tersebut antara lain:

Tabel 3. 17 Daftar Variabel Grafik

No	Nama Variabel
1	Prod_Req
2	Fact_Inv
3	D_OrderPlaced
4	F_IncOrder
5	F_TotCost
6	Dist_Inv
7	W_OrderPlaced
8	D_IncOrder
9	D_TotCost
10	Wh_Inv
11	R_OrderPlaced
12	D_IncOrder
13	W_TotCost
14	Ret_Inv
15	Customer Order
16	R_TotCost

Sumber: Olivia, R., Laura, 2007, hal. 69

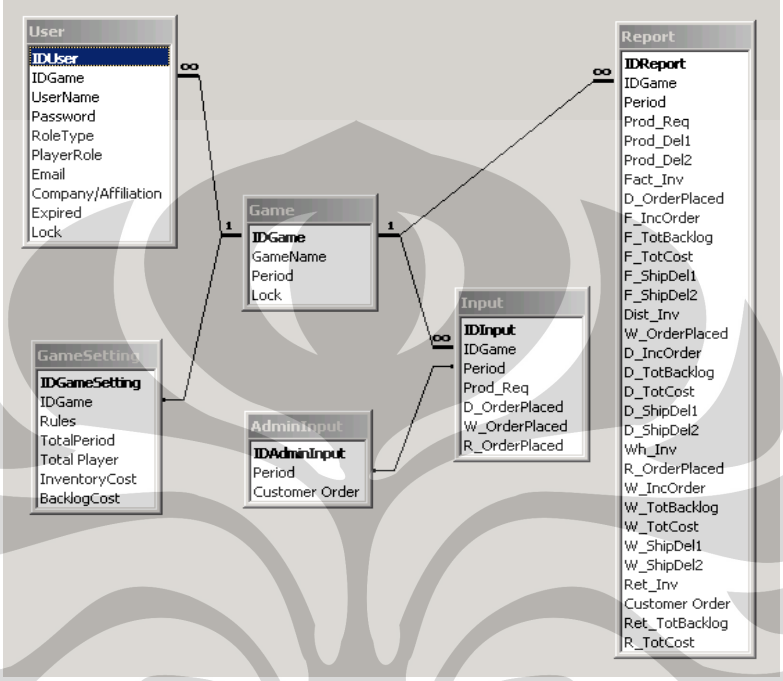
3.1.6. *Web Interface* untuk *Multi User*

Setelah mengidentifikasi dan mempelajari hubungan antarvariabel maka pembuatan *web interface* dapat mulai dilakukan. Pembuatan *web interface* dimulai dengan membuat *database* yang akan menyambungkan *web interface admin-side* dan *player-side*.

3.1.6.1. *Database Update*

Database yang sudah dibuat sebelumnya menggunakan *Ms. Access*. Untuk menyimpan data permainan, maka dibuat 6 tabel yaitu tabel Admin*Input*, tabel *Input*, tabel User, tabel Game, tabel GameSetting dan tabel Report yang masing-masing berisi *field* yang berbeda dan terhubung dengan satu *field* yang sama yaitu IDGame yang menunjukkan *cluster* permainan. *Field* pada tabel Report sesuai dengan 28 variabel yang akan ditampilkan sebagai informasi dan laporan di setiap periode. Pada penelitian kali ini, *database* yang digunakan adalah *database* yang telah dibuat sebelumnya dengan konsep yang sama namun mengalami pembaruan

data (*database update*) yang akan didokumentasikan ketika permainan baru dijalankan.



Gambar 3. 15 Diagram hubungan antar tabel
(Sumber: Olivia, R., Laura, 2007, hal. 78)

Berdasarkan diagram tersebut, *query* penulis kemudian membuat *query* yang menghubungkan beberapa tabel. *Query-query* ini dibuat untuk menampilkan data yang dibutuhkan oleh pemain terutama untuk laporan simulasi dan grafik setiap periode. Berikut ini adalah daftar *query* serta tabel pembentuknya:

Tabel 3. 18 Tabel Daftar *Query Beer Game*

No	Query	Tabel
1	PlayerProfile	Game
		User
2	Query*Input*	Admin*Input*
		User
		Game
3	QueryReport	Game
4	QueryGraph	Report
5	QueryReportFactory	Game
6	QueryReportDistributor	Report
7	QueryReportWholesaler	User
8	QueryReportRetailer	
9	QueryGraphFactory	
10	QueryGraphDistributor	
11	QueryGraphWholesaler	
12	QueryGraphRetailer	

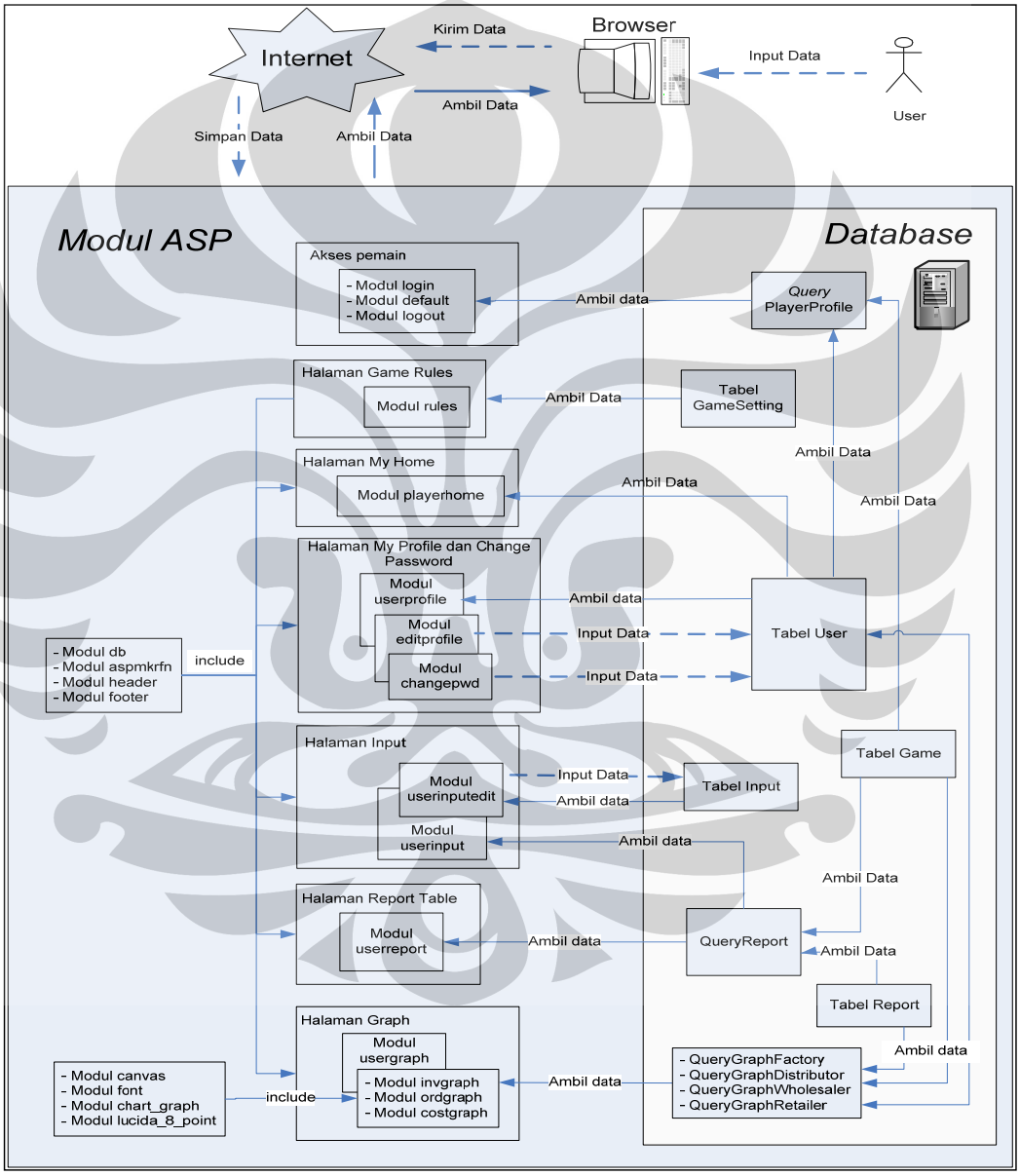
(Sumber: Olivia, R., Laura, 2007, hal. 78)

3.1.6.2. Konsep *Web Interface* dan Modul ASP pada *Player-Side*

Pada penelitian sebelumnya, semua data yang dibuat ditarik langsung dari *database* menggunakan fungsi SQL. Dan semua *file database* dan *file* simulasi diletakkan dalam direktori c:/wwwroot/inetpub/beergame. Hal penting lainnya adalah memastikan bahwa program IIS, Powersim SDK dan SQL dapat berfungsi dengan baik di komputer yang akan digunakan sebagai server.

Pembuatan *web interface* pada *player-side* dilakukan dengan membatasi otoritas pemain yang akan memainkan simulasi tersebut seperti halnya ketika memainkan simulasi tradisional. Batasan-batasan itu misalnya hanya dapat mengubah data pribadi dan input simulasi. Dengan batasan-batasan itulah maka dibuat konsep perancangan *web interface* pada *player-side*. Konsep *web interface* yang sudah dibuat pada penelitian sebelumnya menjadi dasar utama dalam pengembangan *web-based* yang penulis akan lakukan. Konsep perancangan ini akan menghubungkan modul-modul pada *player-side* sehingga dapat menampilkan *interface* simulasi.

Halaman-halaman web pada *player-side* disusun atas 32 modul ASP yang dibuat sesuai konsep perancangan dengan menggunakan UML *Class Diagram*. Modul yang telah dibuat pada penelitian sebelumnya merupakan acuan dalam melakukan pengembangan fitur-fitur yang diteliti oleh penulis. Konsep perancangan *web-interface* secara ringkas dapat dilihat pada gambar 3.16 berikut.

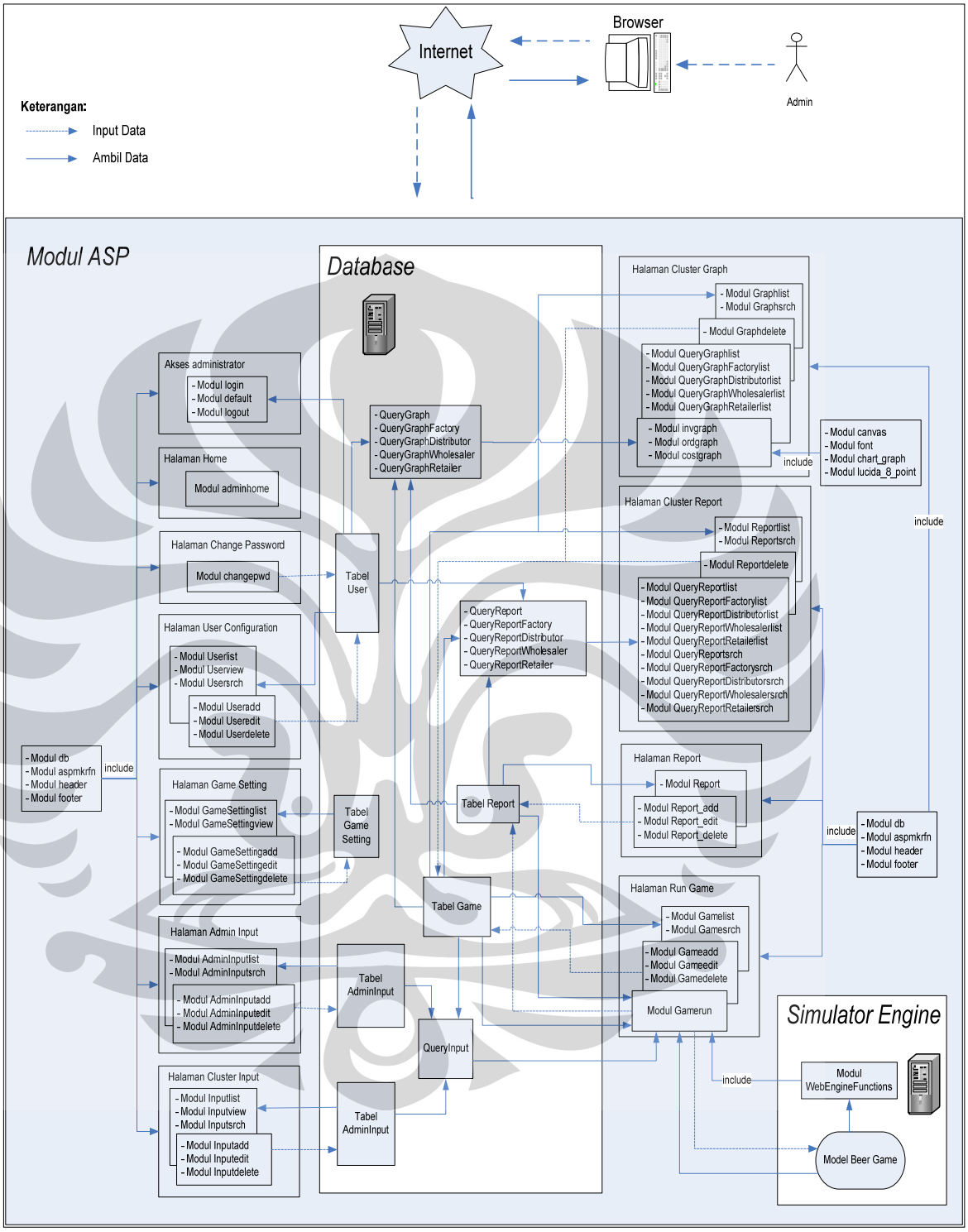


Gambar 3. 16 Skema *web interface* di sisi pemain

(Sumber: Olivia, R., Laura, 2007, hal. 78)

3.1.6.3. Konsep *Web Interface* dan Modul ASP pada *Administrator-Side*

Sama halnya dengan konsep perancangan pada *player-side*, maka konsep perancangan *web interface* pada *admin-side* yang digunakan pada penelitian sebelumnya dapat dilihat pada gambar 3.17 di bawah ini. Konsep ini akan menjadi titik tolak bagi penulis dalam melakukan pengembangan *web interface* pada *admin-side* terutama proses penarikan data yang akan ditampilkan pada grafik di tiap sektor. *Web interface* pada *admin-side* dibentuk oleh 82 modul yang saling dihubungkan satu dengan yang lainnya sehingga dapat menghasilkan halaman web yang interaktif dan komunikatif. Jumlah modul pada *admin-side* lebih banyak dari *player-side* karena halaman-halaman web administrator lebih kompleks karena terdapat sub menu *"Edit", "Add*" sampai sub menu *"Search*". Halaman web tersebut memiliki banyak sub menu karena administrator tidak hanya berhubungan dengan *database engine* tetapi juga dengan *simulator engine* sehingga data yang diolah oleh *admin-side* lebih banyak.



Gambar 3. 17 Skema *web interface* di sisi administrator

(Sumber: Olivia, R., Laura, 2007, hal. 78)

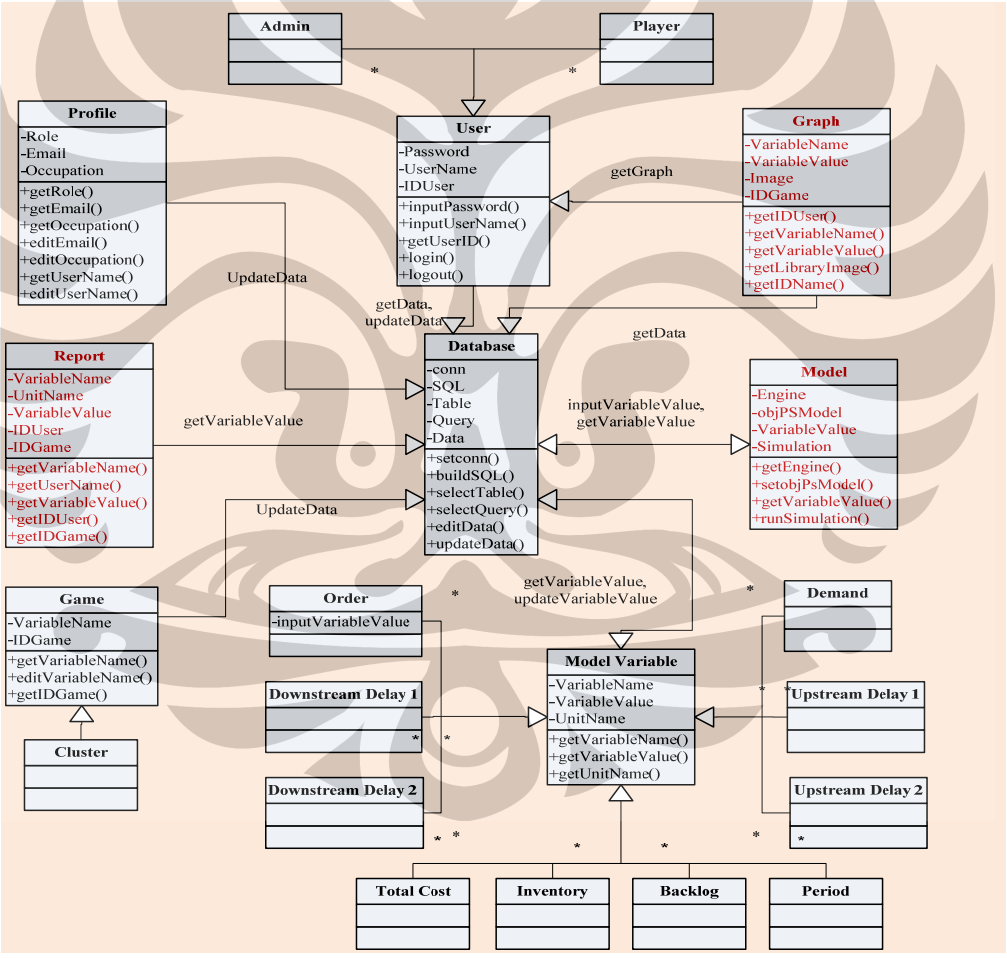
3.3. Pengembangan Fitur *Web Interface*

3.3.1. Konsep Pengembangan *Web Interface* Simulasi

Web-based simulation merupakan simulasi yang dibuat dengan menggunakan *script* pemrograman web yang menjadi ”jembatan” penghubung *simulator engine* dengan *player* sehingga *player* tidak harus memiliki *simulator engin*e untuk memainkan simulasi. Dengan demikian ketika merancang suatu simulasi berbasis aplikasi web, hal paling penting kedua setelah membuat model adalah merancang *web interface*. Konsep pengambangan *web interface* dilakukan berdasarkan metode perancangan simulasi yaitu konseptualisasi, formulasi dan verifikasi. Pengembangan *web interface* dari penelitian sebelumnya dilakukan dengan menganalisa bahwa beberapa fitur penting yang menjadi pendukung pembelajaran membutuhkan penyempurnaan. Tujuan dari penyempurnaan ini adalah pemain mendapatkan pengalaman dan pembelajaran tentang manajemen rantai pasok dengan memberikan mereka kejelasan hasil yang berupa tabel dan grafik yang komunikatif dengan visualisasi yang tepat. Dari hasil analisa dan studi literatur telah yang dilakukan, grafik yang telah dikembangkan pada penelitian sebelumnya membutuhkan penyempurnaan lagi sehingga lebih tervisualisasi dengan lebih jelas, baik dari segi informasi sampai pertimbangan ergonomis pemain yang membacanya. Dengan demikian, pengembangan yang dilakukan oleh penulis adalah penyempurnaan grafik yang dihasilkan oleh permainan simulasi ini baik pada sisi administrator maupun pada sisi pemain. Selain itu, sesuai dengan tujuan yang ingin dicapai dalam permainan ini bahwa pemain mendapatkan pengalaman yang tidak hanya bermain namun pembelajaran, maka diberikan tambahan fitur pada *web interface* pemain sehingga memungkinkan mereka dapat mengetahui kondisi tiap sektor setelah periode permainan berakhir. Dengan demikian, mereka dapat menganalisa kondisi permainan dan mendapatkan poin utama dari permainan *Beer Distribution Game* ini.

Penelitian ini merupakan pengembangan dari konsep dasar pada penelitian sebelumnya yang diarahkan pada penyempurnaan visualisasi variabel yang akan dimasukkan ke grafik hasil permainan. Konsep pengembangan terlebih dahulu dilakukan dengan membuat *Class Diagram* untuk mempelajari keterkaitan

antarvariabel yang akan dihubungkan sehingga dapat meletakkan dengan benar klasifikasi dan posisi fitur-fitur yang mengalami penyempurnaan. Secara umum, konsep pengembangan yang dilakukan sesuai dengan skema umum yang telah dibuat seperti pada gambar 3.13, namun secara khusus, pengembangan simulasi dapat dilihat dari penambahan *library* yang akan membentuk grafik yang lebih interaktif dan komunikatif. Hubungan antar kelas-kelas dari model simulasi yang akan disempurnakan dapat dilihat pada gambar 3.18. Kelas yang diberi kotak warna yang berbeda merupakan kelas yang mengalami penyempurnaan antar variabelnya.



Gambar 3. 18 *Class Diagram* pengembangan *web interface*

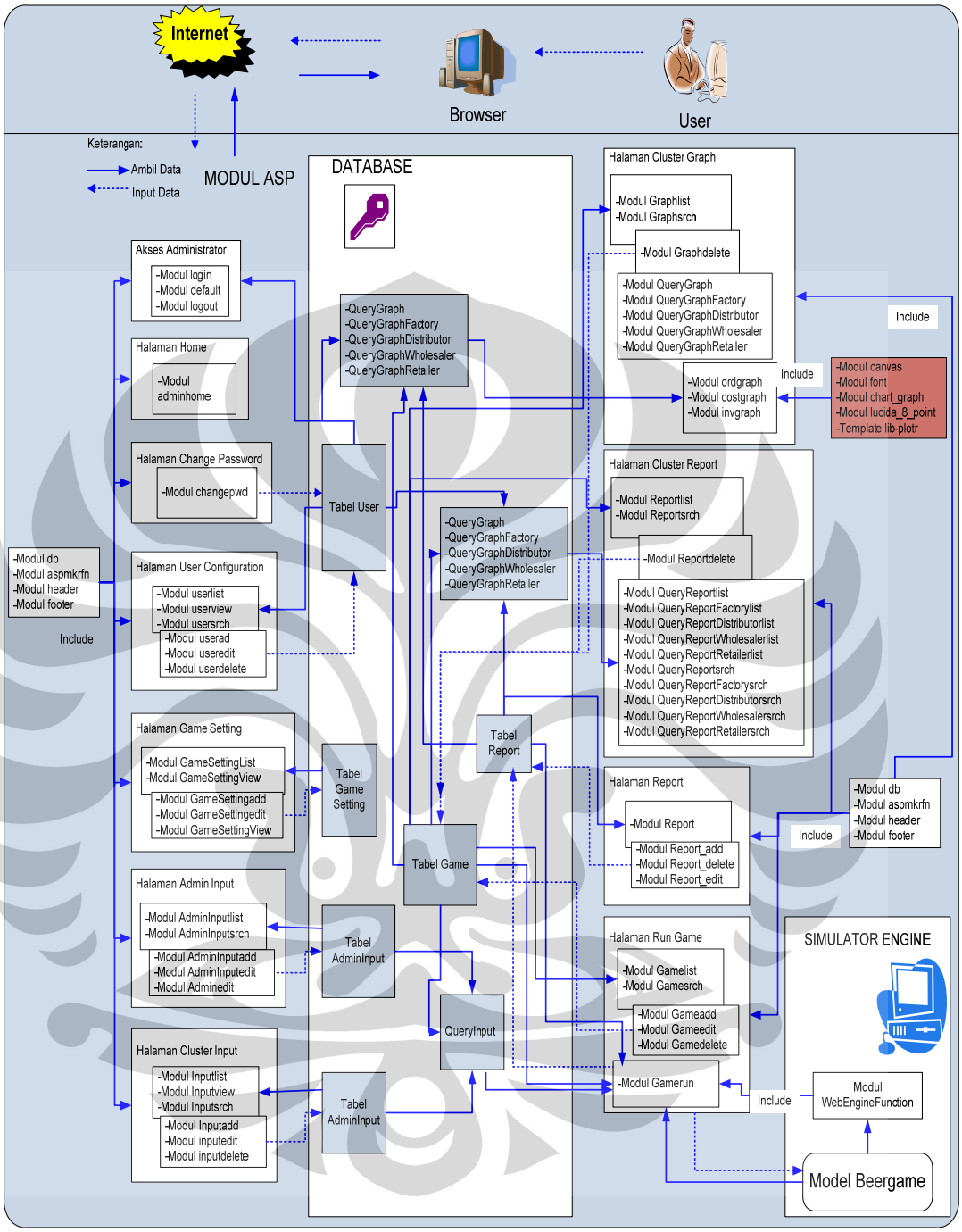
(Sumber: Penulis)

3.3.2. Penyempurnaan *Web Interface Admin-side*

3.3.2.1. Konsep Pengembangan *Web Interface Admin-side*

Dasar utama mengapa fitur grafik yang mengalami penyempurnaan adalah karena informasi yang ditampilkan melalui grafik sangat penting. Grafik menampilkan hasil permainan yang merupakan hubungan variabel sistem dinamis yang dimainkan dalam permainan ini. Dengan demikian karena informasi tersebut penting dan pemain dapat mengerti dengan lebih mudah akan inti permainan yaitu mempelajari fenomena *Bullwhip Effect* pada sistem rantai pasok, maka grafik menjadi fokus yang harus disempurnakan sehingga dapat tervisualisasi dengan lebih komunikatif dan informatif. Hal ini juga merupakan salah satu kriteria *web interface* yang harus dipenuhi ketika merancang suatu web aplikasi.

Untuk mendapatkan kriteria grafik yang tervisualisasi dengan tepat, proses navigasi yang sederhana, mudah dipahami dan terkoneksi dengan baik terhadap server maka berbeda dari metode yang digunakan sebelumnya, pada penelitian ini, grafik yang dibentuk akan ditarik dari *library* yang berisi plot gambar dan warna. *Library* tersebut akan dikoneksikan dengan *database engine* dan *simulator engine* dimana data hasil simulasi akan ditarik dan akan diterjemahkan dalam poin-poin dengan menggunakan bahasa ASP. Hasil yang akan ditampilkan dari grafik tersebut akan lebih dapat dibaca dan dimengerti dengan lebih mudah. *Library* tersebut harus diletakkan pada folder dimana *file beergame* disimpan. Untuk data penelitian ini, *file library* ini diletakkan pada *folder* lib pada direktori c://Inetpub/wwwroot/beergameORY4/. Jika penyimpanan *file library* tidak benar maka grafik pada web tidak akan muncul. Bagian lain pada modul simulasi pada sisi administrator ini tidak mengalami perubahan kecuali modul-modul ASP yang berhubungan dengan grafik dan kaitannya dengan *database engine* dan simulator *engine.* Secara keseluruhan, pengembangan *web interface* di sisi administrator pada penelitian ini dapat dilihat pada gambar 3.19 di bawah ini.



Gambar 3. 19 Skema pengembangan *web interface* di sisi administrator

(Sumber: Penulis)

3.3.2.2. Modul ASP *Admin-side* yang Dikembangkan

Otoritas yang dimiliki administrator sangat luas karena administratorlah yang akan menentukan jalannya simulasi yang sedang dimainkan. Dengan demikian administrator memiliki otoritas yang lebih luas dibandingkan dengan pemain sendiri dan hal ini yang menyebabkan *file* pembentuk halaman-halaman web di sisi administrator membutuhkan direktori khusus. Direktori tersebut yaitu c://Inetpub/wwwroot/beergameORY4/admin/.

Berikut ini merupakan penjelasan singkat mengenai modul-modul ASP pembentuk halaman-halaman web sisi administrator dan hubungan antar modul-modul tersebut dengan *simulator engine* dan *database engine* sehingga membentuk halaman web yang interaktif. Modul-modul tidak ditampilkan secara keseluruhan tetapi hanya menjelaskan modul-modul ASP yang berkaitan dengan membentuk grafik simulasi. Penjelasan modul tersebut adalah sebagai berikut:

1. *headerAdmin.asp*

Modul ini merupakan modul yang membentuk tabel pada menu utama pada *web interface* sisi administrator. Sesuai dengan konsep awalnya, maka modul ini akan selalu diikutsertakan dalam modul lainnya dengan fungsi --#include. Modul ini mengalami penambahan sintaks ASP yaitu dengan memasukkan *script* yang akan membentuk grafik jika halaman grafik dibuka. Kode ASP ditambahkan sebelum sintaks pembentuk *head* dari halaman web, kode tersebut adalah sebagai berikut:

```
<scriptsrc="../lib/plotr/lib/prototype/prototype.js"
type="text/javascript"></script>
<scriptsrc="../lib/plotr/lib/excanvas/excanvas.js"
type="text/javascript"></script>
<script src="../lib/plotr/plotr.js" type="text/javascript"></script>
```

Selain modul ini, beberapa modul lain seperti db.asp, aspmkrfn.as, dan footer.asp juga merupakan *file include* pada modul lainnya.

2. *db.asp*

Modul ini merupakan modul yang menginformasikan letak *path* dari *database*. Jikalau path *database* tidak benar karena salah penulisan maka simulasi tidak akan berjalan semestinya karena tidak ada data yang ditarik dari *database engine.* Jikalau dapat berjalan namun data yang akan masuk tidak akan masuk ke *database* yang semestinya. Modul ini termasuk *file include* karena simulasi permainan ini harus menarik data ataupun menyimpan data dari atau ke *database.* Contoh *path* pada modul ini adalah sebagai berikut:

```
Server.MapPath("\beergameORY3\beergame.mdb") & ";"
```

3. *Graphlist.asp*

Modul ini menampilkan 5 sub menu grafik untuk setiap *cluster* pada halaman *cluster graph*. Sub menu grafik tersebut adalah:

- *Cluster Graph* dihasilkan oleh modul QueryGraphlist.asp,
- *Factory Graph* dihasilkan oleh modul QueryGraphFactorylist.asp,
- *DistributorGraph* dihasilkan oleh modul QueryGraphDistributorlist.asp,
- *Wholesaler Graph* dihasilkan oleh modul QueryGraphWholesalerlist. asp,
- *Retailer Graph* dihasilkan oleh modul QueryGraphRetailerlist.asp.

Berbeda dari modul sebelumnya, modul yang membentuk ini mengalami perubahan akibat perbedaan logika dari *script* sehingga dapat menampilkan grafik yang lebih jelas dan interaktif. Perbedaannya terdapat pada logika penarikan data yang berasal dari *query* dan *field* dari mana data dipanggil berasal. Data yang dipanggil dari *database* dihubungkan dengan plot-plot yang terdapat pada *library* dengan poin-poin yang mewakili tiap data sehingga ketika dikoneksikan maka akan terbentuk garis grafik dan garis bilangan sumbu grafik yang lebih jelas. Setiap grafik memiliki *script* yang sama baik itu grafik *factory, distributor, wholesaler* dan *retailer*. Perbedaannya adalah sumber data grafik yang akan ditarik dari *query* dan *field* yang kemudian akan

disambungkan dengan *script* ke *library*. Berikut ini merupakan penulisan sintaks untuk poin-poin pembentuk grafik simulasi:

```
points=""
rs.Movefirst
Do While Not rs.eof
    if rs.eof then
            points=points&"["&rs("period")&","&rs("Fact_Inv")&"]"
    else
            points=points&"["&rs("period")&","&rs("Fact_Inv")&"],"
    end if
            'objSet.AddPoints Array(rs("Period"), rs("Ret_Inv"))
            rs.movenext
Loop
points2=""
rs.Movefirst
Do While Not rs.eof
    if rs.eof then
            points2=points2&"["&rs("period")&","&rs("Dist_Inv")&"]"
    else
       points2=points2&"["&rs("period")&","&rs("Dist_Inv")&"],"
    end if
            'objSet.AddPoints Array(rs("Period"), rs("Ret_Inv"))
            rs.movenext
    str="<script type='text/javascript'>"&_
                    "var dataset = {"&_
                        "'Factory Inventory':["&points&"],"&_
                        "'Distributor Inventory':["&points2&"],"&_
                        "'Wholesaler Inventory':["&points3&"],"&_
                        "'Retailer Innventory':["&points4&"],"&_
                        "};"&_
```

Script di atas digunakan pada modul-modul pembentuk grafik yaitu modul grafik pada sektor *factory, distributor, wholesaler* dan *retailer.* Selain itu, *script* di atas juga berlaku bagi modul *query* tiap sektor sehingga modul grafik tiap sektor dapat menarik data dengan benar dan *script* dapat membentuk poin-poin yang sesuai dengan data yang diterjemahkan ke bahasa ASP yang berada dalam modul *query* grafik.

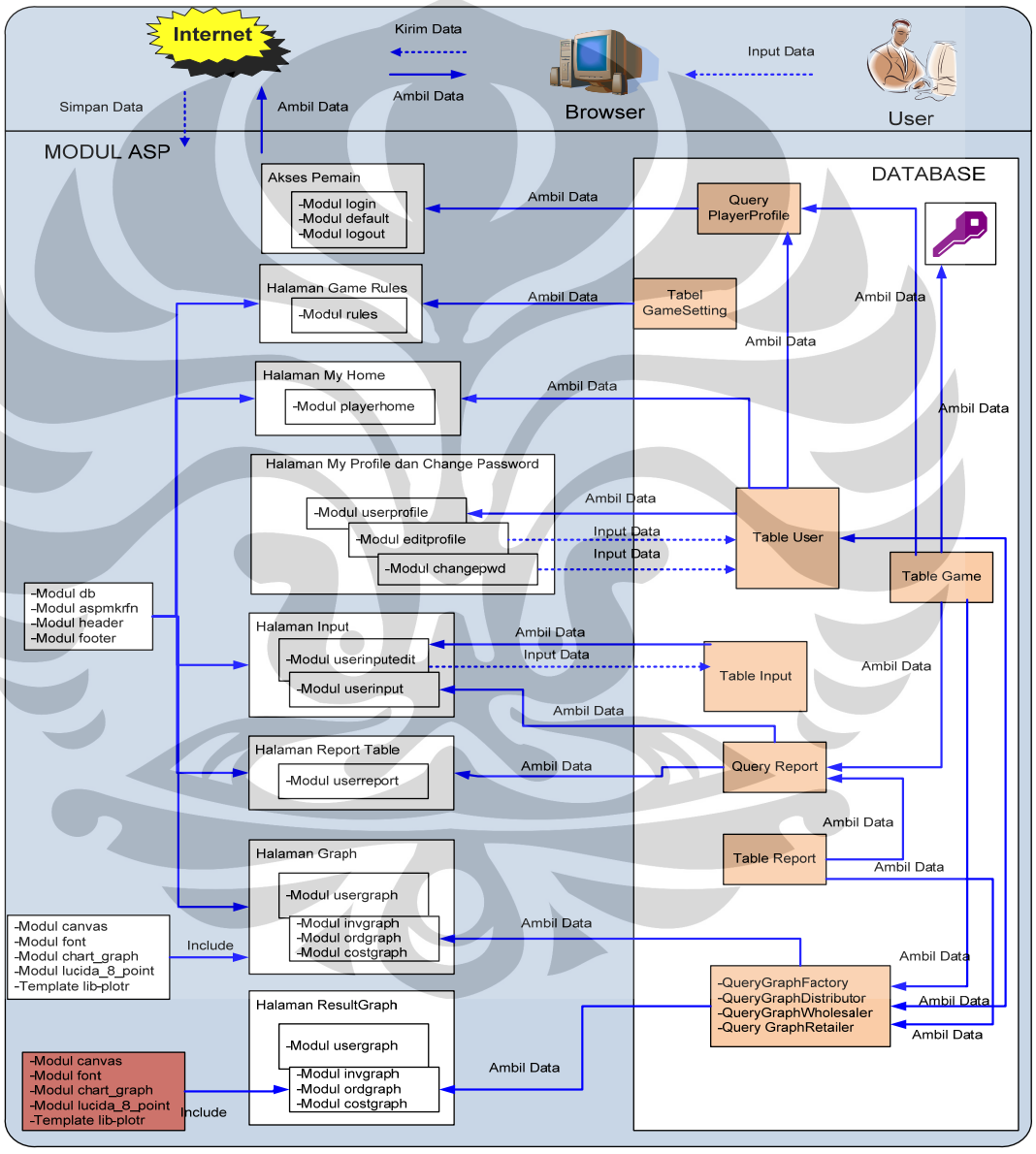
Dengan menggunakan *library* maka tampilan grafik menjadi lebih komunikatif. *Library* ini tidak hanya berisi *template* untuk plot garis, tetapi juga huruf dan pemilihan warna bagi grafik ataupun tulisan. Dengan *library* ini, maka pembuatan grafik menjadi lebih mudah dibandingkan dengan dibuat dengan logika persamaan linear seperti yang telah dibuat sebelumnya. Dengan tampilan grafik yang lebih jelas, maka pemain pun akan lebih tertarik dalam bermain dan memungkinkan mereka untuk dapat melakukan analisa pengambilan keputusan dengan lebih akurat juga. Setelah dilakukan pengujian terhadap sintaks, maka diperoleh bahwa penggunaan sintaks yang menghubungkan grafik dengan *library* tidak mengganggu jalannya permainan atau simulasi ketika dimainkan di *localhost*,

3.3.3. Penyempurnaan *Web Interface Player-side*

3.3.3.1. Konsep Pengembangan *Web Interface Player-side*

Pada awalnya, pemain memiliki akses yang terbatas pada permainan simulasi dan aturan tersebut masih berlaku pada penelitian ini. Pengembangan yang dilakukan oleh penulis adalah penambahan menu pada sisi pemain yaitu menu ResultGraph. Bertolak dari tujuan *Beergame* itu sendiri bahwa setiap pemain pada akhirnya dapat melihat fenomena *Bullwhip effect* pada sistem manajemen rantai pasok, dengan demikian di akhir permainan, pemain sebaiknya membutuhkan data pembanding yaitu hasil permainan yang tidak hanya dari sektornya sendiri tetapi juga dapat melihat kondisi secara keseluruhan dengan menggunakan grafik sebagai representasi yang paling memungkinkan untuk ditampilkan. Selain itu, grafik ini ditampilkan hanya pada periode terakhir yaitu setelah permainan selesai sehingga masih merepresentasikan sistem nyata dimana

tiap sektor tidak dapat melihat kondisi sektor lain ketika bermain. Dari kondisi tersebut maka pemain dapat melakukan analisa dan dengan analisa tersebut maka pemain dapat mencari solusi yang paling tepat untuk mengurangi efek negatif dari fenomena tersebut. Secara ringkas, konsep pengembangan dari pembuatan *web interface* sisi pemain adalah sebagai berikut.



Gambar 3. 20 Skema pengembangan *web interface* di sisi pemain

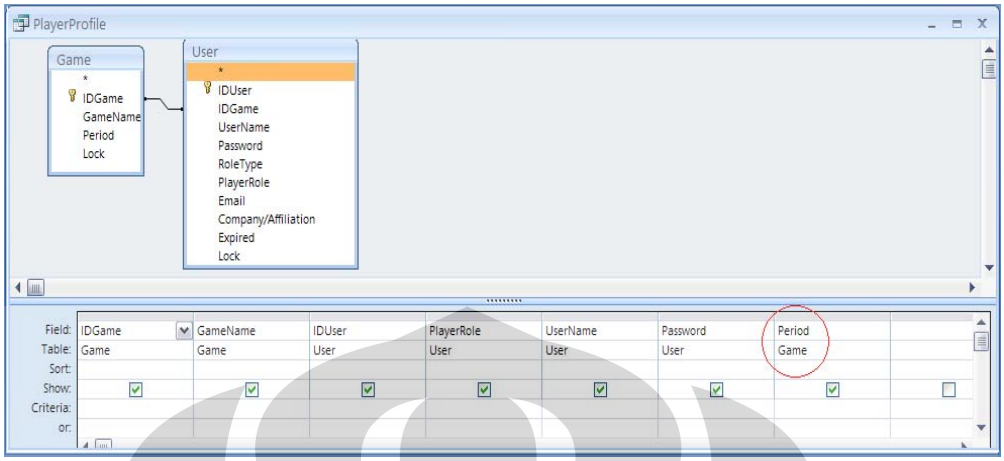
(Sumber: Penulis)

Konsep dalam penyempurnaan grafik pada *web interface* sisi pemain juga menggunakan logika yang sama dengan *web interface* pada sisi administrator. Grafik dibentuk oleh *script* ASP yang membentuk poin-poin yang terhubung dengan *library* dan ditarik dari *query* dimana *query* tersebut menarik data dari *database*. Jadi, halaman web pada sisi pemain adalah sebagai berikut:

1. *Login*
2. *My Home*
3. *Game Rules*
4. *My Profile*
5. *Input*
6. *Report Table*
7. *Graph*
8. ***ResultGraph***
9. *Change Password*
10. *Log Out*

3.3.3.2. Penambahan *Field* pada *Query Database*

Sebelum melakukan pembuatan *script*, maka dilakukan penambahan *field* pada *Query Database* yaitu pada *PlayerProfile*. Penambahan ini dilakukan agar menu Result Graph hanya bisa tampil pada halaman *home* pemain yang telah memasuki periode 12 dan tidak akan muncul jika pemain tersebut belum mencapai periode 12. Dari gambar 3.21 di bawah ini dapat dilihat bahwa *field query* ditambah dengan *periode* serta *table* diganti menjadi *game*. Hal ini juga berkaitan dengan penarikan data dari *database* dimana ketika pemain yang belum memasuki periode 12 memasuki halaman *login web* maka *script* ASP pada halaman login akan melakukan pengecekan nama pemain dan posisi periode pemain. Jika dalam *database* tidak ada *record* periode terakhir maka menu ResulthGraph tidak akan muncul karena tidak ada data yang ditarik dari *database.*



Gambar 3. 21 Skema pengembangan *web interface* di sisi pemain (Sumber: Ms.Access)

3.3.3.3. Modul ASP pada *Player-side* yang dikembangkan

Halaman-halaman web pada sisi pemain juga disusun atas modul-modul ASP yang saling berkaitan dengan yang lainnya dan data yang akan dimunculkan langsung ditarik dari *database*. Berikut ini merupakan penjelasan dari modul-modul yang mengalami penyempurnaan *web interface*. Modul-modul ASP pada sisi pemain tidak ditampilkan secara keseluruhan karena tidak mengalami perubahan kecuali beberapa modul berikut:

1. ***login.asp***

Halaman *login* pemain dirancang agar pemain dapat mengakses permainan dengan *account* yang berbeda-beda dan isi halaman web yang berbeda juga sesuai dengan *database* pemain tersebut yang ditampilkan pada *player home* masing-masing. Logika yang membentuk halaman *login* ini tidak mengalami perubahan. Kode yang sudah ada adalah kode yang berfungsi untuk menarik data berupa *password*, IDUser, IDGame, dan *username* dari *query player* profile untuk disimpan sebagi sesi. Jadi, modul ini mengatur akses *user* terhadap permainan. Pengaturan akses *user* tersebut yang kemudian pada penelitian ini disempurnakan. Modul ini dibuat agar dengan *script* dapat menarik data dengan benar terhadap pemain yang sudah memasuki periode 12,

sehingga diterjemahkan dengan bahasa ASP dan ditampilkan pada halaman web dalam bentuk akses ke menu ResultGraph. Bertolak belakang dengan hal tersebut, pemain yang belum menyelesaikan permainan sampai periode 12 tidak dapat mengakses menu ResultGraph. Modul ini sangat berhubungan dengan *database* yaitu pada *query player profile* yang telah diatur pada bagian sebelumnya.

2. ***header.asp***

Modul ini mengalami penambahan sintaks ASP yaitu dengan memasukkan *script* yang akan membentuk grafik jika halaman grafik dibuka. Kode ASP ditambahkan sebelum sintaks pembentuk *head* dari halaman web, kode tersebut adalah sebagai berikut:

```
        </style>
        <scriptsrc="lib/plotr/lib/prototype/prototype.js"
type="text/javascript"></script>
        <scriptsrc="lib/plotr/lib/excanvas/excanvas.js"
type="text/javascript"></script>
        <script src="lib/plotr/plotr.js" type="text/javascript"></script>
        <meta name="generator" content="ASPMaker v4.0.0.4" />
```

Selain itu, untuk mengatur penambahan menu ResultGraph yang hanya bisa muncul pada periode 12 saja, maka kode ASP yang menyusunnya bagian ini adalah:

```
If Session("usersaja_status") = "login" Then
if Session("period")=12 then%>
    <tr><td><spanclass="aspmaker"><ahref="ResultGraph.asp?cmd=rese
tall">Result Graph</a></span></td></tr>
        <%End If
```

3. ***ResultGraph.asp***

Menu ini merupakan tampilan grafik pada tiap sektor setelah satu *cluster* selesai dimainkan dan hanya tampil pada akhir periode. Modul ini dirancang dengan menghubungkan kode ASP pada modul ini dengan beberapa modul yang lain yaitu modul login.asp, header.asp, *file include* seperti db.asp dan dengan *query* grafik dari *database* yaitu pada modul *all_invgraph.asp, all_ordgraph.asp,* dan *all_costgraph.asp* sebagai *resume* grafik seluruh periode pada *cluster* dimana pemain tersebut berada. Keterkaitan yang terjadi akan menghasilkan grafik *order, cost* dan *inventory* selama periode permainan dari seluruh sektor. Berikut ini merupakan kode ASP dari modul ResulthGraph pada *web interface* sisi pemain.

```
<%
Response.expires = 0
Response.expiresabsolute = Now() - 1
Response.addHeader "pragma", "no-cache"
Response.addHeader "cache-control", "private"
Response.CacheControl = "no-cache"
%>

<%
If Session("usersaja_status") <> "login" Then
 Response.Redirect "login.asp"
End If
%>

<!--#include file="db.asp"-->
<!--#include file="aspmkrfn.asp"-->

<!--#include file="header.asp"-->
<script type="text/javascript" src="ew.js"></script>
```

```
<script type="text/javascript">
<!--
EW_dateSep = "/"; // set date separator
//-->
</script>

<p><span class="aspmaker"><b><h2>Inventory Graphic :</h2><b>
</span></p>
<div><canvas id="lines1" height="500" width="600"></canvas></div>
<!--#include file="all_invgraph.asp"-->

<p><span class="aspmaker"><b><h2>Order Graphic :</h2><b>
</span></p>
<div><canvas id="lines2" height="500" width="600"></canvas></div>
<!--#include file="all_ordgraph.asp"-->

<p><span class="aspmaker"><b><h2>Cost Graphic :</h2><b>
</span></p>
<div><canvas id="lines3" height="500" width="600"></canvas></div>
<!--#include file="all_costgraph.asp"-->
</table>
```

4. ***usergraph.asp***

Logika yang dibuat pada modul ini hampir sama dengan penyempurnaan grafik pada sisi admnistrator, yang membuat berbeda adalah posisi grafik dan grafik yang ditampilkan harus sesuai dengan posisi pemain apakah sebagai *factory* atau posisi yang lain. Modul *usergraph.asp* ini berfungsi mengatur agar 3 grafik hasil simulasi yaitu grafik inventori, total biaya dan total permintaan dapat ditampilkan pada halaman web pemain. Modul ini akan menampilkan grafik sesuai peran dari pemain dengan hubungan pada beberapa modul dimana modul-modul tersebut menarik data dari masing-masing *query*

yang diatur oleh sisi administrator. Agar administrator dapat menarik data dengan benar sesuai *account* pemain, maka data yang ditarik sesuai dengan IDGame yang sama dengan sesi IDGame pemain saat itu, hal ini tidak mengalami perubahan dari penelitian sebelumnya. Modul-modul grafik pada sisi pemain yaitu:

- fact_invgraph.asp
- fact_ordgraph.asp
- fact_costgraph.asp
- dist_invgraph.asp
- dist_ordgraph.asp
- dist_costgraph.asp
- wh_invgraph.asp
- wh_ordgraph.asp
- wh_costgraph.asp
- ret_invgraph.asp
- ret_ordgraph.asp
- ret_costgraph.asp

3.4. Proses *Hosting Web* Simulasi

Setelah melakukan penyempurnaan *web interface* maka penulis melakukan *hosting* web *Beergame Online.* Langkah-langkah dalam melakukan *hosting* web simulasi ini dapat dijelaskan singkat pada poin-poin berikut ini:

1. *Setting* Windows Server 2003

   Langkah ini merupakan pengaturan *Operating System Windows Server* 2003 sehingga dapat digunakan sebagaimana mestinya. Pengaturan ini meliputi instalasi Windows dan komponen IIS (*Internet Information System*).

2. Memindahkan *server* ke *data center* penyedia layanan internet (ISP)

   Server yang sudah diatur membutuhkan tempat untuk mendapatkan koneksi internet sehingga web yang dibuat dapat diakses oleh para

pemain. Server permainan ini diletakkan pada *data center* di PT. Globalport Binekatara dengan menggunakan pelayanan *colocation server*.

3. IP *address* diberikan oleh ISP

   Dari proses di atas, server akan diberikan alamat IP dari ISP agar server dapat terhubung ke internet. Dengan menggunakan IP *address* ini maka penulis juga dapat melakukan proses pengaturan jarak jauh (*remote*) terhadap *server*.

4. Instalasi *software database* dan simulasi (Ms Access dan Powersim SDK)

   Langkah selanjutnya merupakan instalasi Ms Access dan Powersim SDK. Kedua *software* ini merupakan *software* terpenting karena dalam simulasi ini, keduanya digunakan sebagai *database* dan *simulator engine* untuk menjalankan permainan ini.

5. *Upload* folder *Beer Distribution Game* ke *server* dalam direktori c://Inetpub/wwwroot/beergameORY3/.

   Setelah masa persiapan server selesai maka *file* yang berisi modul-modul ASP dan *file-file* pendukung lainnya sudah dapat di-*upload* ke server pada direktori yang benar sehingga ketika membuka domain dari simulasi, data simulasi dapat ditarik dengan benar.

6. Daftar domain ke *Yahoo! Small Business*

   Domain simulasi ini didaftarkan ke *Yahoo! Small Business* sehingga dengan domain tersebut, pemain dapat mengakses permainan ini. Domain yang telah didaftarkan yaitu *www.id-sims.com.*

7. Pointing IP address dan nama server ke domain

   Untuk memanggil data pada *server* maka IP *address* server harus dimasukkan ke data domain sehingga ketika domain diakses, maka data akan diambil dari server dengan IP *address* yang benar.

8. Mengakses nama domain *www.id-sims.com* dengan koneksi internet.
9. Jika sudah dapat diakses, maka verifikasi terhadap simulasi dapat dilakukan.

4. VERIFIKASI *WEB INTERFACE* DAN ANALISA

Berdasarkan konsep simulasi, maka setelah tahap formulasi selanjutnya adalah melakukan tahap verifikasi terhadap web yang telah dibuat. Verifikasi ini dilakukan dalam 4 tahap yaitu verifikasi *web interface* dari kedua sisi *interface* yaitu pemain dan administrator, kemudian memainkan satu *cluster* permainan dengan *multi-user* dan setelah itu melakukan akses web simulasi *beer game*.

4.1. Verifikasi *Web Interface* Sisi Administrator

Verifikasi terhadap halaman web sisi administrator dilakukan untuk memastikan bahwa tiap halaman web sudah terintegrasi dengan sempurna dan tiap halaman dapat berfungsi sesuai dengan tujuannya. Hal pertama yang harus dilakukan adalah mengakses *web interface* sisi administrator melalui alamat http://localhost/beergameORY4/admin/login.asp. Verifikasi yang dilakukan dibagi menjadi tiga tahap.

Tahap pertama adalah melakukan pengujian terhadap tiap halaman web, apakah tiap halaman dapat dibuka dan diakses dengan benar. Administrator memasuki tiap menu dan sub menu yang terdapat dalam web sisi administrator. Jikalau tiap halaman dapat terbuka dengan sempurna maka tahap pertama dinyatakan selesai.

Tahap selanjutnya merupakan tahap dimana administrator memasukkan atau mengubah data pada menu *Change Password* dan tiap sub menu yang ada yaitu *Edit, Add, Advance Search,* dan *Delete*. Jika pesan yang muncul adalah "*Record update is successful for key =*" untuk semua sub menu maka proses *update* data yang terdapat pada halaman administrator telah berhasil. Untuk memastikan data tersimpan dengan benar maka pengecekan dapat dilakukan melalui *database*. Dengan berhasilnya *update* data, maka tahap kedua telah selesai dilakukan.

Tahap terakhir adalah tahap menjalankan simulasi dengan submenu *Run*. *Input* data dari penelitian ini merupakan *output* pada penelitian sebelumnya dan model telah terkoneksi dengan baik, dengan demikian indikator verifikasi

submenu *Run* adalah pesan ”*Run Game is successful for Game with Key* = 1”. Angka yang terdapat pada pesan tersebut misalnya 1 atau 5 merupakan IDGame dari simulasi untuk membedakannya dengan *Game* yang lain. Jika tahap ini telah berhasil maka verifikasi *web interface* pada sisi administrator telah selesai dilakukan.

4.2. Verifikasi *Web Interface* Sisi Pemain

Sebelum memulai verifikasi, pemain harus mengakses terlebih dahulu *web interface* pemain melalui alamat http://localhost/beergameORY4/login.asp yang dapat dibuka melalui browser internet. Verifikasi halaman web pemain dilakukan terlebih dahulu dengan memasukkan *username* dan *password* pemain yang telah didaftar oleh administrator dimana tiap pemain sudah dibagi menurut *cluster-cluster* permainan. Pemain akan memasuki halaman *My Home* yang berisi menu dan submenu untuk dijalankan ketika simulasi dimainkan. Verifikasi *web interface* pada sisi pemain dilakukan untuk mengetahui apakah halaman web dapat diakses dengan sempurna dan data yang dimasukkan dapat tersimpan dengan baik pada *database*.

Verifikasi terhadap halaman web pemain dilakukan dalam dua tahap yaitu melakukan pengecekan apakah tiap halaman menu dan submenu dapat dibuka serta melakukan *update* data yang dimasukkan baik itu data permainan maupun data pemain. Menu pada halaman web pemain berbeda dari halaman web administrator. Halaman web pada pemain terdiri dari 9 menu dengan sub menu yaitu *Edit, Add, Advance Search,* dan *Submit*. Jika semua halaman dapat terbuka maka selanjutnya penulis memasukkan atau mengganti data pada submenu tersebut. Jika bagian ini berhasil maka akan muncul pesan ”*Update Record is Successful For Key* = 1“. Tahap kedua adalah melakukan verifikasi terhadap menu *Input* dari *web interface*. Pemain akan memasukkan nilai *input* ke dalam tabel yang telah disediakan untuk permaian pada periode tertentu. Setelah itu pemain menekan submenu *Submit* untuk mengirim data yang dimasukkan. Jika bekerja dengan baik maka akan keluar pesan “*Update Record is Successful for Key* = 1” dan hasil dari input yang baru dimasukkan dapat dilihat pada tabel input.

Sesuai dengan konsep permainan bahwa ketika semua pemain telah memasukkan *input* masing-masing sektor maka IDGame akan dikunci oleh administrator sehingga pemain tidak dapat memasukkan data ke dalam tabel *input* dan akan muncul pesan "*Sorry, the game is being locked!*" dan nilai input sebelumnya tidak akan berubah. Setelah simulasi dijalankan oleh administrator maka untuk mengetahui apakah data yang telah dimasukkan tersimpan dengan baik, pemain dapat mellihat *report* permainan pada menu *Report Table*. Untuk melihat pengembangan yang telah dilakukan, maka ketika pemain telah selesai bermain atau telah sampai pada periode terakhir, maka menu baru akan muncul yaitu menu *ResultGraph* yang akan memaparkan hasil permainan dari tiap sektor dalam bentuk grafik. Dari keseluruhan tahap yang dilakukan, dapat diketahui bahwa *web interface* pada sisi pemain sudah siap untuk dimainkan.

4.3. Verifikasi *Web Interface Multi-user* dalam *Localhost*

Setelah menguji tiap halaman web dapat berjalan dengan baik, maka tahap pengujian selanjutnya adalah tahap menguji jalannya permainan dengan menjalankan permainan simulasi dari dua sisi yang bersamaan yaitu dengan dari sisi pemain dan sisi administrator. Untuk permainan *localhost*, maka pengujian akan dilakukan oleh penulis sendiri melalui 5 peran yaitu administrator dan 4 pemain yang berperan sebagai sektor-sektor dalam rantai pasok. Pada verifikasi ini, maka data yang dipakai sebagai *input* adalah data yang terdapat pada tabel 3.12. Untuk melihat apakah web yang telah dikembangkan telah dibuat dengan benar, maka hasil yang akan diperoleh dari permainan dibandingkan dengan hasil yang telah diperoleh pada web sebelumnya akan menghasilkan grafik dan laporan yang sama. Dengan tujuan memudahkan proses dan membuatnya lebih sederhana, maka proses permainan *multi-user* pada simulasi ini mengalami beberapa perubahan. Permainan dimulai dari sisi administrator yaitu:

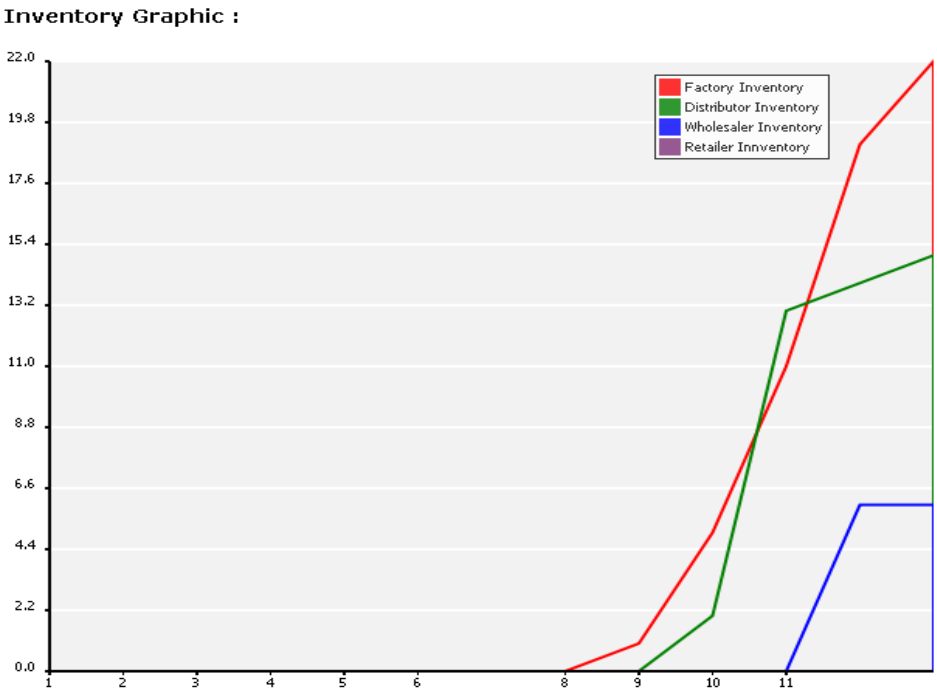
1. Administrator akan membuat *cluster* baru dengan nama *Beer Game Cluster* 7 (IDGame = 11), periode = 1 sampai 12 dan status = No di halaman *Run Game.* Periode tiap *cluster* dibuat terlebih dahulu dengan

nilai *input* untuk tiap sektor masih 0 sebagai data awal yang akan berubah ketika pemain telah memasukkan input permainannya.

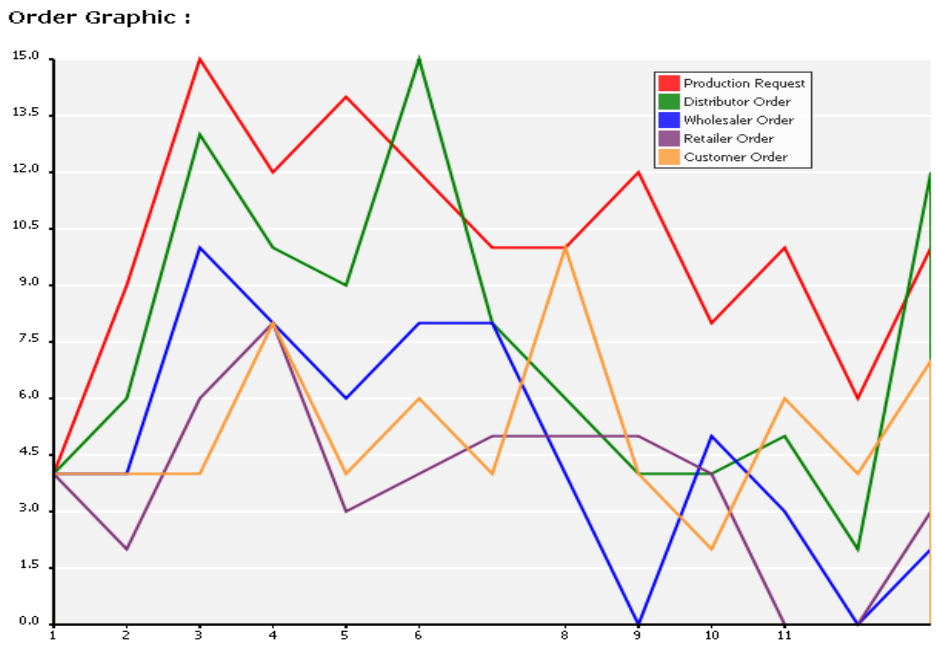
2. Administrator memasukkan data pemain baru untuk *Cluster* 7 pada halaman *User Configuration* dengan menggunakan Submenu *Add*.
3. Memasukkan nilai *customer order* selama 12 periode.
4. Membuat data baru untuk periode 0 di halaman *Report* pada submenu *Add* dengan IDGame = 11.

Setelah tahap di atas selesai, maka penulis akan masuk ke halaman web pemain dengan *username* dan *password* sesuai dengan data yang dimasukkan oleh administrator sesuai dengan *cluster* yang akan dimainkannya yaitu *cluster* 7 dengan IDGame = 11. Untuk *localhost*, maka penulis harus melakukan empat kali dalam memasukkan data *input* dengan peran yang berbeda-beda. Jika semua sektor telah memasukkan data *input* pada tabel maka penulis akan masuk lagi ke halaman web administrator untuk mengunci permainan pada periode tersebut sehingga data tidak dapat diubah lagi. Kemudian, simulasi dijalankan dengan submenu *Run* pada halaman *Run Game*. Jika simulasi berhasil maka pesan "*Run Game is successful for Game With Key* = 11". Setelah periode selesai, maka administrator akan kembali membuka *Game* yang baru di submenu *Edit* dan mengganti periode ke periode 2. Selanjutnya, akan kembali ke tahap selanjutnya dimana pemain akan menginput data pada periode selanjutnya. Pada periode terakhir yaitu periode 12, maka pemain dapat melakukan analisa terhadap kondisi keseluruhan yang terjadi pada sepanjang periode permainan melalui menu *ResultGraph*. Menu tersebut hanya akan muncul jika pemain telah sampai pada periode 12.

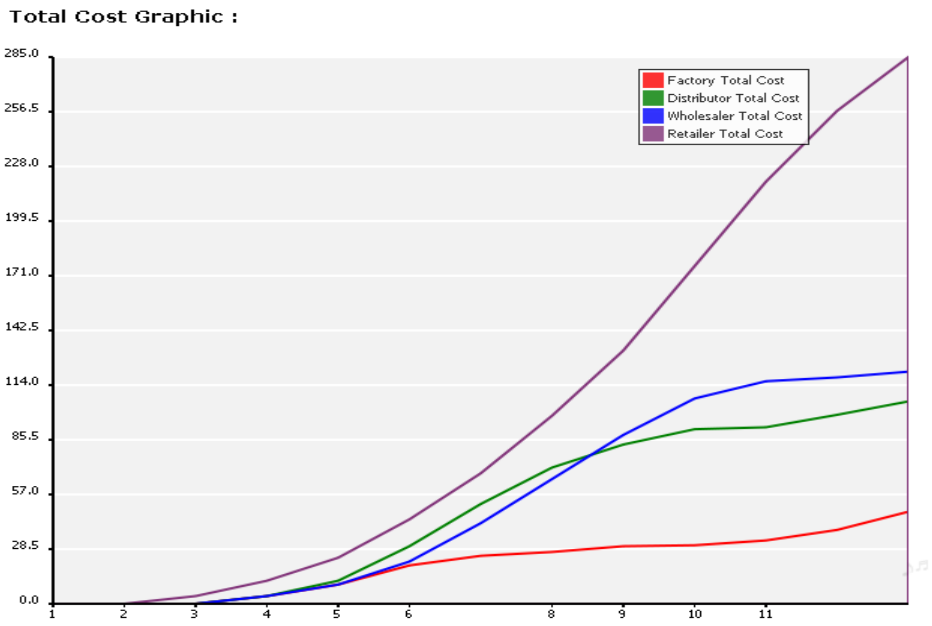
Hasil dari simulasi ini merupakan grafik dan tabel laporan tiap sektor. Pembandingan dilakukan terhadap web *localhost* yang telah dibuat pada penelitian sebelumnya. Berikut ini adalah grafik hasil simulasi *web-based simulation* sebelumnya dengan *web-based simulation* yang sudah dikembangkan dalam *localhost* pada *cluster* 7. Keterangan grafik adalah sumbu y mewakili unit barang dan sumbu x mewakili periode permainan.



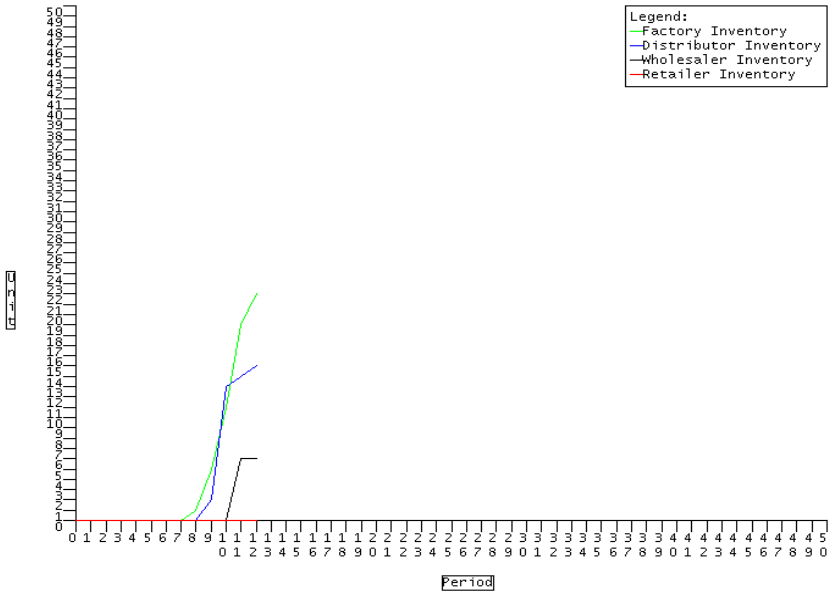
Gambar 4. 1 Pengembangan grafik *inventory cluster* 7



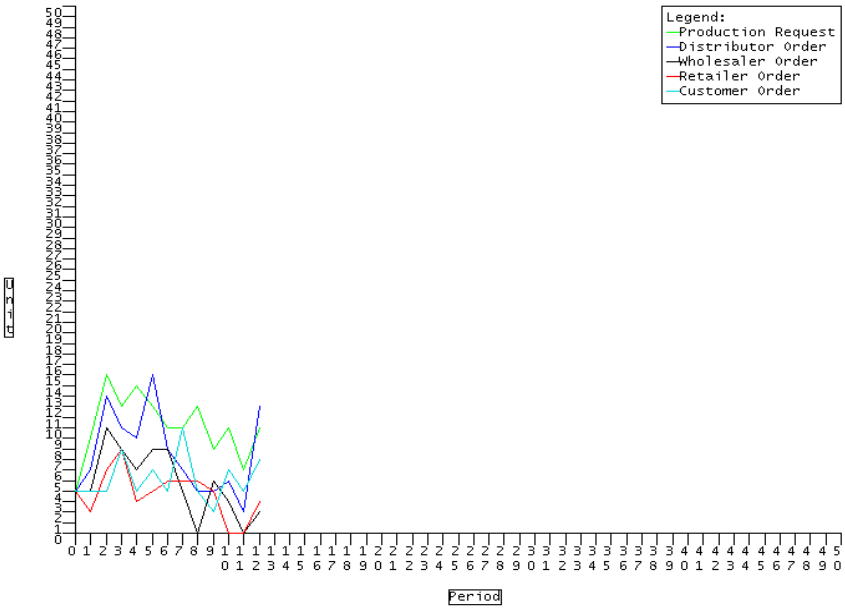
Gambar 4. 2 Pengembangan grafik order *cluster* 7



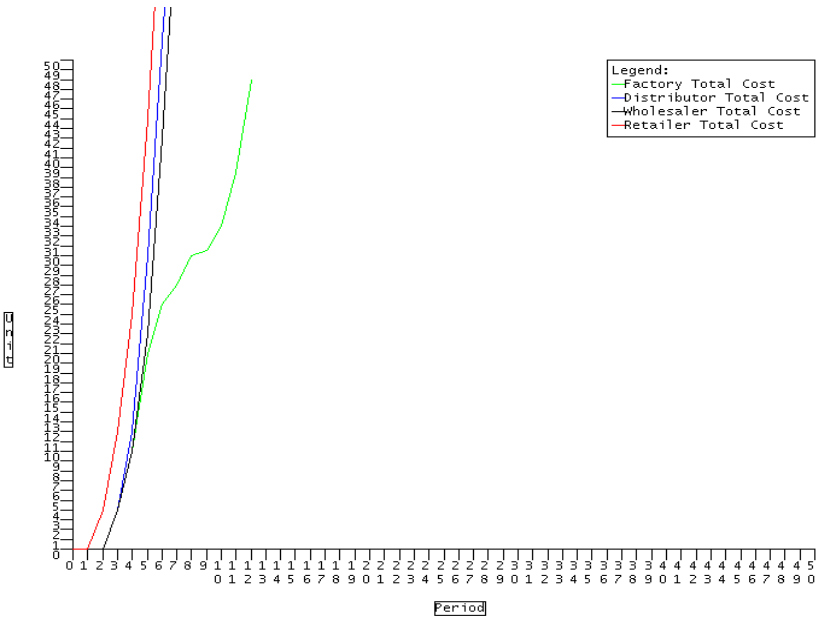
Gambar 4. 3 Pengembangan grafik total biaya *cluster* 7



Gambar 4. 4 Grafik *inventory pembanding* untuk *cluster* 7



Gambar 4. 5 Grafik order *pembanding* untuk *cluster* 7



Gambar 4. 6 Grafik total biaya *pembanding* untuk *cluster* 7

Berdasarkan hasil simulasi dari kedua web, maka hasil dari kedua grafik di atas sama. Dengan demikian, pengembangan *web-based simulation* yang telah dilakukan dapat berjalan seperti *web-based* yang telah dibuat sebelumnya dan secara tidak langsung juga akan menghasilkan grafik yang sama dengan model yang dibuat di Powersim Studio 2005.

Tabel 4.1 di bawah ini merupakan laporan hasil simulasi dari tiap sektor yang merupakan nilai dari variabel-variabel utama dari permainan ini. Dari data laporan inilah maka grafik di atas dapat dibentuk. Laporan *web-based* yang telah dikembangkan oleh penulis ternyata juga sama dengan laporan yang dihasilkan oleh *web-based* pada penelitian sebelumnya. Laporan tiap sektor tersebut adalah sebagai berikut.

Tabel 4. 1 Laporan Sektor *Factory*

IDReport	IDGame	GameName (*)	User Name (*)	Player Role	Period	Production Request	Production Delay 1	Production Delay 2	Inventory	Incoming Order	Backlog	Last Period Total Cost	Shipping Delay 1	Shipping Delay 2
91	11	Beer Game Cluster 7	benny	Factory	0	4 unit	0 unit	0 unit	0 unit	0 unit	0 unit	$ 0	0 unit	0 unit
92	11	Beer Game Cluster 7	benny	Factory	1	9 unit	4 unit	0 unit	0 unit	4 unit	0 unit	$ 0	0 unit	0 unit
93	11	Beer Game Cluster 7	benny	Factory	2	15 unit	9 unit	4 unit	0 unit	6 unit	4 unit	$ 0	0 unit	0 unit
94	11	Beer Game Cluster 7	benny	Factory	3	12 unit	15 unit	9 unit	0 unit	13 unit	6 unit	$ 4	4 unit	0 unit
95	11	Beer Game Cluster 7	benny	Factory	4	14 unit	12 unit	15 unit	0 unit	10 unit	10 unit	$ 10	9 unit	4 unit
96	11	Beer Game Cluster 7	benny	Factory	5	12 unit	14 unit	12 unit	0 unit	9 unit	5 unit	$ 20	15 unit	9 unit
97	11	Beer Game Cluster 7	benny	Factory	6	10 unit	12 unit	14 unit	0 unit	15 unit	2 unit	$ 25	12 unit	15 unit
98	11	Beer Game Cluster 7	benny	Factory	7	10 unit	10 unit	12 unit	0 unit	8 unit	3 unit	$ 27	14 unit	12 unit
99	11	Beer Game Cluster 7	benny	Factory	8	12 unit	10 unit	10 unit	1 unit	6 unit	0 unit	$ 30	11 unit	14 unit
100	11	Beer Game Cluster 7	benny	Factory	9	8 unit	12 unit	10 unit	5 unit	4 unit	0 unit	$ 30.5	6 unit	11 unit
101	11	Beer Game Cluster 7	benny	Factory	10	10 unit	8 unit	12 unit	11 unit	4 unit	0 unit	$ 33	4 unit	6 unit
102	11	Beer Game Cluster 7	benny	Factory	11	6 unit	10 unit	8 unit	19 unit	5 unit	0 unit	$ 38.5	4 unit	4 unit
103	11	Beer Game Cluster 7	benny	Factory	12	10 unit	6 unit	10 unit	22 unit	2 unit	0 unit	$ 48	5 unit	4 unit

Sumber: Penulis

Tabel 4. 2 Laporan Sektor *Distributor*

IDReport	IDGame	GameName (*)	User Name (*)	Player Role	Period	Order Placed	Upstream Delay 1	Upstream Delay 2	Inventory	Incoming Order	Backlog	Last Period Total Cost	Downstream Delay 1	Downstream Delay 2
91	11	Beer Game Cluster 7	olan	Distributor	0	4 unit	0 unit	0 unit	0 unit	0 unit	0 unit	$ 0	0 unit	0 unit
92	11	Beer Game Cluster 7	olan	Distributor	1	6 unit	0 unit	0 unit	0 unit	4 unit	0 unit	$ 0	0 unit	0 unit
93	11	Beer Game Cluster 7	olan	Distributor	2	13 unit	0 unit	0 unit	0 unit	4 unit	4 unit	$ 0	0 unit	0 unit
94	11	Beer Game Cluster 7	olan	Distributor	3	10 unit	4 unit	0 unit	0 unit	10 unit	8 unit	$ 4	0 unit	0 unit
95	11	Beer Game Cluster 7	olan	Distributor	4	9 unit	9 unit	4 unit	0 unit	8 unit	18 unit	$ 12	0 unit	0 unit
96	11	Beer Game Cluster 7	olan	Distributor	5	15 unit	15 unit	9 unit	0 unit	6 unit	22 unit	$ 30	4 unit	0 unit
97	11	Beer Game Cluster 7	olan	Distributor	6	8 unit	12 unit	15 unit	0 unit	8 unit	19 unit	$ 52	9 unit	4 unit
98	11	Beer Game Cluster 7	olan	Distributor	7	6 unit	14 unit	12 unit	0 unit	8 unit	12 unit	$ 71	15 unit	9 unit
99	11	Beer Game Cluster 7	olan	Distributor	8	4 unit	11 unit	14 unit	0 unit	4 unit	8 unit	$ 83	12 unit	15 unit
100	11	Beer Game Cluster 7	olan	Distributor	9	4 unit	6 unit	11 unit	2 unit	0 unit	0 unit	$ 91	12 unit	12 unit
101	11	Beer Game Cluster 7	olan	Distributor	10	5 unit	4 unit	6 unit	13 unit	5 unit	0 unit	$ 92	0 unit	12 unit
102	11	Beer Game Cluster 7	olan	Distributor	11	2 unit	4 unit	4 unit	14 unit	3 unit	0 unit	$ 98.5	5 unit	0 unit
103	11	Beer Game Cluster 7	olan	Distributor	12	12 unit	5 unit	4 unit	15 unit	0 unit	0 unit	$ 105.5	3 unit	5 unit

Sumber: Penulis

Tabel 4. 3 Laporan Sektor *Wholesaler*

IDReport	IDGame	GameName (*)	User Name (*)	Player Role	Period	Order Placed	Upstream Delay 1	Upstream Delay 2	Inventory	Incoming Order	Backlog	Last Period Total Cost	Upstream Delay 1	Upstream Delay 2
91	11	Beer Game Cluster 7	lany	Wholesaler	0	4 unit	0 unit	0 unit	0 unit	0 unit	0 unit	$ 0	0 unit	0 unit
92	11	Beer Game Cluster 7	lany	Wholesaler	1	4 unit	0 unit	0 unit	0 unit	4 unit	0 unit	$ 0	0 unit	0 unit
93	11	Beer Game Cluster 7	lany	Wholesaler	2	10 unit	0 unit	0 unit	0 unit	2 unit	4 unit	$ 0	0 unit	0 unit
94	11	Beer Game Cluster 7	lany	Wholesaler	3	8 unit	0 unit	0 unit	0 unit	6 unit	6 unit	$ 4	0 unit	0 unit
95	11	Beer Game Cluster 7	lany	Wholesaler	4	6 unit	0 unit	0 unit	0 unit	8 unit	12 unit	$ 10	0 unit	0 unit
96	11	Beer Game Cluster 7	lany	Wholesaler	5	8 unit	4 unit	0 unit	0 unit	3 unit	20 unit	$ 22	0 unit	0 unit
97	11	Beer Game Cluster 7	lany	Wholesaler	6	8 unit	9 unit	4 unit	0 unit	4 unit	23 unit	$ 42	0 unit	0 unit
98	11	Beer Game Cluster 7	lany	Wholesaler	7	4 unit	15 unit	9 unit	0 unit	5 unit	23 unit	$ 65	4 unit	0 unit
99	11	Beer Game Cluster 7	lany	Wholesaler	8	0 unit	12 unit	15 unit	0 unit	5 unit	19 unit	$ 88	9 unit	4 unit
100	11	Beer Game Cluster 7	lany	Wholesaler	9	5 unit	12 unit	12 unit	0 unit	5 unit	9 unit	$ 107	15 unit	9 unit
101	11	Beer Game Cluster 7	lany	Wholesaler	10	3 unit	0 unit	12 unit	0 unit	4 unit	2 unit	$ 116	12 unit	15 unit
102	11	Beer Game Cluster 7	lany	Wholesaler	11	0 unit	5 unit	0 unit	6 unit	0 unit	0 unit	$ 118	6 unit	12 unit
103	11	Beer Game Cluster 7	lany	Wholesaler	12	2 unit	3 unit	5 unit	6 unit	0 unit	0 unit	$ 121	0 unit	6 unit

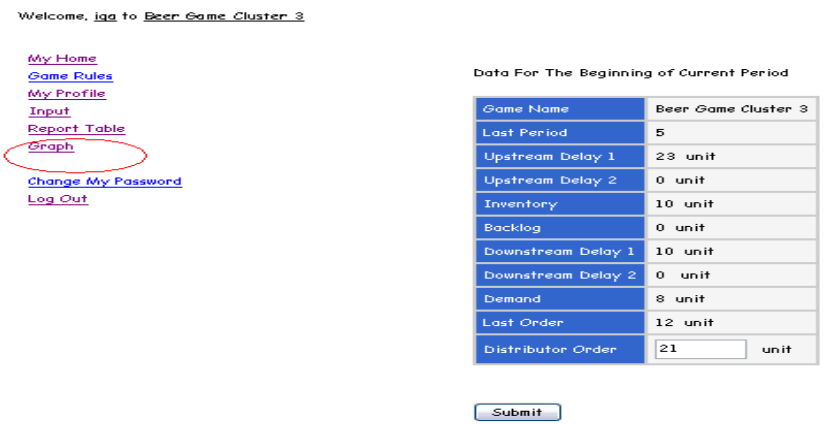
Sumber: Penulis

Tabel 4. 4 Laporan Sektor *Retailer*

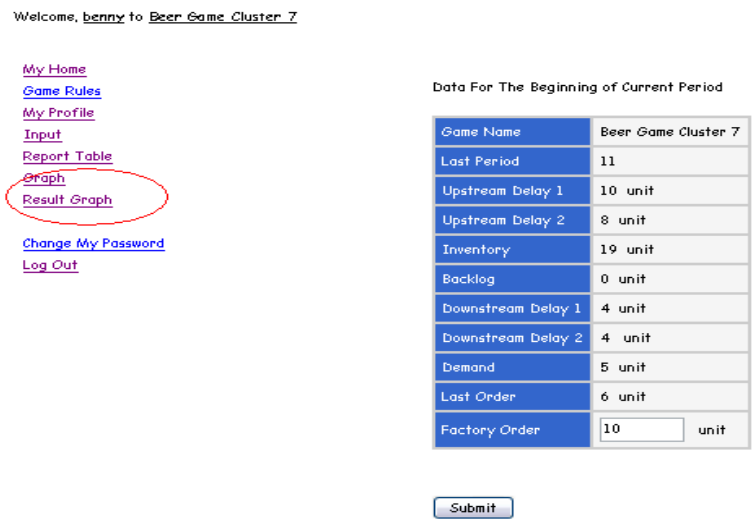
IDReport	IDGame	GameName (*)	User Name (*)	Player Role	Period	Order Placed	Upstream Delay 1	Upstream Delay 2	Inventory	Customer Order	Backlog	Last Period Total Cost
91	11	Beer Game Cluster 7	ria	Retailer	0	4 unit	0 unit	0 unit	0 unit	4 unit	0 unit	$ 0
92	11	Beer Game Cluster 7	ria	Retailer	1	2 unit	0 unit	0 unit	0 unit	4 unit	4 unit	$ 0
93	11	Beer Game Cluster 7	ria	Retailer	2	6 unit	0 unit	0 unit	0 unit	4 unit	8 unit	$ 4
94	11	Beer Game Cluster 7	ria	Retailer	3	8 unit	0 unit	0 unit	0 unit	8 unit	12 unit	$ 12
95	11	Beer Game Cluster 7	ria	Retailer	4	3 unit	0 unit	0 unit	0 unit	4 unit	20 unit	$ 24
96	11	Beer Game Cluster 7	ria	Retailer	5	4 unit	0 unit	0 unit	0 unit	6 unit	24 unit	$ 44
97	11	Beer Game Cluster 7	ria	Retailer	6	5 unit	0 unit	0 unit	0 unit	4 unit	30 unit	$ 68
98	11	Beer Game Cluster 7	ria	Retailer	7	5 unit	4 unit	0 unit	0 unit	10 unit	34 unit	$ 98
99	11	Beer Game Cluster 7	ria	Retailer	8	5 unit	9 unit	4 unit	0 unit	4 unit	44 unit	$ 132
100	11	Beer Game Cluster 7	ria	Retailer	9	4 unit	15 unit	9 unit	0 unit	2 unit	44 unit	$ 176
101	11	Beer Game Cluster 7	ria	Retailer	10	0 unit	12 unit	15 unit	0 unit	6 unit	37 unit	$ 220
102	11	Beer Game Cluster 7	ria	Retailer	11	0 unit	6 unit	12 unit	0 unit	4 unit	28 unit	$ 257
103	11	Beer Game Cluster 7	ria	Retailer	12	3 unit	0 unit	6 unit	0 unit	7 unit	20 unit	$ 285

Sumber: Penulis

Pada *web interface* sisi pemain, verifikasi yang dilakukan adalah verifikasi terhadap menu baru yang ditambahkan. Data pembandingnya adalah berasal dari *cluster* 3 yang belum memasuki periode 12 sehingga dapat diketahui apakah menu *ResultGraph* hanya muncul pada periode 12 atau tidak berhasil.



Gambar 4. 7 *Web-interface* pemain *cluster 3*



Gambar 4. 8 *Web-interface* pemain *cluster* 7

4.4. Verifikasi Web Simulasi *Beer Distribution Game Online*

Sebelum melakukan verifikasi, administrator harus melakukan tahap-tahap persiapan sehingga *server* dapat dipakai dengan baik.

Dengan berhasilnya verifikasi *web-interface* yang dijalankan pada *multi-user* maka secara tidak langsung ketika *web-based* dijalankan pada *server* yang telah dikoneksikan dengan internet, *web-based* dapat dimainkan seperti halnya ketika dimainkan di *localhost.* Verifikasi yang dilakukan pertama kali adalah dengan mengkoneksikan *browser* dengan internet dan mengakses alamat web www.id-sims.com. Jika sudah dapat dikoneksikan maka, halaman web simulasi akan langsung terbuka dengan tampilan *web interface* sisi pemain sehingga pemain dapat langsung memasukkan *username* dan *password*. Dengan jaringan internet, maka penulis tidak harus memasukkan data input permainan tiap sektor dengan satu komputer tetapi dengan *multi-user* secara langsung. Bahkan pemain pun dapat bermain dengan jarak yang saling berjauhan. Verifikasi web simulasi dilakukan seperti melakukan verifikasi dengan *multi-user* pada sisi pemain. Indikator keberhasilan halaman web dapat diakses juga sama dengan cara sebelumnya.

Untuk *web-interface* sisi administrator, administrator harus masuk ke link berikut ini: www.id-sims.com/admin/login.asp. Setelah berhasil masuk ke halaman web, administrator akan melakukan langkah-langkah seperti yang telah dijelaskan pada bagian 4.3. Setelah itu, maka verifikasi dilakukan seperti tahap-tahap yang telah dijelaskan pada bagian sebelumnya dan ternyata hasil yang diperoleh sama dengan hasil yang diperoleh pada *web-based simulation* pada *localhost.*

4.5. Analisa

Pengembangan *web-based simulation* yang telah dilakukan lebih difokuskan pada grafik dan menu baru sehingga permainan lebih menarik dengan tampilan data yang komunikatif. Pengembangan ini dilakukan dengan menggunakan *image generator* yang diletakkan dalam satu folder "lib" pada folder *beer game* sehingga dapat langsung diakses oleh *simulator engine* dan *database* melalui *script* ASP walaupun dipindahkan ke komputer lain. Grafik yang dihasilkan merupakan grafik yang lebih interaktif. Bagian lain dari *web-based* yang tidak berhubungan dengan pengembangan grafik dan menu baru tidak mengalami perubahan. Dengan tampilan grafik sederhana yang telah dihasilkan seperti pada gambar 4.1 sampai 4.3, maka saran-saran pada penelitian sebelumnya telah dapat dikembangkan sesuai dengan batasan-batasan yang telah ditetapkan pada awal penelitian ini. Walaupun masih dalam tahap pengembangan dari penelitian sebelumnya, grafik dan *web-design* secara keseluruhan sebaiknya terus dikembangkan sehingga lebih informatif. Selain itu, karena sistem *account* yang belum fleksibel, maka sebaiknya dibuat sistem *account* sehingga dapat didaftarkan secara langsung pada web simulasi.

Pengembangan selanjutnya adalah merancang web simulasi sehingga dapat dimainkan dengan menggunakan jaringan internet secara langsung sehingga tidak dalam cakupan *localhost* saja. Oleh karena itu, penulis melakukan tahapan-tahapan *web-hosting* sehingga *file beer game* dapat di-*upload* ke *server* yang telah mendapatkan koneksi ke jaringan internet. Hasil dari tahapan-tahapan yang telah dilakukan adalah web simulasi dengan alamat www.id-sims.com. Proses *web-*

hosting sebenarnya membutuhkan waktu yang lama. Oleh karena itu, pada penelitian selanjutnya sebaiknya *server* telah diatur secara berkala sehingga *file beer game* dan juga web simulasi tidak mengalami gangguan yang terlalu lama.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Output dari penelitian ini adalah permainan simulasi *Beer Distribution Game Online*. Program ini disimpan dalam direktori di *server* dimana *server* tersebut sudah dikoneksikan dengan internet sehingga ketika pemain mengetik nama domain yaitu *www.id-sims.com* maka pemain dapat langsung memainkan simulasi tersebut. Terdapat 2 program yang bekerja di *server* yaitu Powersim SDK dan Ms Access. Powersim berfungsi sebagai *simulator engine* dari modelnya sedangkan Ms Access berfungsi sebagai *database engine*. Sisi pemain hanya dihubungkan dengan *database* engine saja sedangkan sisi administrator berhubungan dengan kedua *software* tersebut. Komputer pemain tidak membutuhkan *software* tersebut karena sudah dihubungkan dengan bahasa ASP. Hal inilah yang menjadi kelebihan aplikasi berbasis web.

Penelitian ini merupakan pengembangan dari penelitian sebelumnya. Dengan demikian, proses pembuatan simulasi web ini dikembangkan dari proses pembuatan simulasi web sebelumnya. Berikut ini merupakan tahap pembuatan simulasi web:

1. Pembuatan model *Beer Distribution Game*
2. Verifikasi model *Beer Distribution Game*
3. *Database Update*
4. Pembuatan *web interface* sisi admin dan sisi pemain
5. Penyempurnaan fitur *web interface* pada sisi admin
6. Penambahan fitur *web interface* pada sisi pemain
7. Verifikasi web yang sudah disempurnakan
8. *Upload* simulasi ke *server*
9. Verifikasi web setelah dikoneksikan dengan internet

Verifikasi tahap akhir ini merupakan pengujian web agar dapat dimainkan melalui internet oleh pemain yang sudah didaftarkan oleh administrator.

Verifikasi dilakukan sesuai dengan aturan permainan yang sudah merupakan standar permainan *Beer Game* yang sudah ada. Verifikasi ini menghasilkan web dengan penyempurnaan grafik dan tambahan fitur pada sisi pemain yaitu fitur dimana setelah periode terakhir dimainkan, tiap pemain pada tiap sektor dapat melakukan analisa berdasarkan grafik hasil permainan dari seluruh sektor yang dikeluarkan oleh simulasi. Dengan demikian tiap pemain mendapatkan informasi untuk menarik kesimpulan terhadap adanya *Bullwhip Effect* yang dipelajari dalam simulasi ini. Pembelajaran dari simulasi juga mendukung proses pengambilan keputusan manajerial terhadap manajemen rantai suplai dari tiap sektor dan dapat melatih setiap pemain untuk berpikir kreatif dalam menemukan solusi dari permasalahan rantai suplai ini. Dari hasil verifikasi tersebut maka penulis telah berhasil mengaplikasikan simulasi model *Beer Game* dalam bentuk web.

Kelebihan dari hasil penelitian ini jika dibandingkan dengan penelitian sebelumnya adalah penyempurnaan grafik sehingga dapat mengkomunikasikan data dengan lebih jelas serta penambahan fitur agar pada periode terakhir pemain dapat melakukan analisa terhadap hasil yang diperoleh. Penulis juga mendokumentasikan modul pembelajaran dan panduan permainan dari penelitian ini agar dapat digunakan sebagai referensi pada pengembangan penelitian selanjutnya.

5.2. Saran

Pada pengembangan penelitian selanjutnya, sebaiknya web yang sudah ada dikembangkan dengan desain yang lebih menarik dan penambahan fitur sebagai hasil analisis dari tiap sektor. Pengembangan selanjutnya sebaiknya mempertimbangkan faktor ergonomi dan juga dapat dilakukan dengan menggunakan model lain yang lebih kompleks.

REFERENSI

Dessouky, Maged M., et. al. (2000, March). A Methodology For Developing A Web-Based Factory Simulator For Manufacturing Education. *Department of Industrial and Systems Engineering, University of Southern California. Los Angeles*.

Duke, Richard D. (1980, September). *A Paradigm For Game Design*, *Simulation & Games*, Vol. 11, No. 3.

Enciso, Ricardo Zamora. (2001). *Simulation Games, A Learning Tool*. ISAGA 2001 Proceedings Conference (International Simulation and Gaming Association).

Fishwick, Paul A. (1997). *Web Based Simulation*. Proceeding of the 1997 Winter Simulation Conference. University of Florida.

Harrel, Charles, et. al. (2000). *Simulation Using ProModel*. Mc Graw Hill.

Jacobs, F. Robert. (2000). *Playing The Beer Distribution Game Over The Internet*, *Production and Operation Management*, Vol. 9, No. 1.

Olivia, R, Laura. (2007). Perancangan Simulasi Sistem Dinamis Berbasis Aplikasi Web Menggunakan Powersim SDK dan ASP, Skripsi. Universitas Indonesia.

Nance, Richard E. dan Robert G. Sargent, (2002, January-February). Perspective on the Evolution of Simulation. *Operation Research*, Vol. 50, No. 1, 50th *Anniversary Issue*.

Newman, Frans. (2001). *Pemrograman Client/Server dengan ASP*. Jakarta: Elex Media Komputindo, Kelompok Gramedia.

Page, Ernest H., et. al. (2000, January). Web-Based Simulation: Revolution or Evolution?, *ACM Transactions on Modeling and Computer Simulation*, Vol. 10, No. 1.

Rausch, E. (1994), *Simulation and Games in Futuring and Other Uses*.

Riis. O. Jens. (1995). *Simulation Games and Learning in Production Management.* Denmark: Chapman & Hall.

Veith, Tamie L., et., al. (1998). World Wide Web-based Simulation. *International Journal Eng. Ed.*, Vol. 14, No. 5.

Wahidin. (2004). *ASP Untuk Orang Awam*. Palembang: Maxikom.

Zagal, José P., et. al. (2006, March). Collaborative games: Lessons learned from board games. *SIMULATION & GAMING*, Vol. 37 No. 1.

<http://www.albany.educprsdsconf2005proceedpapersSAMUR364.pdf
SAMUR364>

<http://www.clo_feature.asp.htm>

<http://www.systemdynamics.org/newsletters/2006Oct/L10-6.htm>