HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

ifa INSTITUTE FOR INTERNET TECHNOLOGIES AND APPLICATIONS

Entwicklung einer Bibliothek zur Simulation und Optimierung von Supply Chains

Bachelorarbeit
Abteilung Informatik

Autor: Mohamedou Moustapha Ba
Betreuer: Pr.Dr. Andreas Rinkel
Experte: Dr. U. Schimpel
Gegenleser: Pr.Dr. Luc Bläser

SCSOA

Supply Chain Simulation und Optimierung in Arena

Techinscherbericht

Dokumentinformationen

Änderungsgeschichte

Datum	Version	Änderung	Autor
11.11.2012	0.1	Dokument erstellt	mba
20.11.2012	0.2	Retailer Module Dokumentation	mba
29.11.2012	0.3	Supplier, Wholesaler, Factory hinzugefügt	mba
30.11.2012	0.4	Bilder und Abbildungen hinzugefügt	mba
05.12.2012	0.5	Statistische Erfassung von Messwerten	mba
11.12.2012	0.6	Testsimulation Dokumentieren	mba

Inhalt

Abbildungsverzeichnis

Einführung

Bei dem Begriff Supply chain Management (SCM) oder Lieferkettenmanagement(Wertschöpfungskette) handelt es sich um die Planung und Management aller Aufgaben bei Lieferantenauswahl, Beschaffung und Umwandlung sowie aller Aufgaben der Logistik.
Supply Chains agieren heutzutage in einer dynamischen Umwelt und sind einem permanenten Veränderungsprozess ausgesetzt. Zur Bewertung und zum Management von Supply
Chains wird meist ein modellbasierter Ansatz verfolgt, wobei vielfach Prozesskettenorientierte Beschreibungen verwendet werden.

Die Kernaufgabe von SCM liegt darin die Komplexität der Lieferkette aufrechtzuerhalten, die das Funktionieren von Koordinierung und Zusammenarbeit von den vorgelagerten(„upstream“)und nachgelagerten(„downstream“) Partner notwendig ist. Diese Lieferketten können einfach aber sehr komplex aufbaut sein. Wenn die Wertschöpfungskette sehr komplex ist, wird die Optimale Koordinierung sowie eine gute Zusammenarbeit der beteiligten Partner durch den sogenannten Bullwhip-Effekt erschwert. Der Bullwip-Effect resultiert dann durch die Erhöhung an Variabilität von Bestellungen einerseits und Lagerbeständen andererseits, wenn wir die Lieferkette hinaufwandern. Um das Bullwhip-Effekt-Problem zu lösen, braucht das SCM den Einsatz von Simulationssystemen. Simulationssystemen ermöglichen die Analyse und die Optimierung von Supply chain Systemen mit Hilfe von Simulationsmodellen.

1.1. Aufgabenstellung vom Betreuer

„Simulationssysteme werden zur Analyse oder zum Bewerten und Optimieren von komplexen Systemen eingesetzt; insbesondere, wenn sich die Systeme aufgrund ihrer Komplexität einer analytischen Lösung verschliessen. Sogenannte Supply Chains, als Teilgebiet der Logistik, stellen solche komplexe Systeme dar. Hier gilt es anhand einer Reihe von Parametern z.B. die Kosten für Lagerbestände, Transport, Verkauf, und Ressourceneinsatz zu optimieren.
Ziele der Arbeit:
1. Definieren von Bausteinen zur skalierbaren Modellierung und Optimierung von Supply Chains.
2. Implementieren einer Template Bibliothek in Arena. Arena ist eine Simulationsumgebung der Firma Rockwell und wird weltweit eingesetzt.
3. Nachweis der Einsetzbarkeit der Bibliothek durch Beispielsimulationen
4. Diskussion der Ergebnisse/Anforderungen mit Dr. U.
Schimpel, IBM Zurich Research Laboratory, dessen Team sich mit der gleichen Thematik auseinandersetzt, geplant ist hier eine Intensivierung der Zusammenarbeit zwischen der IBM und der HSR“

1.2. Ziel der Bachelorarbeit

Hauptziel der Arbeit war die Entwicklung einer Template Library in Arena zur Simulation und Optimierung eines 4-Stufigen Supply Chain Modells. Dazu gehörte einerseits die Einarbeitung in Arena Grundlagen und Arena Template Library Entwicklung, und andererseits das Studium von Supply Chain Konzepte sowie Simulationskonzepte.

1.3. Vorgehen /Aufbau der Arbeit

Für die Erreichung der Ziele dieser Bachelorarbeit wurden folgenden Schritte durchgeführt:

- Einarbeitung in Simulationskonzepte
- Einarbeitung in Arena (Grundlagen)
- Einarbeitung in Arena Template Library Entwicklung
- Supply Chain Konzepte studieren
- Supply Chain Simulationsmodelle in Arena studieren
- Anforderungen an Bausteine definieren
- Supply Chain Konzeptuelles Modell für das Projekt SCSOA erstellen
- Supply Chain Modellbausteine in Arena implementieren
- Arena Template Bibliothek (SCSOA) implementieren

- SCSOA-Module mit Beispielsimulationen in Arena testen.
- SCSOA-Supply Chain Modell in Arena aufbauen.

1.4. Einführung in Arena

1.4.1. Kurzbeschreibung von ARENA

Arena ist ein von der Rockwell Software Inc. entwickeltes Werkzeug, das es Systemanalytikern erlaubt, animierte Simulationsmodelle zu erschaffen. Arena bietet unter anderer folgenden Funktionalitäten an:

- Programm zur Modellierung und Simulation realer Prozesse unter Berücksichtigung von Zufallselementen.
- Basis : SIMAN
- Einsatz in der Entwicklung von Produktions-und Dienstleistungsprozessen
- Liefert Kennzahlen aus Simulationsläufen und Animationen der Prozesse
- Modellierung auf diversen Abstraktionsniveaus:
  - Unter Verwendung komfortabler Prozess-Elemente mit vorgefertigten Animationsmöglichkeiten
  - Unmittelbare graphische Abbildung von SIMAN-Programmen ohne Animation von Entities
  - Einbindung von SIMAN-Programmen bzw. VB bzw. C

1.4.2. Aufbau des ARENA-Programm-Fensters

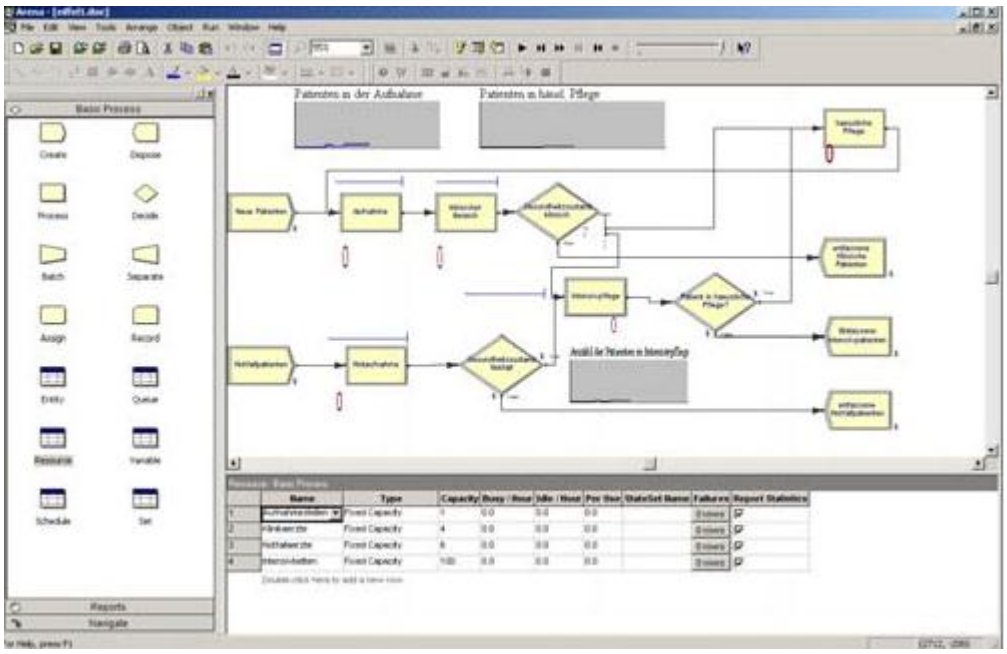


Abbildung 1: Aufbau des Arena Programm-Fensters

- In der links stehenden Projekt-Leiste finden sich Felder mit verschiedenen Muster-Prozessen und Spreadsheets.Weitere Felder können, falls nicht vorhanden, hinzugefügt werden(„attach“) und wieder entfernt werden („detach“)

- Rechts oben kann ein Modell unter Verschiebung von Prozess-Elementen aus der Projektleiste aufgebaut werden.
- Rechts unter können die Spreadsheets editiert und eingesehen werden.

1.4.3. Grundlegende Elemente von ARENA

Folgende grundlegende Elemente sind in Arena definiert:

- MODELS
- ENTITIES(ATTIBUTES)
- VARIABLES
- EXPRESSIONS
- PROCESSES
- SPREADSHEETS

Wichtig: Vermeidung von Umlauten und ähnliches.

1.4.3.1. MODELS

ARENA-Modelle sind gerichtete Graphen (Flussdiagramme), deren Knoten man als Prozesse bezeichnet (Ankunftsströme, Bedienstellen, Verzweigungen und Abgangsströme)
• Knoten verfügen je über einen oder mehrere Eingangs- und/oder Ausgangsstellen;
• Jede Ausgangsstelle (bzw. Eingangsstelle) eines Knotens muss über eine gerichtete Kante mit einer Eingangsstelle (bzw. Ausgangsstelle) eines Knotens verbunden sein.
• Jede auftretende Kante muss zwei Knoten verbinden.
• Die Graphen müssen aber nicht zusammenhängend sein (Die wechselseitige Steuerung von nicht verbundenen Teilgraphen ist in ARENA mittels Signalen möglich)
• Ablage von Modellen als Dateien mit der Endung (.doe). Modell und Simulationsergebnisse auch in gleichnamiger Access-Datenbank.

1.4.3.2. ENTITIES(ATTRIBUTES)

Auf den Kanten eines ARENA-Modells bewegen sich Entities, ausgestattet mit Attributen (z.B. Fertigungszustand, Gesamtwartezeit , Auftragsmenge,etc.).
Trifft ein Entity während der Simulation auf einen speziellen Prozess, so können hierdurch Vorgänge ausgelöst werden, z.B.:
• Abhängig vom Wert eines Attributes findet eine Bedienung statt
• Der Wert eines Attributes des Entity wird verändert
• Eine Kennzahl (Wartezeit, Kundenzahl, Rückstandzahl. . .) wird ausgegeben
• Der Zustand einer Ressource ändert sich (z.B. von IDLE zu ACTIVE)
• Die Länge einer Warteschlange zu diesem Prozess erhöht sich

1.4.3.3. VARIABLES

Neben den Entity-gebundenen Attributen ermöglicht ARENA auch die Verwaltung von Variablen, d.h. Modell-gebundenen Attributen. Trifft ein Entity auf einen Prozess, so hat man auf alle zuvor erklärten Variablen Zugriff bzw. kann auch neue Variablen definieren.

1.4.3.4. EXPRESSIONS

ARENA verfügt bei nahezu jedem Objekt oder Prozess über Schnittstellen, um dieses bzw. diesen zu konfigurieren. Dies kann durch Eingabe geeigneter Zahlwerte geschehen. Stattdessen ist es aber auch meist möglich, Ausdrücke einzusetzen, die einen Wert abhängig vom aktuellen Simulationszustand des Modells berechnen.
Über das Kontextmenü(rechte Maustaste) und die Auswahl von „Build Expression“ können solche Expressions erzeugt werden. Mit ihnen hat man Zugriff auf die Zufallsgeneratoren von ARENA, auf Systemvariablen und Attribute aktuell vorliegender Entities.
Es können arithmetische und andere mathematische Operationen integriert werden.

1.4.3.5. PROCESSES

Die wichtigsten Prozesstypen in ARENA sind die „basic processes“:

Abbildung 2: Arena Process Module

- Start- bzw. Endknoten eines ARENA-Modells müssen CREATE- (Quelle) bzw. DISPOSE-Prozesse (Senke) sein. Mit CREATE-Prozessen werden während der Simulation sog. Entities – auch als Batches - in das Modell eingeführt. Die Wartezeit zwischen zwei derartigen Zeitpunkten kann – z.B. mittels einer WS-Verteilung – spezifiziert werden. Mittels SCHEDULES können auch instationäre Ankunftsprozesse modelliert werden.
- PROCESS-Prozesse beinhalten die Bereitstellung einer Dienstleistung. Es können hierbei Ressourcen belegt und wieder freigegeben werden. Wenn alle Ressourcen eines PROCESS-Knotens belegt sind, werden wartende Entities in einer Warteschlange verwaltet, die bei Einrichtung der Ressourcen automatisch erstellt wird. Bedienzeit(Verteilung)en und Warteschlangendisziplinen können spezifiziert werden. Verfügbarkeit und Ausfälle von Ressourcen können berücksichtigt werden.

• DECIDE-Prozesse dienen der - zufälligen oder nach vom Modellierer festzulegenden Kriterien folgenden - Verzweigung innerhalb der Graphen.
• BATCH- und SEPARATE-Prozesse sammeln bzw. trennen Entities.
• ASSIGN-Prozesse ermöglichen an Ort und Stelle die Generierung von Variablen, Attributen etc., auf die im weiteren Verlauf zurückgegriffen werden kann
• RECORD-Prozesse ermöglichen den Export von benutzerdefinierten Kennzahlen während der Simulation

1.4.3.6. SPREADSHEETS

ARENA verfügt über eine Spreadsheet-Sichtweise auf verschiedene Modell-Objekte. Neben der Möglichkeit, Modell-Objekte „on the fly“, d.h während der Generierung von Prozessen zu erzeugen, ist dies auch innerhalb der Spreadsheets möglich

Die wichtigsten Spreadsheets in Arena sind:

Abbildung 3: Arena Spreadsheet Module

- ENTITY: Hier werden die verschiedenen Entity-Typen, die durch CREATE-Prozesse erzeugt werden können, aufgeführt. Hier lässt sich u.a. ihre visuelle Darstellung während der Animation ändern.
- QUEUE: Hier werden alle Warteschlangen des Modells aufgeführt.
- RESOURCE: Die verschiedenen Bedienstellen aller verwendeten PROCESS-Knoten
- VARIABLE: Alle verwendeten (benutzerdefinierten globalen) Variablen. Diese können über ROWS bzw. COLUMNS auch als (ein- bzw. zweidimensionale) Felder definiert werden. Eine Initialisierung ist möglich.

- SCHEDULE: Die Zeitpläne für die Verfügbarkeit von Ressourcen
- SET: Ermöglicht die Zusammenfassung gleichartiger Objekte (z.B. Entity-Typen oder Ressourcen) in einer Menge. Zugriff auf Objekte erfolgt über Indizes.

1.4.3.7. Simulationsläufe

Start der Modell-Simulation über das Run-Menü
• Im Run/Setup-Dialogfenster u.a. Festlegung von
– Anzahl unabhängiger Wiederholungen
– Dauer eines einzelnen Simulationslaufes

– Zeitskala für die Wiedergabe der Simulationsläufe
– Ausführungsform der Berichterstattung
• Simulation erfolgt durch Abarbeitung eines zu dem Modell gebildeten SIMAN-Programms.
• Intern:“Event-Calendar“:Zeitgesteuerte Verwaltung von Entities, d.h. deren Terminen und Attributen.

1.4.3.8. Weitere Features von Arena

Für detailliertere „high Level“-Darstellung sind weiter Prozesse-Elemente erforderlich. Dazu wird die Prozessleiste(AdvanceProcess.tpo) in die Projekt-Leiste hinzugefügt. Die Endung „tpo“ steht für Template Panel Object. Diese Datei wurde automatisch im Arena Template Library Tool aus der korrespondierende Template Library Datei(AdvanceProcess.tpl) mit dem Befehl „Generate tpo“ erzeugt. Folgende Erweiterungen der Prozessleiste sind im „Advanced Process Panel“ zu finden:

Abbildung 4: Arena Advanced Panel Process Module

- SEIZE (Belegung), DELAY (Verzögerung) und RELEASE (Freigabe) detaillieren die Vorgehensweise des PROCESS-Knotens.
- HOLD und SIGNAL: HOLD-Prozesse verzögern die Abarbeitung einer Queue durch Ressourcen, bis eine Bedingung erfüllt oder eine externes Signal (mit SIGNAL) gesendet ist
- READWRITE: Schnittstelle zur Daten-Ein- und Ausgabe

Dazu kommen Erweiterte Objekte: weitere Spreadsheet-Sichten auf Objekte finden sich ebenfalls im Advanced Process Template Panel:

Abbildung 5: Advanced Process Panel Spreadsheets

- FAILURE: Modellierung von Up- und Downtimes für den Ausfall von Ressourcen (Verweis des RESOURCE-Spreadsheet auf diese Tabelle)
- STATISTIC: Auf Basis von Attributen und Variablen können hier benutzerdefinierte Statistiken für die Simulation definiert werden.
- EXPRESSION: Häufiger verwendete Expressions können wie Funktionen an dieser Stelle abgelegt und benannt werden
- BLOCKS und ELEMENTS: graphische Darstellung von elementaren “SIMAN-Modellen mit BLOCKS-Templates aus der Projekt-Leiste. Verfügbarkeit von Entities, Attributes, Variables und Expressions über das ELEMENTS-Template. Auf diesem Niveau keine unmittelbare Animation möglich
  • Input-Analyzer: Aus Roh-Daten Schätzung WS-Modellen für Bedien- oder Zwischenankunftzeiten mit nachgelagertem Import in die zu erstellenden Modelle
  • PROCESS-Analyzer und OptQuest (aus dem Tools-Menü): Analyse-Programme, die dem Vergleich von Simulations-Szenarios bzw. der Auffindung optimaler Modelle und somit der Entscheidungsfindung etwa im Management-Bereich dienen.

1.5. Definitionen und Abkürzungen

Begriff/Abkürzung	Definition
SCSOA	Name des Projektes: Supply Chain Simulation und Optimierung in Arena
Arena	Simulationsumgebung von Firma Rockwell
Supply Chain	Wertschöpfungskette
Retailer	Einzelhändler in einer Wertschöpfungskette
Wholesaler	Grosshändler in einer Wertschöpfungskette
Factory	Fabrik in einer Wertschöpfungskette
Supplier	Zulieferer in einer Wertschöpfungskette
DC	Distribution Center

1.6. Referenzen

[1]http://www.actsolutions.it/File/Arena/Arena%20Template%20Developer's%20Guide.pdf
[2] http://www.actsolutions.it/File/Arena/Arena%20Basic%20Edition%20User's%20Guide.pdf

2. SCSOA-Projekt Übersicht

2.1. Fachliche Beschreibungen

Bevor es in Kapitel 4 mit der technischen Beschreibung um die Details der SCSOA-Implementierung gehen wird, folgt nun zunächst die fachliche Beschreibung der im Rahmen dieser Arbeit entwickelten Template Bibliothek. Anstatt auf das Wie und damit die Implementierung einzugehen, werden das zu Grunde liegende Modellbausteine und seine Anforderungen spezifiziert. Neben den getroffenen notwendigen Annahmen, die dem Anspruch genügen sollen, die komplexe Wirklichkeit in dem dieser Arbeit zu Grunde liegenden Supply Chain-Modell zweckmäßig zu vereinfachen, stehen
dabei in erster Linie die für die Simulation des Logistik- und Distributionsprozesses eingesetzten Bausteine im Mittelpunkt.

2.1.1. Annahmen

In diesem Supply Chain Simulationsmodell sind vereinfachende Annahmen zu treffen. Damit sollen die Ursache-Wirkungs-Zusammenhängen im Simulationsmodell als Supply Chain System zweckmäßig abgebildet werden können. Die Herausforderung besteht darin, trotz der vereinfachenden Annahmen einen hinreichend großen Informationsgehalt aufrecht zu erhalten. Das heisst, die Annahmen sollen zum einen nicht auf Kosten der Lösungsqualität getroffen werden und zum anderen das Anwendungsspektrum der Simulation nicht bereits im Vorfeld eingrenzen.

a. **Ein-Produkt-Fall**
   Um den Distributionsprozess nicht unnötig zu verkomplizieren, existiert in dem entwickelten Supply chain-Simulationsmodell nur ein einziges Produkt. In Anlehnung an die Volkswirtschaftslehre, die sich in der Mikroökonomie im Rahmen der Theorie der Unternehmung stets auf 1-Gut-Unternehmungen konzentriert, sollen die Erkenntnisse, die der 1-Produkt-Fall liefert, analog auch auf Mehr-Produkt-Fälle übertragbar angesehen werden.
b. **Gegebene Produktion beim Supplier**
   Die Produktion ist nicht Bestandteil der Optimierung. Die im Rahmen
   der Produktionsplanung und -steuerung notwendige Auftragsgrößenplanung

und Materialdisiposition werden nicht in Betracht gezogen. Die Produktion beim Supplier hat immer genug rohes Material für die Produktion.

c. **Transport**
Der eigentliche Warentransport von den Produktionsstandorten zu den Lagerstandorten und von dort zu den Kunden wird Einfachheit halber nicht explizit modelliert.
Allgemeine Bemerkungen zur Modellierung von Transportern in Arena

- ARENA-Objekt: Transporter
  - Carts, Handwagen, Trucks und Gabelstapler
- Modellierung von Transportvorgängen auf der Basis von Route setzt voraus:
  - Transportgerät (TG) ist immer verfügbar,
  - TG befindet immer an dem Punkt, an dem der Transport starten soll und
  - es treten keine zusätzlichen Verzögerungen auf
- Transportvorgänge sind komplexer, d.h.
  - Anfordern eines TG
  - Auswahl eines TG, bei mehr als einem freien TG
  - Ausgewähltes TG muss dann von seiner aktuellen Position zu der betreffende Beladenposition fahren
- Generische Ausdruck Transporter
  - eine oder mehrere identische „transfer devices“ (Ressourcen), die durch Entitäten zum Zweck der Bewegung von einer Station zu anderen belegt werden können.
- Der Transport einer Entität mit einem Transporter erfordert drei Aktivitäten:
  – Anfordern eines Transporters (Request)
  – den eigentlichen Transport (Transport)
  – die Freigabe des Transporters (Free)

2.2. Supply Chain modellbausteine

Im Anschluss folgt eine kurze fachliche Beschreibung der verwendeten Komponenten:

2.2.1. Retailer

Die Komponente **Retailer** ist Ausganspunkt der gesamten Wertschöpfungskette. Sie deckt als Einzelhandel den Bedarf der Endverbraucher ab und bestellt Ware beim Lieferanten (Wholesaler), sobald sein individueller Meldebestand unterschritten ist. Unter Meldebestand versteht man diejenige Lagerbestandsmenge, bei deren Erreichen oder Unterschreiten einen Nachschubauftrag ausgelöst wird. Dieser Bellvorgang ist als der auslösende Prozess anzusehen, der den Gesamtprozess der Simulation anstößt.

2.2.2. Wholesaler

Die Komponente **Wholesaler** ist der Nachfolger vom **Retailer** in der Lieferkette. Der Wholesaler bearbeitet den Nachschubauftrag vom Retailer und liefert die Bestellte Menge nach einer festgelegten Lieferzeit beim Retailer aus. Sobald sein individueller Meldebestand erreicht ist, bestellt er eine Nachschubmenge bei seinem Lieferanten (Factory). Während dieses Beschaffungszeitraumes werden alle eingegangenenNachschubaufträge des Retailers in die Rückstände registriert.
Nach einer Bestellung liegen alle bestellten Artikel bis zum Versand erst einmal in Ihren Rückständen.
Mit Hilfe der Rückstandsabfrage kann überprüft werden, ob bestellte Waren eingetroffen sind und zum Versand bereits liegen. Danach wird die Ware beim Retailer ausgeliefert.

2.2.3. Factory

Die Komponente **Factory** ist der Nachfolger vom Wholesaler in der Lieferkette. Die Factory bearbeitet den Nachschubauftrag vom Wholesaler und liefert die Bestellte Menge nach einer festgelegten Lieferzeit beim Wholesaler aus. Sobald sein individueller Meldebestand erreicht ist, bestellt er eine Nachschubmenge bei seinem Lieferanten (Supplier). Während dieses Beschaffungszeitraumes werden alle Nachschubaufträge des Wholesalers in die Rückstände

registriert. Wenn wieder genug Bestand vorhanden ist, werden die Rückstände kompensiert und die Ware wird beim Wholesaler nach einer Verzögerung ausgeliefert. Die Komponente Factory stellt Produkte für den Wholesaler her.

2.2.4. Supplier

Die Komponente **Supplier** ist der Nachfolger von der Factory in der Lieferkette und befindet sich am Ende der konzipierten Lieferkette. Der **Supplier** stellt Komponenten für die Factory her. Er bearbeitet die Nachschubaufträge von der Factory und liefert die bestellte Menge an Komponente bei der Factory nach einer festgelegten Lieferzeit aus. Es gibt keine Teillieferung oder Rückstandsliste beim Supplier. Sobald sein Minimalbestand unterschritten ist, wird die Produktion neue gestartet. Die Komponente wird produziert bis den Höchstbestand erreicht ist. Während dieser Zeit werden alle eingegangenen Nachschubaufträge von der Factory nicht erfüllt und als entgangene Aufträge beim Supplier registriert.

2.3. Information -und Materialfluss

Charakteristisches Merkmal des zu Grunde liegenden logistischen Prozesses entlang der Wertschöpfungskette ist die Asymmetrie des Informations- und Warenflusses. Abb. XX verdeutlicht dieses Phänomen bezogen auf die Komponenten der Logistiksimulaton.

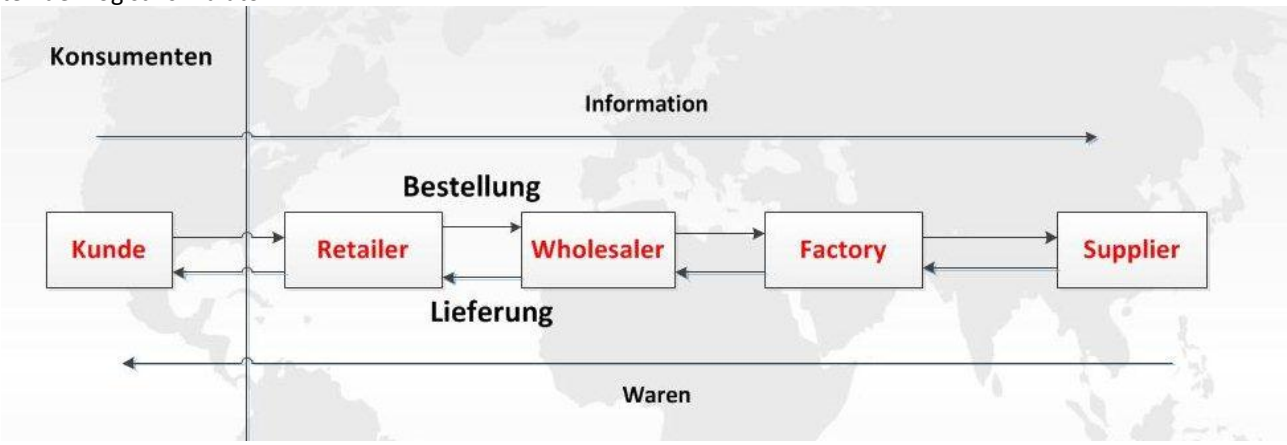


Abbildung 6: Informations -und Warenfluss entlang der Lieferkette

Der Informationsfluss beginnt mit der Warenbestellung eines Konsumenten, die eine neue Position in der Auftragsliste des betroffenen Retailers auslöst. Damit die Retailer Komponente den Bedarf ihrer Kunden decken kann, bestellt sie ihrerseits Ware bei beim Wholesaler. Sobald ein Wholesaler eine Bestellung erhält, setzt der Warenfluss in entgegen gesetzter Richtung ein. Diese gleiche Interaktion besteht zwischen Wholesaler und Factory aber auch zwischen Factory und Supplier.

2.4. Technische Beschreibungen

In den folgenden Kapiteln werden die einzelnen Bausteinen des Supply Chain Modells auf der Implementierungsebene beschrieben. Für jede Komponente wird auf die Implementierungen Ihres Arena Modells eingegangen. Und anschliessend wird die daraus entwickelte Template Bibliotheken detailliert beschreiben.

2.4.1. Retailer-Modell

Das Retailer-Modell soll einen Einzelhändler in einer Lieferkette simulieren. Die Funktionalität eines Retailer wird in der Logik des Retailer-Modelles definiert. Der Retailer befindet sich am Anfang der Lieferkette und kommuniziert direkt mit dem Endkunden.

2.4.1.1. Retailer Konzeptuelles Modell

Das konzeptuelle Modell des Retailers beschreibt die interne Logik des Retailers. Die Nachfrage des Kunden kommt beim Retailer an und wird direkt bearbeitet. Der Retailer überprüft, ob er genug Bestand im Lager hat, um den KundenAuftrag zu erfüllen. Falls Ja, wird die Bestellmenge des Kunden aus dem Lagerbestand des Produktes entnommen. Die Ware wird beim Kunden in der festgelegten Lieferzeit beim Kunden ausgeliefert. Falls er nicht genug Bestand im Lager hat, um die Nachfrage des Kunden zu erfüllen, geht der Kunden leer nach Hause. Alle unerfüllten Kundenaufträge beim Retailer werden als entgangene Aufträge berechnet. Es gibt beim Retailer keine Teillieferung oder Rückstand. Der Retailer arbeitet mit einem maximalen Lagerbestand, einem Startbestand und einem Meldebestand(minimalen Lagerbestand). Sobald der Lagerbestand den Meldebestand erreicht, wird einen Nachschub beim Lieferanten (Wholesaler) angefordert. Die Nachschubmenge wird automatisch als die Differenz zwischen den maximalen Lagerbestand und den Startbestand berechnet: *Nachschubmenge = Maximal Lagerbestand – Startbestand.*

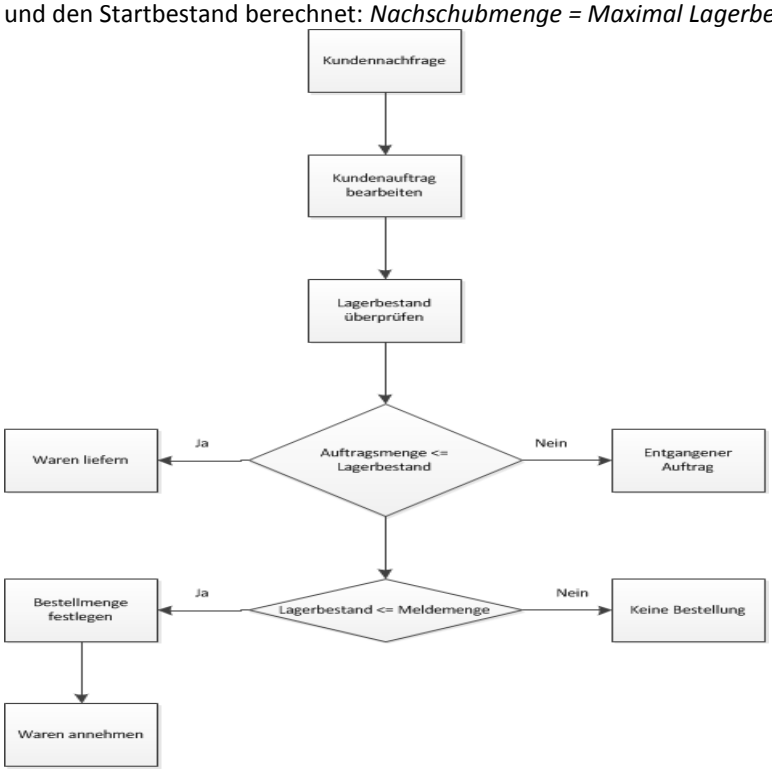


Abbildung 7 :Retailer Conceptual Model

2.4.1.2. Retailer-Modell in Arena

Das konzeptuelle Modell des Retailers wird in einem Arena-Modell umgesetzt. Das implementierte Retailer Simulationsmodell besteht aus Drei Segmenten: Auftragsgenerierung, Auftragsbearbeitung und Bestellprozess(Lagerbestandsführung). (Abbildung 3) Für die Implementierung des Retailer Grundmodelles werden Module aus dem *Basic Process Panel sowie* Module aus dem *Advanced Process Panel benutzt.* Es gibt zwei *Type* von Modulen*: Flowchart modules und Data modules.* Die Fussdiagramm-Module werden in das Modellfenster gezogen und miteinander verbunden. Das Flussdiagramm beschreibt dann die Logik des Retailer-Prozesses. Das Modelfenster besteht aus zwei Bereichen: der obere Teil zeigt die Grafik des Modells, der untere die Daten des Modells.

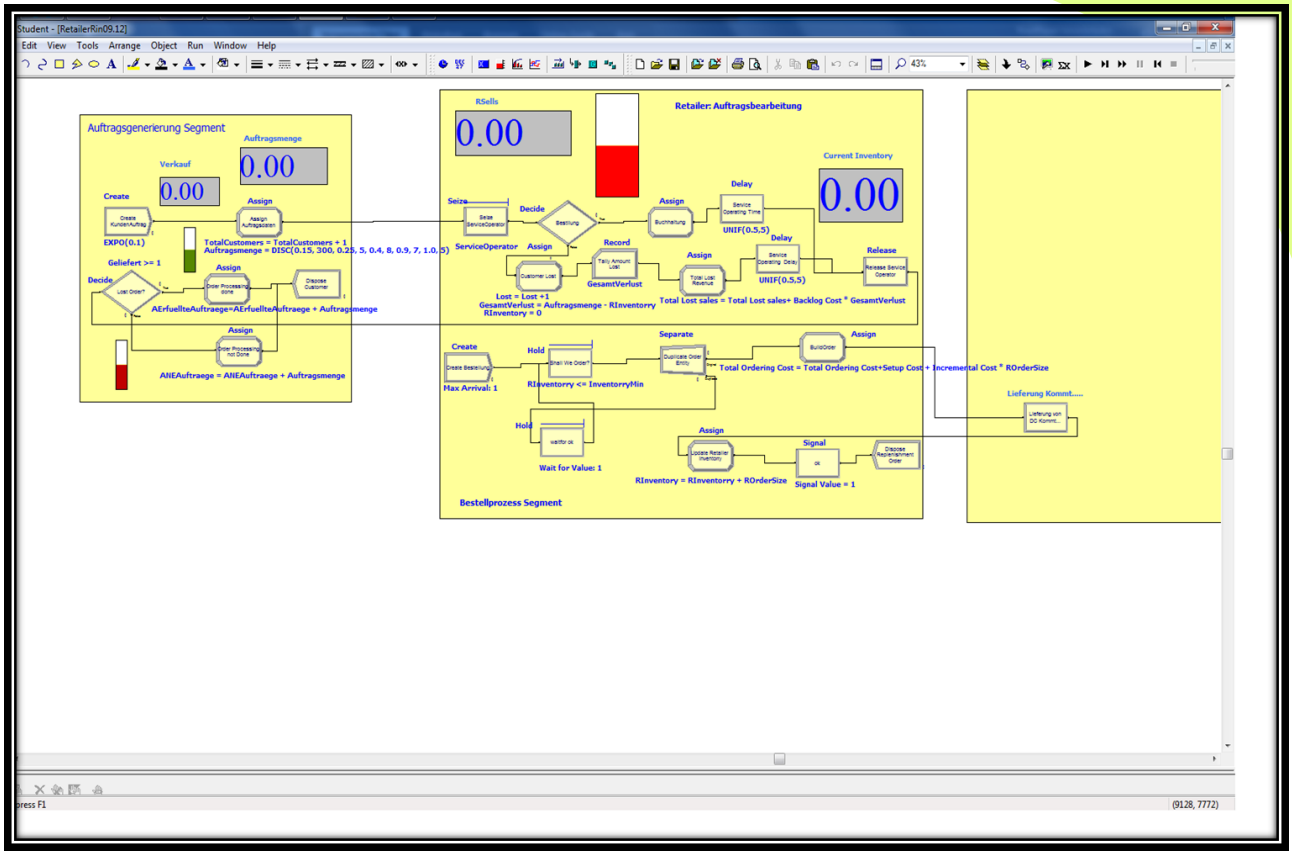


Abbildung 8 :Retailer Arena-Modell

2.4.1.3. Auftragsgenerierung

In diesem Segment wird die Simulation der Retailerkunden durchgeführt. Für die Simulation der Kunden und Kundenaufträge werden in diesem Segment zwei Arena Module verwendet: Create und Assign-Module. Die Retailerkunden kommen zum Retailer mit einer bestimmten Nachfrage. Die Ankunftszeit der Retailerkunden sowie zwischen Ankunftszeit der Retailerkunden wird mit einer bestimmten statistische Verteilung in einem Arena Create-Modul festgelegt. Die Auftragsdaten werden im Assign-Modul mit einer bestimmen statistische Verteilung festgelegt. Die Kunden-Entität wird im Create-Module erzeugt und geht weiter zum Assign-Modul. Im Assign-Module bekommt die Kunden-Entität einem Attribut (Auftragsmenge)zugewiesen. Die Kunden-Entität verlässt das Auftragsgenerierungssegment und betritt das Auftragsbearbeitungssegment.

Zwei Module aus dem *Basic Process Panel wurden benutzt:*

- Per drag & drop ein Create module ins Modellfenster ziehen. Es soll einen Ankunftsprozess nachbilden. (Hier ankommende Kunden in einem Einzelhandel). Im unteren Fenster können die Datenfelder geändert werden oder per Doppelklick auf das Create Modul :
  - Name: „Create KundenAuftrag“
  - Entity Type: „KundenAuftrag“
  - Type :“Random(Expo)“ mit Value:“0.1“. Damit kommen die Kunden mit einer exponentiell verteilten Zwischenankunftszeit an. Im Mittel liegen 6 Minuten zwischen zwei Ankünfte
- Per drag & drop ein *Assign Module* aus dem *Basic Process Panel* ins Modellfenster ziehen und rechts neben dem *Create module* plazieren. Das *Assign Module* in Arena weist Variablen Werte zu. Für die Ankommende Kunden wird eine Variable „Auftragsmenge“ definiert. Per Doppelklick lässt sich das Assign Module editieren:
  - Name :“ Assign Auftragsdaten“

Unter „Assignments“ kann man mit „Add“ Wertzuweisungen machen:

- Type: „Attribute“
- Attribute Name: „Auftragsmenge“
- New Value :“DISC(0.15,3,0.25,5,0.4,8,0.9,7,1.0,5)“. Damit wird die Auftragsmenge für die ankommende Kunden mit einer diskreten Distribution verteilt. Für jede Wahrscheinlichkeit wird eine Zahl als Auftragsmenge zugewiesen.

2.4.1.4. Auftragsbearbeitungssegment

In diesem Segment werden die Aufträge der Retailerkunden bearbeitet. Für diese Bearbeitung werden folgende Arena-Module verwendet:

Aus dem *Basic Process Panel*

- 3 Assign Module
- 1 Decide Module : Repräsentiert einen Entscheidungsprozess in einem Modell in Arena
- 1 Record Module: Sammelt während der Simulation Zwischenergebnisse.
- 1 Dispose Module: modelliert den Abgangsprozess der Kunden.

Aus dem Advanced Process Panel

- 1 Seize Module: Modelliert einen Schalter oder eine Kasse in einem Einzelhandel, von dem die Kunden bedient werden.(Damit müssen ankommende Kunden warten)
- 2 Delay Module: Simuliert die Wartezeit für Kunden, wenn sie bedient werden.
- 1 Release Module: Die verwendete Ressource wird freigegeben, um den nächsten Kunden bedienen zu können.

Die Kunden-Entität betritt zuerst das Seize-Module, welches die Warteschlange an der Kasse mit einem Angestellten(Ressource) simulieren soll. Die Warteschlange arbeitet mit FIFO-Prinzip (First In First Out). Die Kunden-Entität geht weiter zum Decide-Module. Im Decide-Module wird das Attribut *Auftragsmenge* der Kunden-Entität mit dem aktuellen Lagerbestand verglichen: die Bedingung (Auftragsmenge <= Retailer_Inventory) wird überprüft. Die Kunden-Entität hat in diesem Fall zwei mögliche Ausgänge. Wenn die Bedingung erfüllt ist, nimmt die Kunden-Entität den True-Ausgang und betritt das Assign-Module(Buchhaltung), wo die Auftragsmenge von dem aktuellen Lagerbestand abgebucht wird: *Retailer_Inventory = Retailer_Inventory – Auftragsmenge. Die Kunden-Entität geht weiter zum Delay-Module, wo sie die simulierte Zeit für die Bedienung an der Kasse verweilen muss. Danach betritt sie das Release-Module, wo die Ressource (Service Operator) freigegeben wird. Wenn die Bedingung* (Auftragsmenge <= Retailer_Inventory) nicht erfüllt ist, nimmt die Kunden-Entität in diesem Fall den False-Ausgang und betritt das Assign-Module (Lost Customer), wo die entgangene Aufträge beim Retailer berechnet werden. Im Lost-Customer-Module werden folgende Anweisungen definiert:

- Retailer_Inventory = 0, die Variable *Retailer_Inventory* wird auf null gesetzt, da es keine Teillieferung beim Retailer gibt.
- Lost = Lost+ 1, die Variable *Lost* berechnet Anzahl Retailerkunden, welche mit unerfüllten Aufträge beim Retailer ausgegangen sind. Die Variable Lost wird bei jedem durchlauf in diesem Module inkrementiert.
- AmountLost = Auftragsmenge – Retailer_Inventory, das Attribut AmountLost berechnet beim Retailer Anzahl entgangene Aufträge.

Die Kunden-Entität kommt raus aus dem Lost-Customer-Module und geht weiter zum Record-Module(Tally Amount Lost).In diesem Record-Module wird eine statistische Berechnung für AmountLost durchgeführt. Das Ergebnis dieser Berechnung wird automatisch im Simulation-Report im Bereich „Tally“ aufgeführt. Danach betritt die Kunden-Entität verweilt in einem Delay-Module für die festgelegte Bedienzeit und dann betritt sie ein Dispose-Modul, wo sie das Retailer-System verlässt.

2.4.1.5. Bestandsführungssegment

In diesem Segment wird den Lagerbestand beim Retailer regelmässig überwacht. Sobald der festgelegte Meldebestand erreicht wird, soll eine Nachschubanforderung beim Lieferanten(DC) veranlasst. Bei diesem Segment werden folgende Arena-Module verwendet:

Aus dem *Basic Process Panel*

- 2 Assign Module
- 1 Create Module
- 1 Process Module : modelliert den Bestellprozess beim Lieferanten

- 1 Dispose Module: modelliert den Abgangsprozess der Kunden.
- 1 Separate Module: Ströme von Entitäten können zusammen gefasst werden (Batch) und später wieder getrennt werden (Separate)

Aus dem Advanced Process Panel

- 2 Hold Module: Modelliert einen Lager in einem Einzelhandel
- 1 Signal Module: sendet Signal beim Retailer, wenn die Ware beim Retailer angenommen wurde.

Im Create-Module wird eine Nachschub-Entität nach einer festgesetzten Review-Zeit erzeugt. Die Nachschub-Entität geht nach jeder Review-Zeit weiter zum Hold-Module(Shall we Order?). In diesem Hold-Module muss die Nachschub-Entität verweilen bis die Bedingung „ Retailer_Inventory <= InventoryMin“ wahr wird. Die Nachschub-Entität betritt danach ein Separate-Module, wo ihr Informationsgehalt dupliziert wird. Eine Kopie der Nachschub-Entität nimmt den Original-Ausgang und betritt das Assign-Module (Order From DC). In diesem Assign-Module werden folgende Anweisungen festgelegt:

- *ReorderSize = InventoryMax – InventoryMin*, das Attribut ReorderSize bestimmt die Nachschubmenge. Diese Bestellmenge wird beim Lieferanten als Nachschub angefordert.
  Die Nachschub-Entität verlässt danach das Retailer-System und betritt ein Delay-Module. Das Delay-Modul soll das Lieferanten-Module (DC) simulieren. Die Nachschub-Entität muss im Delay-Module für die simulierte Lieferzeit (Lead Time) verweilen. Danach kommt die Nachschub-Entität zurück zum Retailer-System und betritt das Assign-Module(Update Retailer Inventory), wo der Lagerbestand mit der Nachschubmenge aufgefüllt wird: *Retailer_Inventory = Retailer_Inventory + ReorderSize. Danach betritt die Nachschub-Entität ein Signal-Module(Signal Ok), wo ein definiertes Signal ausgelöst wird. Dieses Signal bestätigt den Wareneingang beim Retailer und wird im Hold-Module (Wait for Ok).*

Die zwei Kopie der Nachschub-Entität im Separate-Module nimmt den Copy-Ausgang und betritt das Hold-Module(Wait for Ok), wo sie verweilen muss bis das Signal Ok ausgelöst wird. Das Signal Ok signalisiert den nächsten Bestellprozess beim Retailer. Die Nachschub-Entität kehrt als neue ankommende Nachschub-Entität zum Hold-Module(Shall we Order)zurück.

2.4.2. Retailer-Module in Arena

Das Retailer-Modell in Arena kann mit Hilfe von Arena Template Library in einem Retailer-Module umgewandelt werden. Für die Entwicklung einer Template Bibliothek müsse folgende Schritte durchgeführt werden:

1. Dialog Form Window : User Interface Entwurf
2. Logic Window: Funktionalität und Logik Implementation
3. User View Window: Aussehen der Bibliothek für den Endbenutzer definieren
4. Panel Icon: Module-Symbol für die Bibliothek definieren.
5. Switch Windows: Für die Definition von Variablen für Umschaltfunktionen
   In folgenden Kapiteln wird die Implementierung vom Retailer-Module beschrieben.

2.4.2.1. Dialog Form Window

Im Dialog Form Window wurde die Benutzeroberfläche für das Retailer Module gestaltet. Die Folgende Abbildung zeigt diese Benutzeroberfläche:

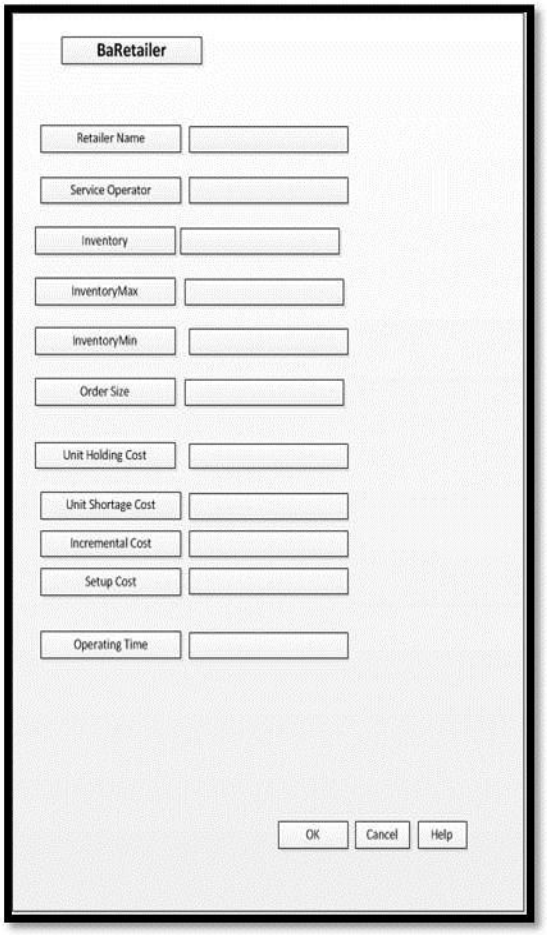


Abbildung 9: Retailer User Dialog Form

Mit dem Dialog Form Window wird das User Interface des Retailer-Modules entworfen. Alle Input Daten, die für die Editierung des Retailer-Modules benötigt werden, sind als Controls definiert. Folgende Tabelle beinhaltet alle verwendete Variablen im Retailer Simulationsmodell:

Variable/Attribute	Beschreibung
Retailer_Inventory	Start Lagerbestand beim Retailer
InventoryMax	Maximaler Lagerbestand beim Retailer
InventoryMin	Minimaler Lagerbestand beim Retailer
Order Size	Auftragsmenge von Retailerkunden
Total Customers	Liefert Anzahl Retailerkunden in einem gegebenen Zeitfenster.
Lost	Liefert Anzahl Retailerkunden mit unerfüllten Aufträge
AmountLost	Liefert Anzahl entgangene Aufträge
ReorderSize	Liefert die Nachschubmenge beim Retailer
Unit Holding Cost	Liefert die Lagerhaltungskosten
Unit Shortage Cost	Liefert die Kosten für Rückstände
Incremental Cost	Liefert die Stückgutkosten
Setup Cost	Fixe Bestellkosten (Fallen bei jeder Bestellung beim Lieferanten an)

Total Ordering Cost	Gesamte Bestellkosten

Ressource	Beschreibung
Service Operator	Angestellter beim Retailer (Verkäufer)
Inventory Evaluator	Angestellter beim Retailer (zuständig für Lagerbestandsführung)

Entitäten	Beschreibung
Kunden	Retailerkunden sind die Entitäten im Retailer-Modell (Auftragsbearbeitung) wandern.
Nachschub	Nachschub-Entität wird im (Bestandsführungssegment) erzeugt und wird als Nachschubmenge Beim Lieferanten(DC) empfangen.

Folgende Controls werden für das User Interface definiert:

Entitäten	Beschreibung
Retailer Inventory	Erlaubt die Eingabe von Startbestand durch den Benutzer
InventoryMax	Erlaubt die Eingabe von maximalem Bestand durch den Benutzer
InventoryMin	Erlaubt die Eingabe von minimalem Bestand durch den Benutzer
Order Size	Erlaubt die Eingabe von Auftragsmenge durch den Benutzer
Unit Holding Cost	Erlaubt die Eingabe von Lagerhaltungskosten
Unit Shortage Cost	Erlaubt die Eingabe von Rückstandkosten
Incremental Cost	Erlaubt die Eingabe von Stückgutkosten
Setup Cost	Erlaubt die Eingabe von Bestellkosten
Service Operaror	Erlaubt die Eingabe von Ressourcen
Retailer Name	Definiert ein Retailer-Module-Objekt.

Diese Controls werden als Operanden im User Interface festgelegt. Jedes Operand-Element wird mit Hilfe vom Operanden-Editor je nach Funktionalität im Retailer-Module, mit den passenden Eigenschaften editiert.

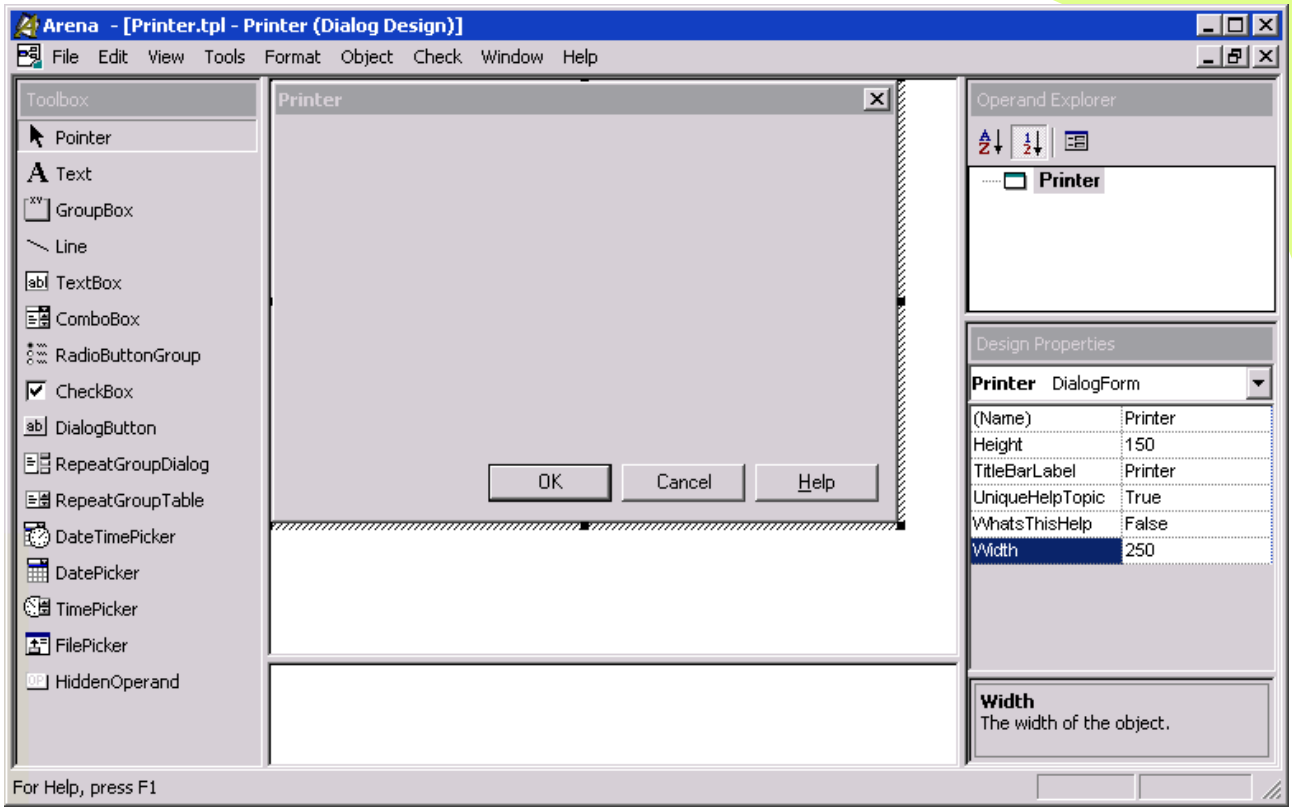


Abbildung 10: Arena Template Dialog Form Window[1]

Der Dialog box Window besteht aus folgenden Komponenten:

- **Dialog Form:** dieses Dialog box Form Layout Fenster ist in der Mitte vom Dialog box Window und dient der Gestaltung der Benutzeroberfläche
- **Toolbox:** beinhaltet die notwendigen Controls für die Gestaltung der Benutzeroberfläche (z.B text boxes, combo box oder dialog box buttons)
- **Operand Explorer:** zeigt die Struktur der Benutzeroberfläche eines Modules (Operanden, Form, Repeat Group)
- **Design Properties:** erlaubt die Bearbeitung von ausgewählten Objekten einer Benutzeroberfläche.

2.4.2.2. Operanden im Retailer-Module Dialog Form

Im Design Properties Grid im Dialog Form Window(Rechts unten) können Operand-Eigenschaften definiert werden. Die Wichtigsten Eigenschaften für Jeden Operanden werden tabellarisch beschrieben.

a) Retailer Inventory Operand

Um den Retailer Inventory Operand im Dialog box Form hinfügen zu können:

1. Per Drag & Drop ein TextBox aus der Toolbox Sektion im Dialog Form Fenster ziehen.
2. Im Design Properties Window, das TextBox wie in folgende Tabelle editieren :
   Die Retailer Inventory soll nur Integer Werte annehmen (Value: Integer). Dieser Operand soll dann dem vom Benutzer eingegebenen Werte der Module-Logik weitergeben(LogicProperties Type: Basic). Die anderen Operanden werden auf die gleiche Weise editiert.

Name	LogicProperties	Value(Data Type)
Retailer Inventory	Basic	Integer

b) InventoryMax Operand

Um den InventoryMin Operand im Dialog box Form hinfügen zu können:

1. Per Drag & Drop ein TextBox aus der Toolbox Sektion im Dialog Form Fenster ziehen.
2. Im Design Properties Window, das TextBox , wie in folgende Tabelle, editieren :

Name	LogicProperties	Value(Data Type)
InventoryMax	Basic	Integer

c) InventoryMin Operand

Name	LogicProperties	Value(Data Type)
InventoryMin	Basic	Integer

d) Order Size Operand

Name	LogicProperties	Value(Data Type)
Order Size	Basic	Expression

e) Unit Holding Cost Operand

Name	LogicProperties	Value(Data Type)
Unit Holding Cost	Basic	Integer

f) Unit Shortage Cost Operand

Name	LogicProperties	Value(Data Type)
Unit Shortage Cost	Basic	Integer

g) Incremental Cost Operand

Name	LogicProperties	Value(Data Type)
Incremental Cost	Basic	Integer

h) Setup Cost Operand

Name	LogicProperties	Value(Data Type)
Setup Cost	Basic	Integer

i) Service Operator Operand

Der Service Operator soll eine Ressource im Retailer Module repräsentieren und wird deswegen als Element definiert. Als Value(Data Tape) wird „SymboleName“ definiert. Damit kann nur einen gültigen Symbole Name vom Benutzer eingegeben werden. Das Folgende Bild zeigt das dialog Fenster für Element-Properties:

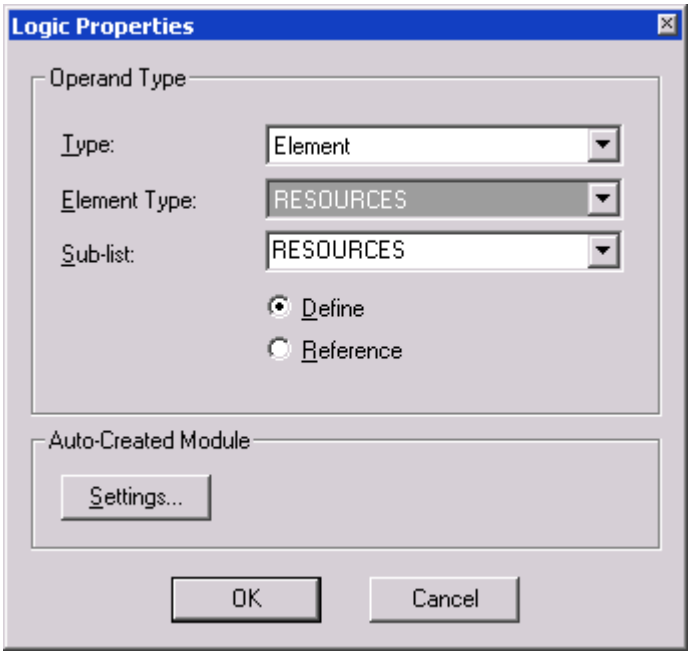


Abbildung 11:LogicProperties Fenster[1]

Name	LogicProperties	Value(Data Type)
Service Operator	Element(Ressource)	SymboleName

j) Retailer Name Operand

Der Retailer Name Operand soll eine Ressource im Retailer Module repräsentieren und wird deswegen als Element definiert. Als Value(Data Tape) wird „SymboleName“ definiert. Damit kann nur einen gültigen Symbole Name vom Benutzer eingegeben werden

Name	LogicProperties	Value(Data Type)
Retailer Name	Element(Ressource)	SymboleName

k) Hidden Operatoren : Entry/Exit Point

Die Hinden Operatoren definieren in einem Module die Verbindungspunkte: Exit Point (Ausgang von Datenfluss aus dem Module) und Entry Point (Eingang von Datenfluss zum Module).Die Hidden Operatoren haben für Value(Data Type) „Label“. Im LogicProperties kann den Type (Entry Point oder Exit Point)für die Hidden Operatoren definiert werden. Für diese Operanden sind zwei Typen vorhanden: „Standard“ für jeden Modulebaustein und „Queue“ nur Queue-Module. Folgende Abbildungen zeigen diese Optionen:

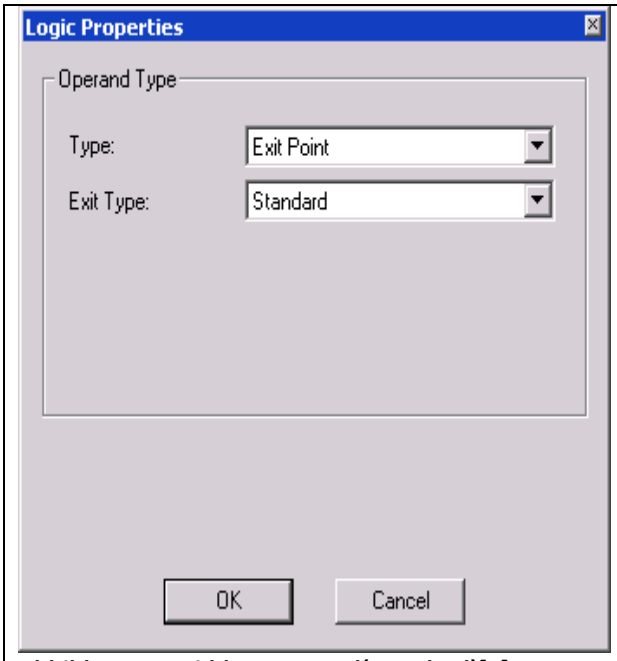


Abbildung 12: Hidden Operand(Standard)[1]

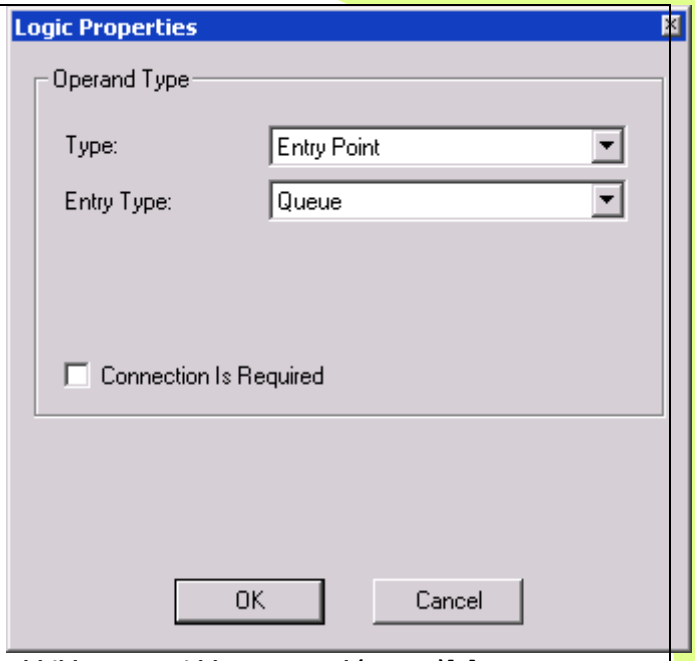


Abbildung 13: Hidden Operand (Queue)[1]

Name	LogicProperties	Value(Data Type)
KundenLabel	Entry Point	Label
DCLabel	Entry Point	Label
NSLabel	Exit Point	Label
DisposeLabel1	Exit Point	Label
DisposeLabel2	Exit Point	Label

Die Operanden werden im Logic Window referenziert, um die Eingabewerte vom Benutzer in der Module-Logik übernommen werden zu können.

2.4.2.3. Retailer Module Logic Window

Im Logic Window wird die ganze Funktionalität vom Retailer-Module implementiert. Die Implementierung der Logik im Logic Window ist ähnlich wie die Implementierung im Arena Model Window. Der einzige Unterschied liegt darin, dass die Run-Funktion im Logik Window nicht verfügbar ist. In diesem Kapitel wird die Implementierung der Retailer-Module Logik beschrieben.

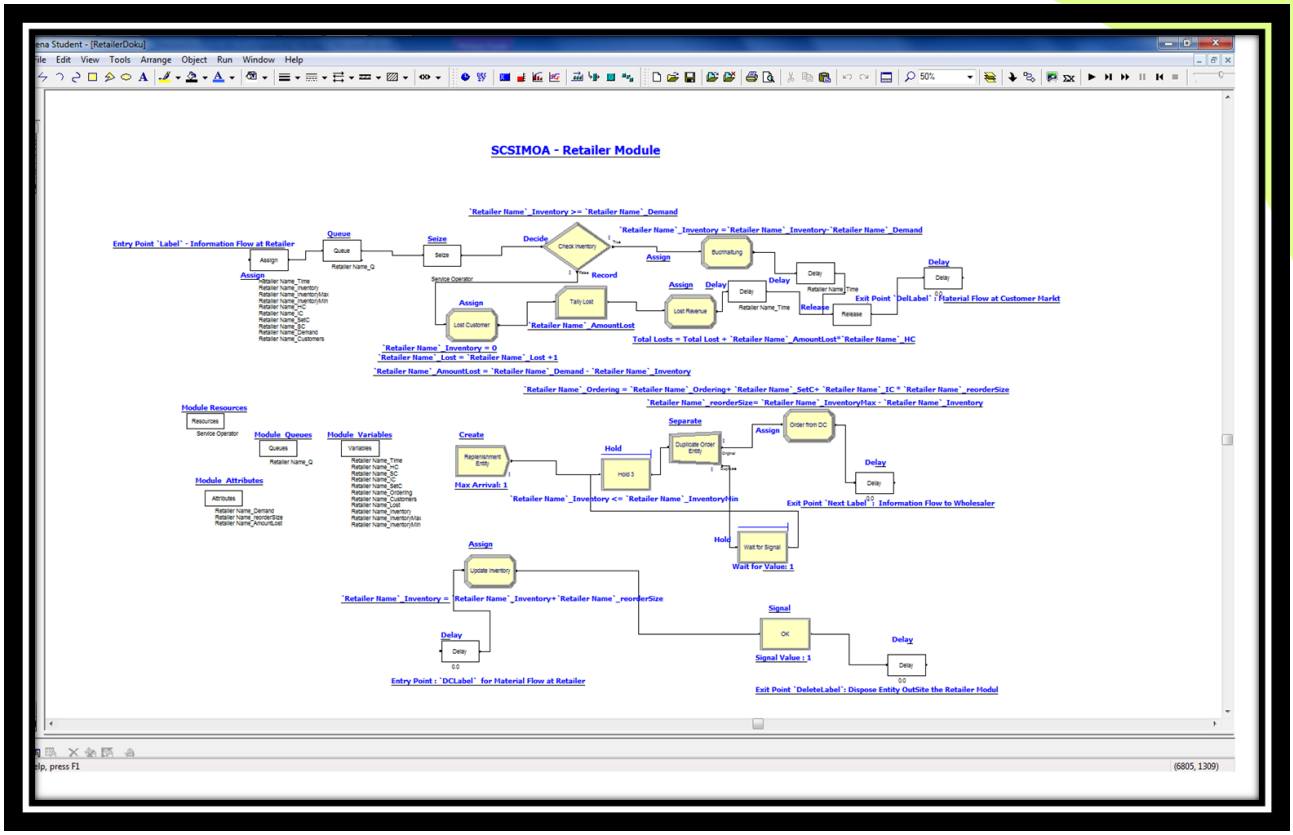


Abbildung 14: Retailer Module-Logik

2.4.2.4. Das Grundmodell erstellen

Das Grundmodell für die Definition der Logik im Retailer Module wird ähnlich wie das Retailer Arena Modell aufgebaut. Folgende Komponenten wurden für den Aufbau des Modelles verwendet:

Aus dem Basic Process Panel

- 1 Decide Module
- 5 Assign Module
- 1 Create Module
- 1 Separate Module
- 1 Record Module
- Kein Dispose Module mehr, aus Designgründen sollen Entitäten ausserhalb vom Module vernichtet werden. Diese Arena Module habe die gleiche Funktionalität wie im Retailer Arena Modell und werden daher nicht erläutert.

Aus dem Advanced Process Panel

- 2 Hold Module
- 1 Signal Module

Aus dem Element Panel

- 1 Resources Module: Definieren von Ressourcen in der Modell-Logik
- 1 Attributes Module: Definieren von Attributen in der Modell-Logik
- 1 Queues Module : Definieren von Queues (Warteschlangen) in der Modell-Logik
- 1 Variables Module: Definieren von Variablen in der Modell Logik

Aus dem Blocks Panel

- 1 Queue Module: Warteschlange für die ankommenden Kunden-Entitäten im Retailer Module. Diese Kunden-Entitäten werden ausserhalb vom Retailer Module generiert wie z.B in einem Create Module, deswegen

werden zuerst in einer Warteschlange (Queue) empfangen bevor sie ihren Weg innerhalb des Retailer Modules weitergehen.

- 1 Assign Module: Definiert als Eingangsbaustein im Retailer Module und referenziert den Entry Point (KundenLabel).Hier werden die Referenzen von den Module Operanden für die Lokalen Variablen und Attributen definiert.
- 1 Seize Module: simuliert die Warteschlange an der Kasse beim Einzelhandel
- 6 Delay Module: 2 Delay Module modellieren die Bedienzeit an der Kasse. Die 4 andere Delay Module sind als Endknoten definiert. 1 Delay Module referenziert den Entry Point (DCLabel) und amtiert aus Eingang für den Materialfluss zum Retailer Module.
- 1 Release Module: Resource freigeben

2.4.2.5. Variables

Das Element (Variables) aus dem Element Panel wird für die Definition von Variablen in der Module Logik. Zwei Arten von Variablen werden definiert: Variable und Interne Variable. Variable wird definiert, um einen entsprechenden Operand im Dialog Form Window zu referenzieren. Interne Variable wird definiert um Statistische Messungen zu durchführen zu können. Alle Operanden werden im Logik-Window referenziert. Ein Operand wird referenziert indem man ihn in Anführungszeichen setzt (`Operand`).
Folgende Variable wurden definiert:

Variable	Operand Referenz	Interne Variable	Beschreibung
Retailer_Inventory	`Retailer Inventory`		Startbestand
Retailer_InventoryMax	`InventoryMax`		Maximalbestand
Retailer_InventoryMin	`InventoryMin`		Meldebestand
Retailer_Holding	`Unit Holding Cost`		Lagerhaltungskosten
Retailer_Shortage	`Unit Shortage Cost`		Kosten für Rückstände
Retailer_Setup	`Setup Cost`		Bestellabwicklungskosten: fallen bei jeder Bestellung automatisch an.
Retailer_Incremental	`Incremental Cost`		Stückgutkosten
Total Customers		Total Customers	Akkumulator :liefert Anzahl Kunden
Lost		Lost	Akkumulator: liefert Anzahl entgangene Aufträge
Total Ordering Cost		Total Ordering Cost	Akkumulator: liefert die gesamte Bestellkosten

2.4.2.6. Attributes

Der Element-Baustein „Attributes“ wird verwendet um interne Attribute zu definieren. Alle Attribute werden im Element-„Attributes“ gespeichert. Folgende Attribute wurden definiert:

Attribute	Beschreibung	Operand Referenz
ReorderSize	Definiert die Nachschubmenge	
OrderSize	Definiert die Auftragsmenge	`Order Size`
AmountLost	Definiert Entgangene Aufträge	

2.4.2.7. Queues

Das Element-„Queues“ wird verwendet um Queue in der Module-Logik zu definieren. Alle Queues werden im Element-„Queues“ gespeichert. Folgende Queues wurden definiert:

Queue	Beschreibung	Operand Referenz
`Retaile Name`-Queue	Warteschlange für die ankommenden Entitäten ins Retailer Module	`Retailer Name`
`Service Operator`-Queue	Warteschlange an der Kasse	`Service Operator`

2.4.2.8. Operand Referenz

Das Assign Module in Arena wird bekanntlich für Wertzuweisungen verwendet. Aus dem Arena Blocks Panel wird ein Assign Module per Drag &Drop im Module Logic Window platziert. Per Doppelklick auf Assign Block lässt sich das Assign Block editieren. Die Folgende Abbildung zeigt das Assign Bock mit Operand Referenz:

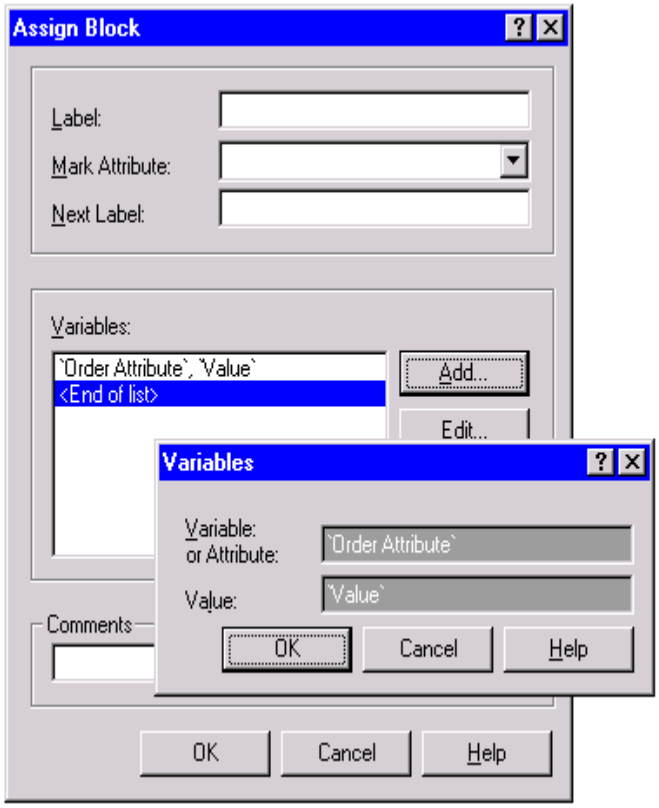


Abbildung 15: Operand Referenz[1]

2.4.2.9. Entry/ Exit Point Referenz

Die Entry und Exit Points Operanden werden für den Information-/Materialfluss im Retailer-Module definiert. Um diese Operanden im Logic Window zu referenzieren müssen Block-Elemente statt Basic Process Element verwendet werden. Für den Zugriff im Retailer Module auf die Entry/Exit Point werden Delay Block mit einem Delay Zeit von 0.00 verwendet. Wenn das Delay Block als Entry Point dienen soll, wird die Referenz im Label Feld gemacht. Wenn das Delay Block als Exit Point dienen soll, wird die Referenz im Next Label Feld gemacht. Die Folgende Abbildung zeigt die Referenz auf einem Entry Point in einem Delay Block:

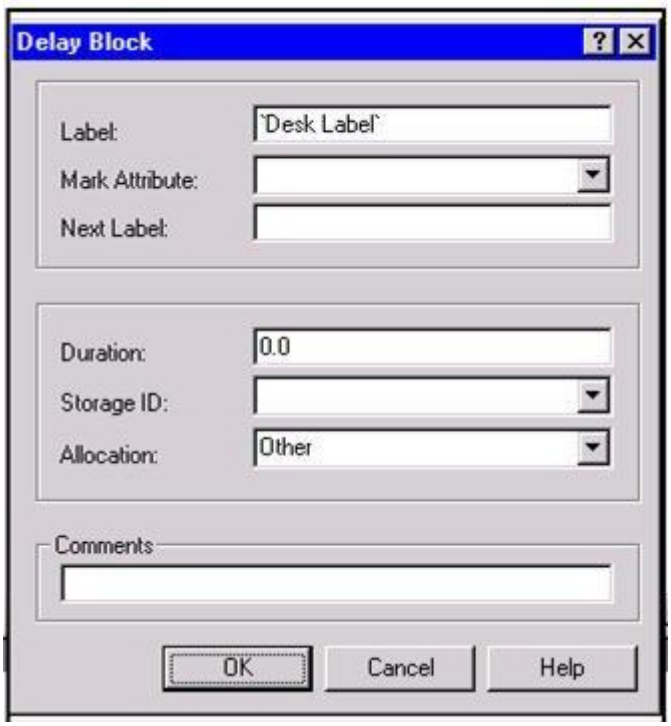


Abbildung 16: Entry/Exit Point Referenz[1]

2.4.2.10. User View Dialog

Das User View Window im Retailer Module zeigt wie das Retailer Module in einem Simulationsmodell aussehen soll. Damit kann man das passende Aussehen der Applikation für den Endbenutzer gestalten. Die folgende Abbildung zeigt das Aussehen des Retailer Modules in einem Arena Modell.
Das User View beim Wholesaler Module wird angezeigt, wenn eine Module-Instanz in einem Arena-Modell verwendet wird.

Das User View zeigt zwei Entry Point Points (Links) und 1 Exit Points(Rechts) an.

Entry Points:

- Für den Informationfluss vom Kunden zum Retailer wird ein Entry Point benötigt.
- Für den Materialfluss von Lieferanten(Wholesaler) zum Retailer wird ein Entry Point benötigt.

Exit Point:

- Für den Informationsfluss vom Retailer zum Wholesaler wird ein Exit Point benötigt.

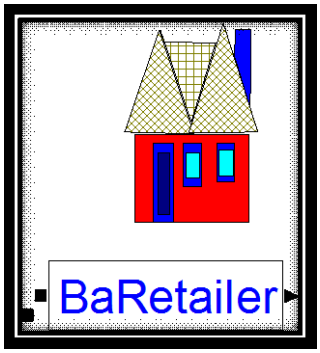


Abbildung 17: Retailer Module User View

2.4.2.11. Panel Icon

Das Panel Icon zeigt wie das Retailer Module im Arena Template Panel aussehen soll. Das entwickelte Retailer Module wird automatisch in einem Template Library File(.tpl) gespeichert. Daraus wird einem Template Panel Object File (.tpo) automatisch in Arena generiert. Das (.tpo)-File wird dann im Arena Project Bar integriert. Dies erfolgt mit dem Befehl in Arena >File> Template Panel > Attach. Um wieder das Module aus dem Project Bar zu entfernen, einfach Retailer Module auswählen und mit dem Befehl > File > Template Panel > Detach, das Module entfernen. Die folgende Abbildung zeigt Symbole für das Retailer Module im Projekt Bar.

Abbildung 18: Retailer-Module Panel Icon

2.4.2.12. Retailer-Module Testen

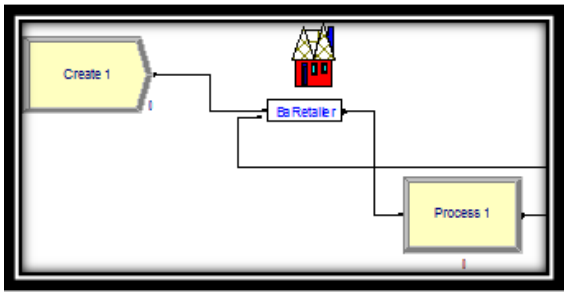


Abbildung 19: Retailer-Module Test Modell

Um das Retailer Module zu testen, muss man zuerst das Retailer Module ins Arena Template Panel importieren. Das importierte Retailer Module wird dann im Arena Project Bar ersichtlich und kann wie jedes Arena Module in einem Arena Modell verwendet werden. Das Retailer Module Test-Modell besteht aus folgenden Modulen:

- 1 Create Module aus dem Basic Process Modell: das Create1 Module generiert die Kunden-Entitäten für das Retailer Module. Über einem Entry Point (KundenLabel) wird das Create1 Module mit dem Retailer Module verbunden.
- 1 Process Module aus dem Basic Process Panel: das Process1 Module simuliert die Funktion eines Lieferanten (Wholesaler)für das Retailer Module. Die Kommunikation zwischen Process1 Module und Retailer Module geschieht in zwei Richtungen und über zwei Verbindungspunkte. Der Exit Point (Next Label1) vom Retailer Module wird mit dem Entry Point vom Process1 Module verbunden: dies simuliert den Informationsfluss von Retailer zum Wholesaler. Der Exit Point vom Process1 Module wird dem Entry Point(DCLabel) vom Retailer Module verbunden: dies simuliert den Materialfluss vom Wholesaler zum Retailer.
- Das Retailer Module: simuliert das Retailer –Modell in einem Supply Chain.

Das Ergebnis der Testsimulation vom Retailer Module wird im Kapitel 11 (Performance Messungen) erläutert.

2.4.3. Wholesaler Module

Der Wholesaler ist der Vorgänger vom Retailer im konzipierten Supply chain Modell. Der Retailer kommuniziert mit dem Wholesaler über zwei Kanäle: Informationsfluss und Materialfluss. Wenn beim Retailer einen Bedarf an Nachschub festgestellt wird, erfolgt die Nachschubforderung beim Wholesaler über den Informationsfluss-Kanal. Der Wholesaler erhält die Nachfrage des Retailers und bearbeitet sie. Wenn der Wholesaler mit seinem aktuellen Lagebestand die Bestellmenge erfüllen kann, wird dann die Bestellmenge vom Lagerbestand direkt entnommen. Die Ware wird dann beim Retailer nach einer bestimmten Lieferzeit(Lead Time) ausgeliefert. Falls die Bestellmenge nicht vom aktuellen Lagerbestand nicht erfüllt werden kann, so wird die Bearbeitung der Nachfrage verzögert. Einen Nachschub wird vom Wholesaler bei seinem Lieferanten (Factory:sein Vorgänger in der Lieferkette) angefordert. Sobald die Ware beim Wholesaler ausgeliefert wird, wird automatisch die pendente Nachfrage des Retailers bearbeitet. Er gibt beim Wholesaler keine Teillieferung. Die komplette Bestellmenge wird von der Nachschubmenge entnommen und die Ware mit einer Verzögerte Lieferzeit (Wholesaler Lead Time = Wholesaler Lead Time + Factory Lead Time) beim Retailer ausgeliefert.

2.4.3.1. Wholesaler Arena-Modell

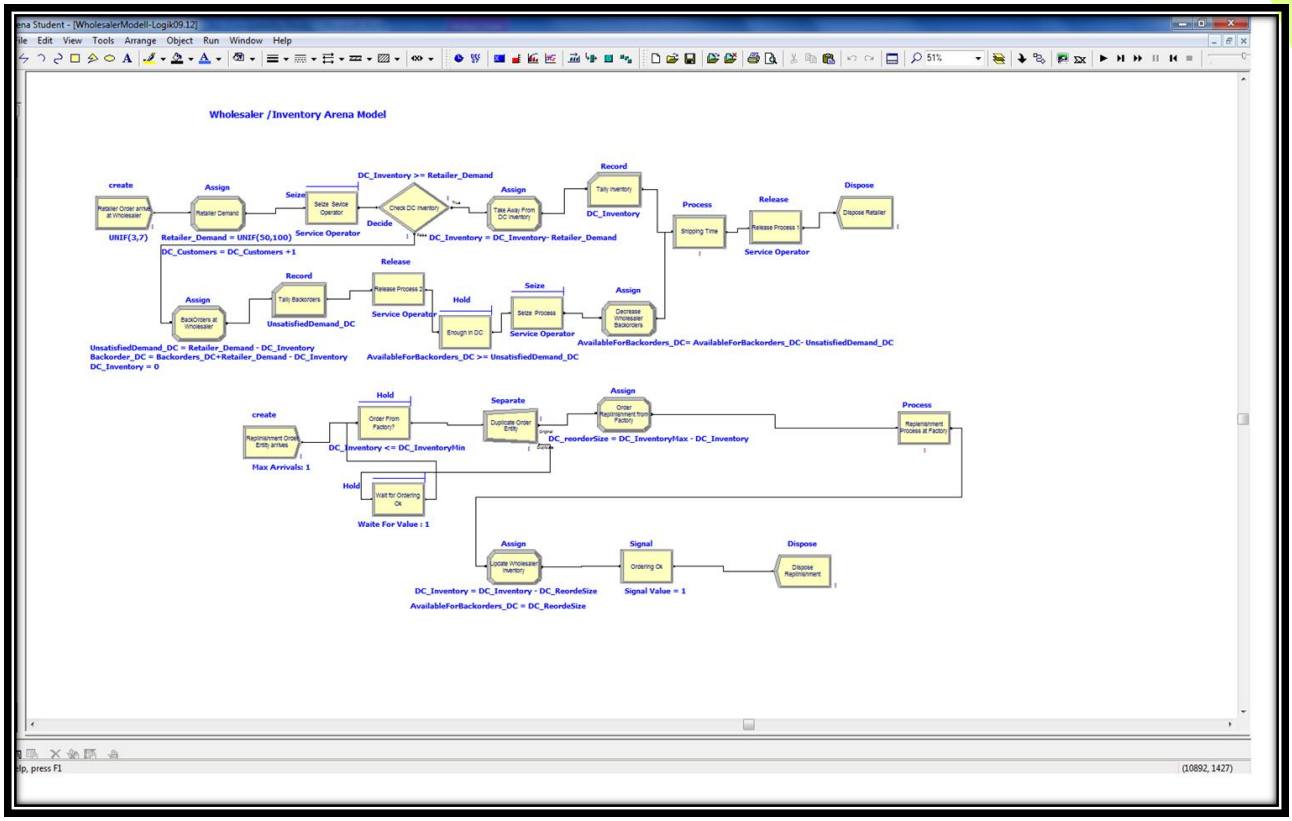


Abbildung 20: Wholesaler Arena Modell

Das Wholesaler Arena-Modell implementiert die Logik beim Wholesaler. Das Simulationsmodell besteht auch aus drei Segmenten: Auftragsgenerierung, Auftragsbearbeitung und Lagerbestandführung. Die Implementierung der Logik ist ähnlich wie die beim Retailer. Die zwei Implementierungen unterscheiden sich nur im Auftragsbearbeitungssegment. In diesem Segment werden keine entgangene Aufträge registriert sondern Rückstände. Diese Rückstände werden in einem Puffer gesammelt bis genug Bestand wieder beim Wholesaler vorhanden ist. Die Bearbeitung der Rückstände erfolgt danach nach dem FCFS-Prinzip (First Come Fisrt Served).
Der Aufbau des Wholesaler Grundmodells wird hier nicht in Detail erläutert. Für eine nähere Erläuterung des Grundmodells wird auf das Retailer Grundmodell verwiesen.

2.4.3.2. Wholesaler Module-Logik

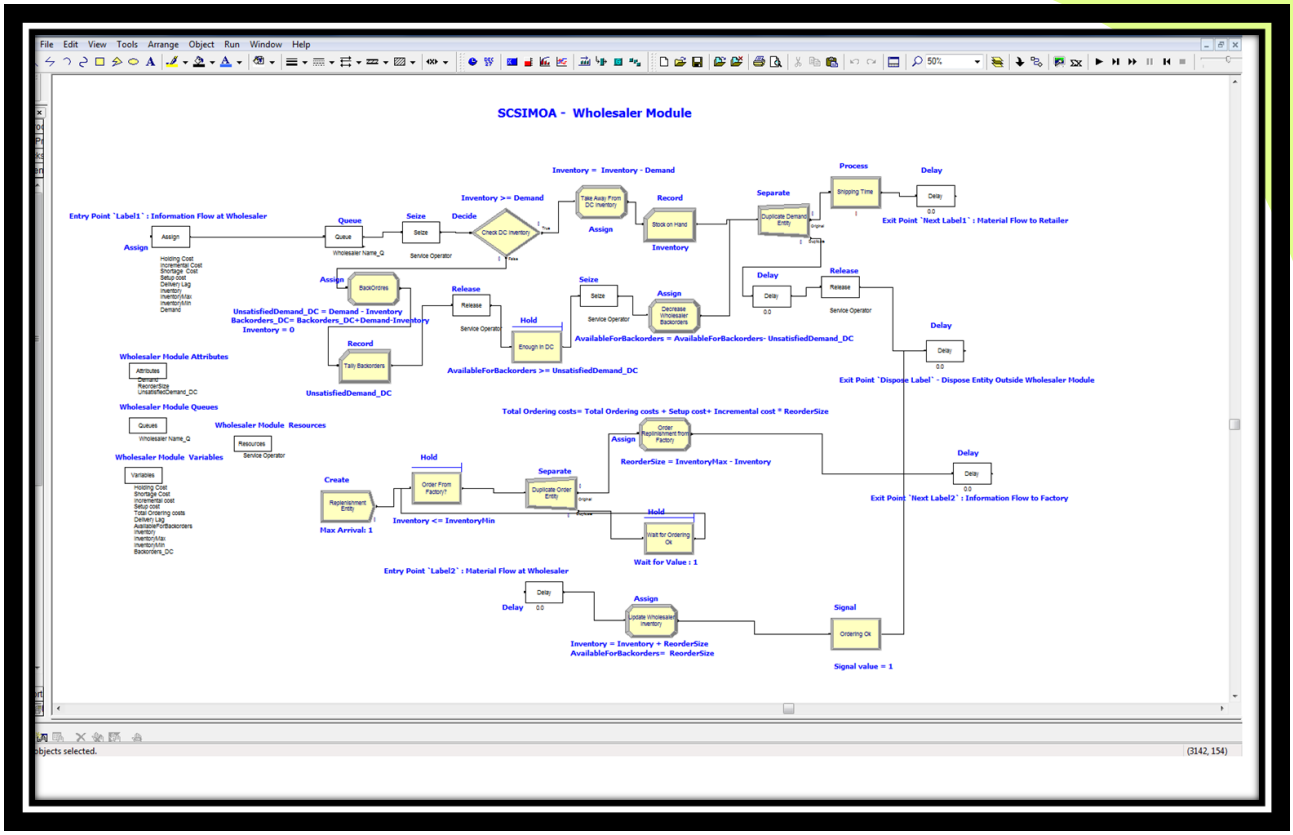


Abbildung 21: Wholesaler Module-Logik

Die Wholesaler Module-Logik ist eigentlich die gleiche Logik wie die Arena-Modell-Logik. Die zwei Implementierungen unterscheiden sich nur bei der Verwendung von Blocks und Elements-Bausteine in der Wholesaler-Module-Logik. Der Einsatz dieser Bausteine für Implementierung der Module-Logik ist gleich wie bei der Retailer-Module-Logik. Von daher werden hier die einzelnen Bausteine nicht erläutert. (Queues,Vaiables und Attributes).

Der Aufbau des Wholesaler Module Grundmodells wird hier nicht in Detail erläutert. Für eine nähere Erläuterung des Grundmodells wird auf das Retailer Module Grundmodell verwiesen.

2.4.3.3. Wholesaler Module User Dialog Form

Im Dialog Form Window wurde die Benutzeroberfläche für das Wholesaler Module gestaltet. Die Gestaltung der Benutzeroberfläche wird hier nicht erläutert und es wird auf das Retailer Module für die Erläuterung verwiesen. Die Folgende Abbildung zeigt diese Benutzeroberfläche:

Abbildung 22: Wholesaler User Dialog Form

2.4.3.4. Wholesaler Module User View

Das User View beim Wholesaler Module wird angezeigt, wenn eine Module-Instanz in einem Arena-Modell verwendet wird.

Das User View zeigt zwei Entry Point Points (Links) und drei Exit Points(Rechts) an.

Entry Points:

- Für den Informationfluss vom Retailer zum Wholesaler wird ein Entry Point benötigt.
- Für den Materialfluss von Lieferanten(Factory) zum Wholesaler wird ein Entry Point benötigt.

Exit Points:

- Für den Materialfluss von Wholesaler zum Retailer wird ein Exit Point benötigt.
- Für den Informationsfluss vom Wholesaler zum Lieferanten(Factory) wird ein Exit Point benötigt.
- Für die Vernichtung von Entitäten ausserhalb des Wholesaler Modules wird auch ein Exit Point benötigt.

Abbildung 23: Wholesaler Module User View

2.4.3.5. Wholesaler Module Panel Icon

Das Panel Icon beim Wholesaler Module repräsentiert das Module-Symbol im ProjectBar, wenn das Wholesaler Module ins Arena-Tool importiert(Arena>File >Template Panel>Attach) wird.

Abbildung 24: Wholesaler Module Panel Icon

2.4.3.6. Wholesaler Module: Arena Test-Modell

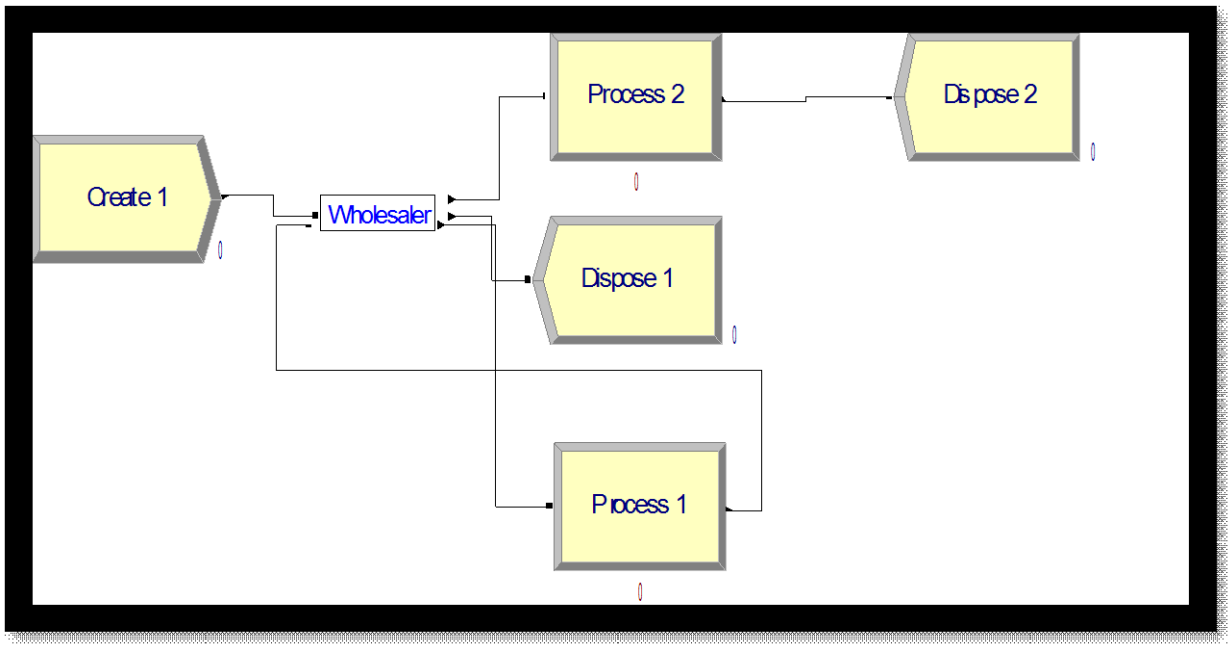


Abbildung 25: Wholesaler Module Arena Test Modell

Das Arena Test-Modell vom Wholesaler Module besteht aus sechs Module: ein Create-Module, zwei Process-Module, zwei Dispose-Module und das Wholesaler Module. Das Create-Module (Create 1) generiert die Retailer-Aufträge für das Wholesaler-Modul. Diese Aufträge betreten das Wholesaler Modul über einen Entry Point, der für den Informationsfluss bestimmt ist. Das Wholesaler Modul bearbeitet die Retailer-Aufträge und liefert die Ware beim Retailer aus. Diese Auslieferung erfolgt über einen Exit Point zum Process-Modul (Prozess 2), der für den Materialfluss zum Retailer bestimmt ist. Der Wholesaler fordert bei Bedarf Nachschub beim Lieferanten(Factory) an. Die Nachschubanforderung erfolgt über einen Exit Point zum Process-Modul(Process 1), der für den Informationsfluss zum Lieferanten (Factory) bestimmt ist. Das Process-Modul(Process 1) simuliert das Factory-Modul und liefert die Nachschubmenge beim Wholesaler aus. Diese Auslieferung beim Wholesaler erfolgt über einen Entry Point, der für den Materialfluss zum Wholesaler bestimmt ist. Im Dispose-Modul erfolgt nur das Vernichten von den Entitäten. Alle Entitäten, die vernichtet werden sollen, werden ausserhalb des Wholesaler Modules vernichtet. Deswegen werden diese Entitäten auch zum Dispose-Modul(Dispose 1) über einen Exit Point geschickt.

2.4.4. Factory Module

Das Factory Module implementiert die Logik in einem Factory in Supply Chain. Die Fabrik oder Factory ist der Vorgänger vom Wholesaler in der Lieferkette. Die beiden Komponenten kommunizieren über den Informationsfluss Kanal und auch über den Materialfluss Kanal. Das Factory-Modul stellt Produkte für den Wholesaler her. Bei Bedarf wird einen Nachschub an Komponenten für die Produktion beim Lieferanten (Supplier) angefordert. Die Nachschubsstrategie folgt das (r, R) Inventory- Control-Modell. Bei dieser Strategie soll einen Nachschub angefordert werden, sobald der Meldebestand erreicht wird.
Das Factory-Modul erhält Wholesaler-nachfragen und bearbeitet sie. Wenn genug Bestand für die Erfüllung der Nachfrage vorhanden ist, wird die Bestellmenge vom aktuellen Lagerbestand entnommen und die Ware beim Wholesaler nach einer bestimmten Lieferzeit ausgeliefert. Wenn nicht genug Bestand vorhanden ist, wird die Auslieferung der Ware verzögert bis wieder genug Bestand beim Factory vorhanden ist. Es gibt auch beim Factory keine Teillieferung.

2.4.4.1. Factory Arena-Modell

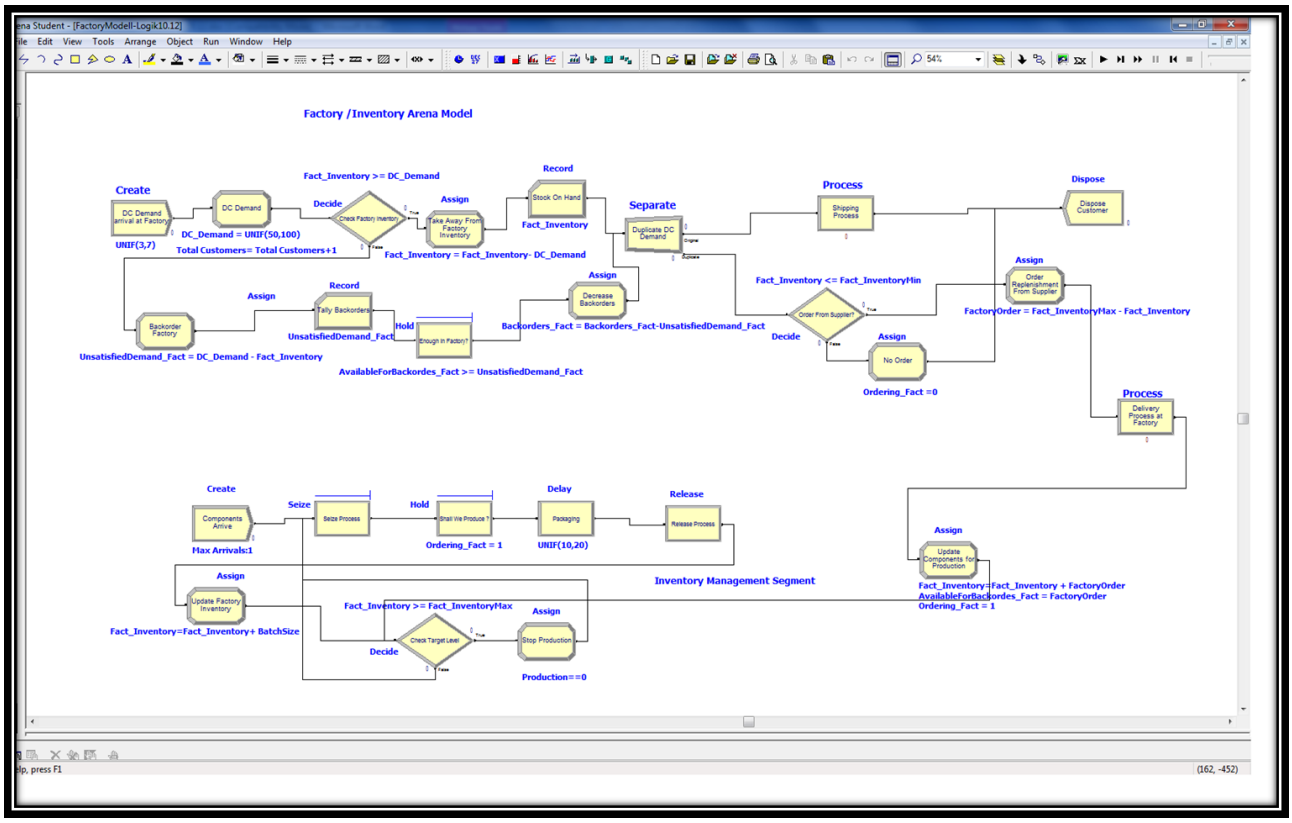


Abbildung 26: Factory Arena Modell

Das Factory –Modell besteht aus zwei wesentliche Segmenten: Demand Management und Inventory Management.

Inventory Management Segment

Das Inventory Management Segment simuliert die Produktion in der Fabrik und kontrolliert das Lagerbestand bei der Fabrik. Das Inventory Management Segment wird mit folgendem Baustein aus dem Basic Process Panel aufgebaut:

- 1 Create Module: liefert die Komponente für die Produktion aus dem Lager
- 1 Seize Module: Stellt die gelieferten Komponenten für die Produktion bereit.
- 1 Hold Module: Gibt das Signal für die Produktion, sobald die Bedingung „Ordering_Fact ==1“ wahr wird.
- 1 Delay Module: simuliert die benötigte Zeit für die Bereitstellung der Produkte.

- 1 Release Module: Für die Freigabe der Ressourcen(Maschine) nachdem der Verarbeitungsprozess fertig ist.
- 1 Assign Module (Update Inventory): Lager bis zum Maximalbestand auffüllen
- 1 Decide Module (Check Target Level): Lagerbestand kontrollieren, damit der Maximalbestand nicht überschritten wird.
- 1 Assign Module (Stopp Production): Signalisiert das Stoppen für die Produktion(„Ordering_Fact ==0“), sobald das Maximalbestand erreicht wird.

Demand Management Segment

Das Demand Management Segment simuliert den Auftragsbearbeitungsprozess bei der Fabrik. Das Segment ist mit folgenden Arena Bausteinen aufgebaut:

Aus dem Basic Process Panel

- Create Module (**DC Demand Arrives at Factory**): Dieses Create Module generiert die Wholesaler-Entitäten und gibt diese weiter zum Assign Module (DC Demand).
- Assign Module (**DC Demand**):In diesem Assign Module werden folgende Wertzuweisungen durchgeführt:
    - Attribute DC_Demand = UNIF(50,100): definiert die Auftragsmenge als eine Uniform Normal Verteilung. Die Auftragsmenge wird zufällig zwischen 50 und 100 ausgewählt und dem Attribute DC_Demand zugewiesen. Jede Wholesaler-Entität bekommt beim Betreten dieses Assign Modules Kopie von DC_Demand zugewiesen bevor sie zum nächsten Decide Module (Check Factory Inventory) geschickt wird.
    - Variable Total Customers = Total Customers + 1: Diese Variable ist ein Akkumulator und soll die Factory Kunden zählen.
- Decide Module (**Check Factory Inventory**): In diesem Module wird überprüft ob die Factory genug Bestand für Erfüllung des Kundenauftrags hat: **Fact_Inventory >= DC_Demand**? Wenn die Bedingung wahr ist, verlässt die Wholesaler Entität über den True-Ausgang das Decide Module **(Check Factory Inventory**) und betritt das Assign Module(**Take Away From Factory Inventory**), wo die Variable Fact_Inventory um DC_Demand dekrementiert wird. Wenn die Bedingung nicht wahr ist, verlässt die Wholesaler Entität über den False-Ausgang das Decide Module(**Check Factory Inventory**)und betritt das Assign Module (**Backorders Factory**), wo die Rückstände registriert werden.
- Assign Module (**Take Away From Factory Inventory**): Auftragsmenge aus dem aktuellen Factory Lagerbestand entnehmen: **Fact_Inventory = Fact_Inventory – DC_Demand**
- Assign Module **(Backorders Factory)**: folgende Anweisung werden in diesem Module gemacht:
    - UnsatisfiedPortionDemand_Fact = DC_Demand – Fact_Inventory: Die Rückstandsmenge wird in die Variable „UnsatisfiedPortionDemand_Fact“ registriert.
    - Fact_Inventory = 0: Der aktuelle Lagerbestand wird auf null gesetzt.
      Die Wholesaler Entität verlässt dieses Modul und betritt das Recod Module (**Tally Backorders**).
- Recod Module (**Tally Backorders**): die Expression „UnsatisfiedPortionDemand“ wird für die Tally-Statistik ausgewertet und die zwischen Ergebnisse werde gesammelt. Die Endergebnisse werden im Simulationsreport in der Tally Sektion erscheinen. Die Wholesaler Entität betritt danach das Hold Module (Enough in Factory?).
- Hold Module (**Enough in Factory?**): Die Rückstände werden erst kompensiert, wenn wieder genügend Bestand im Lager vorhanden ist. Die Wholesaler Entität wird in der Warteschlange im Hole Module solange gehalten bis ausreichend Bestand vorhanden ist, d.h dann bist die Bedingung (**AvailableForBackorders_Fact >= UnsatisfiedPortionDemand_Fact**) wahr wird. Danach darf die Wholesaler Entität das Hold Module verlassen und das Assign Module(**Decrease Backorders**) betreten. Es können mehrere Entitäten in der Schlange sein, aber das Hold Module in Arena arbeitet nach dem FCFS-Prinzip(First Come First Served).
- Assign Module (**Decrease Backorders**): Die Rückstände werden kompensiert und den KundenAuftrag erfüllt : **Backorders_Fact = Backorders_Fact - UnsatisfiedPortionDemand_Fact**
- **Record Module (Stock On Hand):** Sammelt die Statistik-Werte über die Variable Fact_Inventory. Das Endergebnis wird im Simulationsreport in der Tally Sektion erscheinen.
- **Separate Module(Duplicate DC Demand):**An diesem Punkt ist die Auftragsabwicklung fertig und die Waren wird beim Wholesaler ausgeliefert. Die Information über die Wholesaler Entität soll dupliziert werden. Eine

Kopie der Wholesaler Entität geht zum Prozess Module (**Shipping Process**) und die andere zum Decide Module (**Order From Supplier**).

- **Process Module(Shipping Process):** Die Wholesaler Entität muss die festgelegt Lieferzeit zwischen Wholesaler und Factory in diesem Module verweilen. Damit wird den Warentransport zum Wholesaler simuliert.
- **Decide Module(Order From Supplier):** in diesem Module wird die Lagerbestandführung simuliert. Es wird immer überprüft, ob der Lagerbestand den Meldebestand erreicht hat. Also wenn die Bedingung **Fact_Inventory <= Fact_InventoryMin** erfüllt ist, soll einen Nachschub an Komponenten für die Produktion beim Lieferanten(Supplier) angefordert werden(Wholesaler Entität geht weiter zum Assign Module**(Order Replenishment From Supplier).**
  Wenn die Bedingung nicht erfüllt ist, wird keine Bestellung abgewickelt(Wholesaler Entität geht weiter zum Assign Module(**No Order**).
- **Decide Module(Order Replenishment From Supplier):** die Nachschubmenge wird berechnet und die Bestellung zum Lieferanten geschickt(Prozess Module(**Delivery Process at Factory**)). Die folgende Anweisung definiert die Nachschubmenge:
  FactOrder = FactOrderMax – Fact_Inventory.
- Process Module (**Delivery Process at Factory**): simuliert den Lieferanten (Supplier). Die Wholesaler Entität geht zum Supplier mit dem Informationsfluss. Und Kommt zum Wholesaler mit dem Materialfluss zurück. Sie betritt dann das Assign Module (**Update Components for Production).**
- Assign Module (**Update Components for Production):** Dieses Assign Module simuliert den Wareneingang beim Wholesaler. Die Komponenten werden beim Wholesaler ausgeliefert und den Lager aufgefüllt. Folgende Operationen werden durchgeführt:
  - Fact_Inventory = Fact_Inventory + FactOrder: Lagerbestand mit Nachschubmenge auffüllen.
  - AvailableForBackOrder = FactOrder: signalisieren den Wareneingang, damit die Rückstände sofort kompensiert werden.
  - Ordering_Fact = 1: signalieren den Wareneingang, damit die Produktion starten kann.
    Die Wholesaler Entität geht dann weiter zum Inventory Management Segment und betritt das Decide Module ((Check Target Level))) und beendet ihren Weg.

2.4.4.2. Factory Module Logik

Die implementierte Logik im Arena Factory Modell wird direkt für die Factory Module Logik übernommen. Deswegen wird der Modellaufbau nicht erläutert. Im Factory Module Logik wurden auch Bausteine aus Blocks Panel und Bausteine aus dem Elements Panel verwendet. Diese Bausteine erfüllen die gleiche Aufgabe wie beim Retailer Module(Kapitel 4). Deswegen verzichten wir auf die Erklärung ihrer Funktion und es einfach für die nähere Details auf das Retailer Module verwiesen.

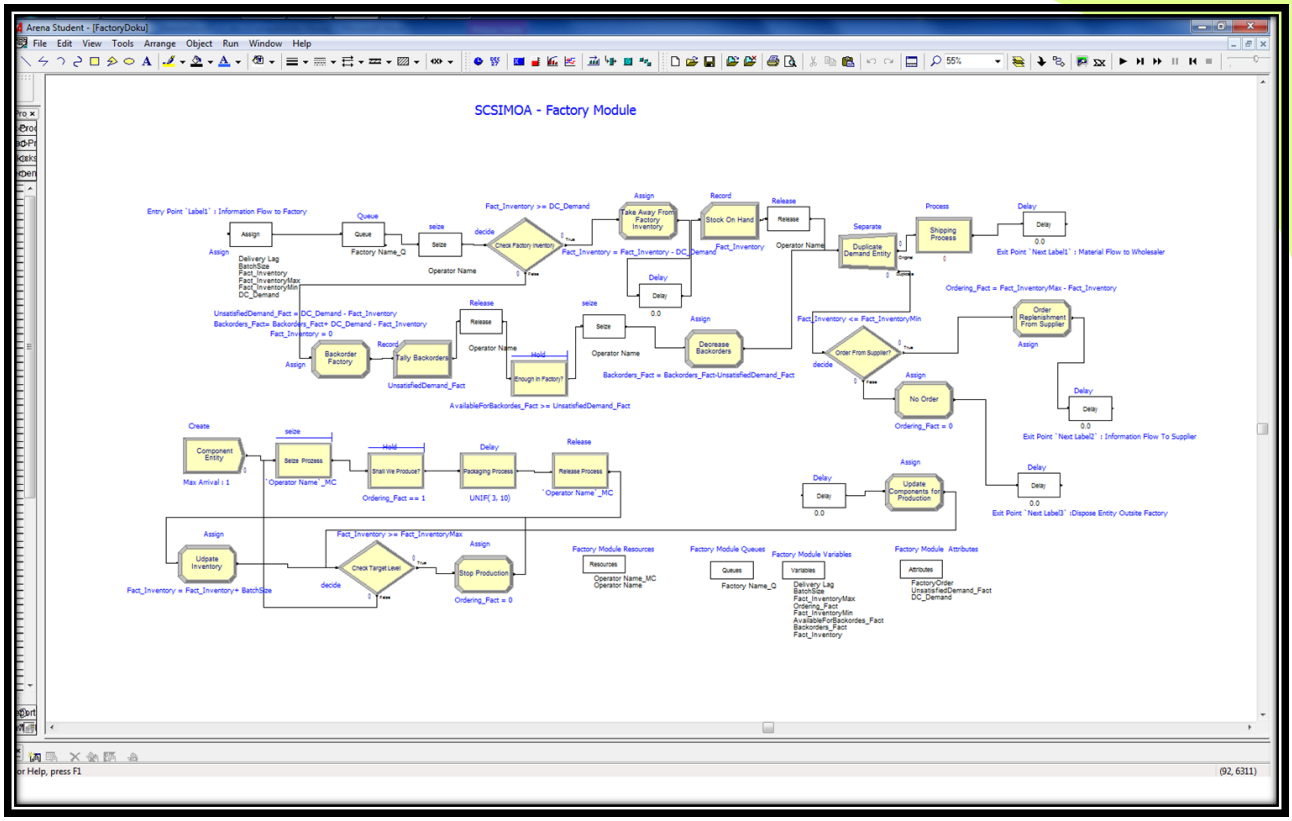


Abbildung 27: Factory Module Logik

2.4.4.3. Factory Modul User Dialog Form

Für die Erklärung der Modellbildung wird auf das Retailer Module(Kapitel 4) verwiesen.

Abbildung 28: Factory User Dialog Form

2.4.4.4. Factory Modul User View

Das User View zeigt zwei Entry Point Points (Links) und drei Exit Points(Rechts)an.

Entry Points:

- Für den Informationfluss vom Wholesaler zum Factory wird ein Entry Point benötigt.
- Für den Materialfluss vom Lieferanten(Supplier) zum Wholesaler wird ein Entry Point benötigt.

Exit Points:

- Für den Materialfluss vom Factory zum Wholesaler wird ein Exit Point benötigt.
- Für den Informationsfluss vom Factory zum Lieferanten(Supplier) wird ein Exit Point benötigt.
- Für die Vernichtung von Entitäten ausserhalb des Factory Modules wird auch ein Exit Point benötigt.

Abbildung 29: Factory Module User View

2.4.4.5. Factory Modul Panel Icon

Das Panel Icon beim Factory Module repräsentiert das Module-Symbol im ProjectBar, wenn das Factory Module ins Arena-Tool importiert(Arena>File >Template Panel>Attach) wird.

Abbildung 30: Factory Module Panel Icon

2.4.4.6. Factory Modul Arena Test-Modell

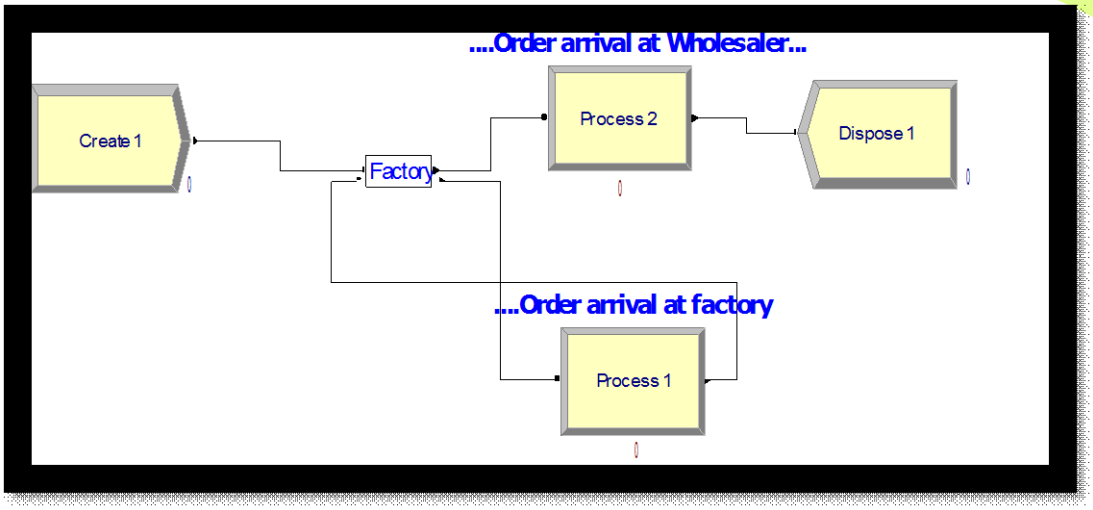


Abbildung 31: Factory Module Arena Test Modell

Das Arena Test-Modell vom Factory Module besteht aus 5 Module: ein Create-Module, zwei Process-Module, 1 Dispose-Module und das Factory Module. Das Create-Module (Create 1) generiert die Wholesaler-Aufträge für das Factory-Modul. Diese Aufträge-Entities betreten das Factory-Modul über einen Entry Point, der für den Informationsfluss bestimmt ist. Das Factory Modul bearbeitet die Wholesaler-Aufträge und liefert die Ware beim Wholesaler aus. Diese Auslieferung erfolgt über einen Exit Point zum Process-Modul (Prozess 2), der für den Materialfluss zum Wholesaler bestimmt ist. Das Factory fordert bei Bedarf Nachschub beim Lieferanten(Supplier) an. Die Nachschubanforderung erfolgt über einen Exit Point zum Process-Modul(Process 1), der für den Informationsfluss zum Lieferanten (Supplier) bestimmt ist. Das Process-Modul(Process 1) simuliert das Supplier-Modul und liefert die Nachschubmenge beim Factory aus. Diese Auslieferung beim Factory erfolgt über einen Entry Point, der für den Materialfluss zum Factory bestimmt ist. Im Dispose-Modul erfolgt nur das Vernichten von den Entitäten. Alle Entitäten, die vernichtet werden sollen, werden ausserhalb des Wholesaler Modules vernichtet. Deswegen werden diese Entitäten auch zum Dispose-Modul(Dispose 1) über einen Exit Point geschickt

2.4.5. Supplier Module

Das Supplier Module simuliert den Supply Chain-Knoten am Ende der Lieferkette. Der Supplier ist der Vorgänger von der Fabrik in der konzipierten Lieferkette. Im Supplier Module werden Komponenten für die Fabrik hergestellt. Beim Supplier ist immer genug Material für die Produktion vorhanden, daher gibt es keinen Bedarf für Nachschub. Die Nachfrage vom Factory kommt zum Supplier über den Informationsfluss-kanal und wird bearbeitet. Falls beim Supplier genug Komponente für den Auftrag vorhanden ist, wird die Bestellmenge vom aktuellen Lagerbestand entnommen und die Ware beim Factory über den Materialfluss-Kanal ausgeliefert. Es gibt keine Teillieferung oder Rückstände beim Supplier. Alle unerfüllten Aufträge werden als entgangene Aufträge registriert. Sobald der Meldebestand beim Supplier erreicht ist, wird die Produktion neue gestartet. Wenn der maximale Lagerbestand erreicht wird, wird automatisch die Produktion gestoppt.

2.4.5.1. Supplier Arena-Modell

Das Supplier Arena Modell implementiert die gleiche Logik wie beim Factory Arena-Modell. Für die Erläuterung des Modelaufbaus wird auf das Factory Arena-Modell verwiesen (Kapitel 9).

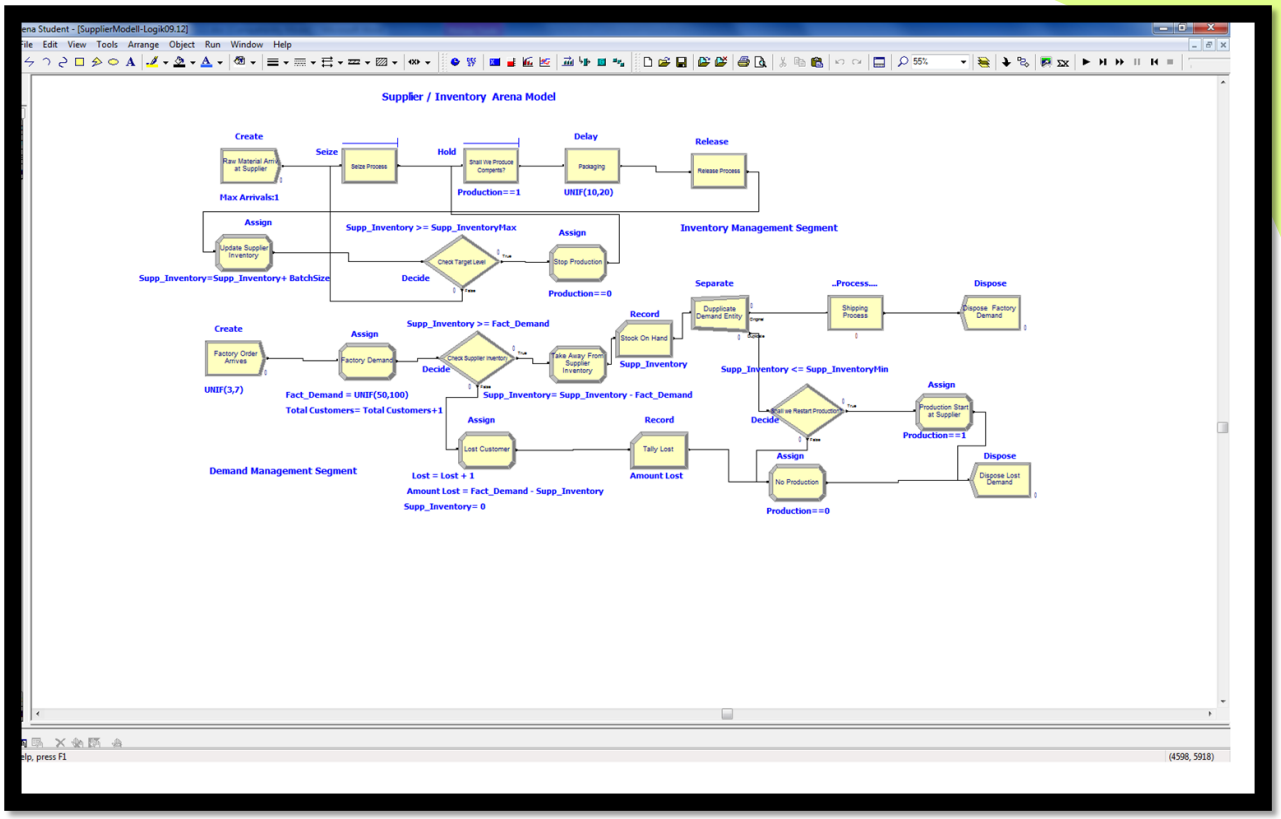


Abbildung 32: Supplier Arena Modell

2.4.5.2. Supplier Module –Logik

Das Supplier Arena-Module-Logik implementiert die gleiche Logik wie beim Factory-Module. Für die Erläuterung des Modelaufbaus wird auf das Factory Arena-Modell verwiesen (Kapitel 10).Der einziger Unterschied liegt darin, dass beim Supplier keine Rückstände und Keine Nachschubanforderung geben.

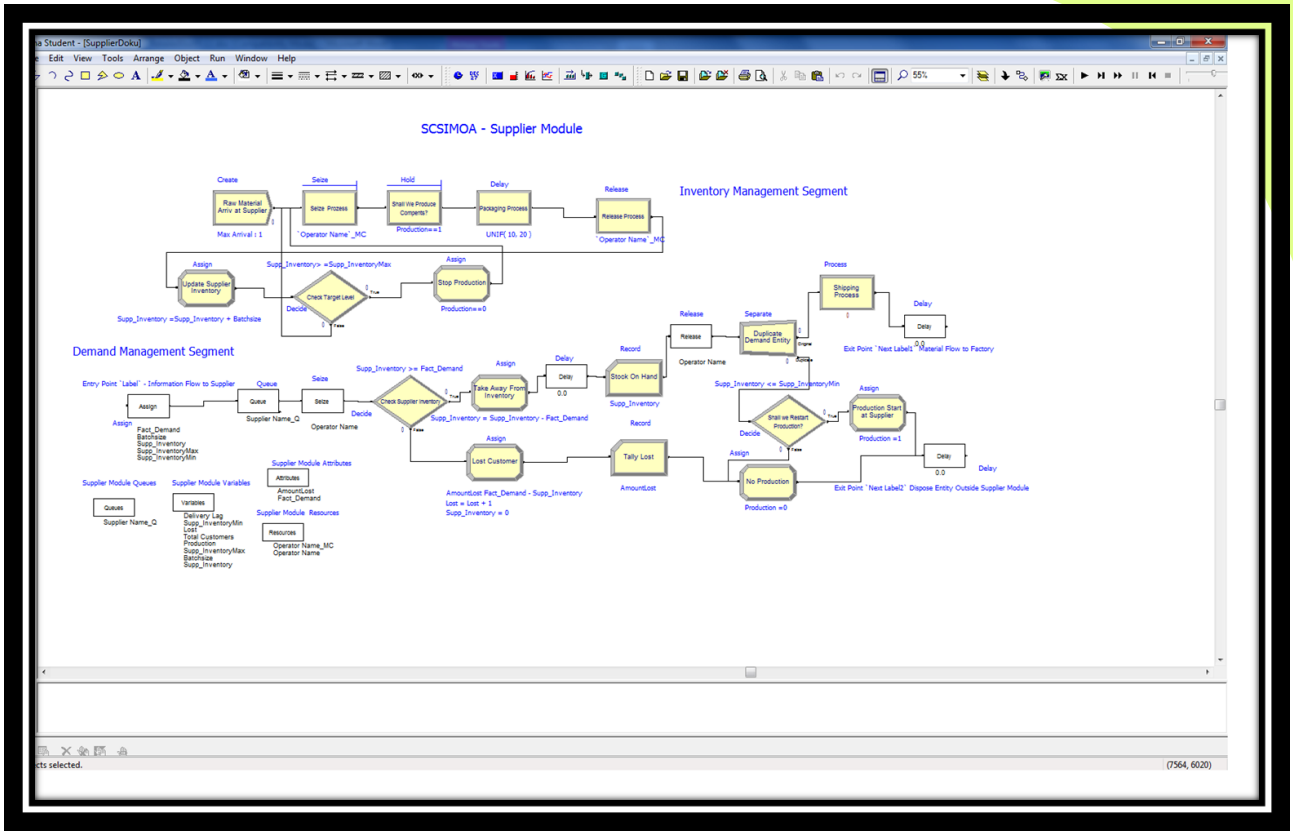


Abbildung 33: Supplier Module Logik

2.4.5.3. Supplier Module User Dialog Form

Für die Erklärung der Modellbildung wird auf das Retailer Module(Kapitel 4) verwiesen.

Abbildung 34: Supplier User Dialog Form

2.4.5.4. Supplier Module User View

Das User View zeigt 1 Entry Point Points (Links) und 2 Exit Points (Rechts) an.

Entry Point:

- Für den Informationfluss vom Factory zu Supplier wird ein Entry Point benötigt.

Exit Points:

- Für den Materialfluss vom Supplier zum Factory wird ein Exit Point benötigt.
- Für die Vernichtung von Entitäten ausserhalb des Factory Modules wird auch ein Exit Point benötigt.

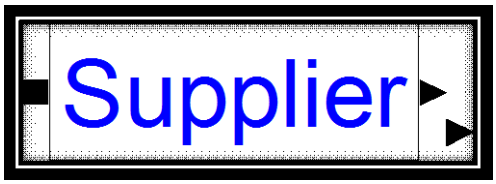


Abbildung 35: Supplier Module User View

2.4.5.5. Supplier Module Panel Icon

Das Panel Icon beim Supplier Module repräsentiert das Module-Symbol im ProjectBar, wenn das Supplier Module ins Arena-Tool importiert(Arena>File >Template Panel>Attach) wird.

Abbildung 36: Supplier Module Panel Icon

2.4.5.6. Supplier Module Arena Test-Modell

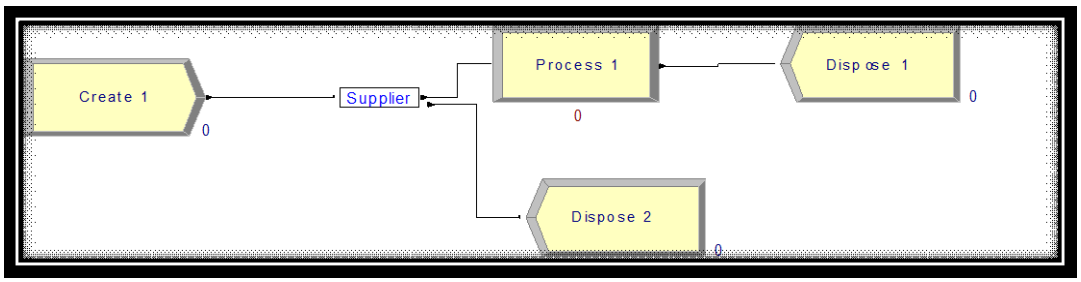


Abbildung 37: Supplier Module Arena Test Modell

3. Performance Messungen

3.1. Retailer-Module Key Performance Indicators

3.1.1. Beispielsimulation mit dem Retailer Module

a. Szenario :

Ein Einzelhändler (Retailer) verkauft in Wagen Notebook- Taschen. In seinem Lager kann er Maximum 500 Artikeln halten. Er arbeitet mit einem Startbestand von 250 Artikeln. Eine Lagerbestandführung wird jeden Tag beim Retailer durchgeführt. Wenn der Lagerbestand den Meldebestand von 150 Artikeln erreicht, wird eine Nachschubmenge von

(R –I(t)) Artikeln beim Lieferanten angefordert. Er Retailer arbeitet mit einem (r, R) Inventory System. Mit r = Melde-bestand, R = Maximalbestand.

- I(t) = Lagerbestand beim Retailer im Zeitpunkt (t).
- I(0) = 250 initialisiert den Startbestand am Beginn der Simulation
  Kunden betreten das Retailer-Geschäft mit einer bestimme Nachfrage. Zunächst wird die vom Kunden nach-gefragte Menge, Falls ein entsprechender Lagerbestand vorhanden ist, ausgeliefert. Anderenfalls wird die entgangene Menge registriert. Für die entgangene Aufträge werden Lagerzinsen (Unit Shortage Cost) be-rechnet. Diese Kosten werden bilden zusammen mit den Bestellkosten beim Lieferanten die Gesamtkosten beim Retailer.

b. Ankunftsprozess:
   - Die Kunden kommen zum Retailer mit einer exponentiell verteilten Zwischen Ankunftszeit an. („Ran-dom(Expo)“ mit Value: „0.5“).
   - Die Auftragsmenge ist als diskrete Verteilung mit folgende Variablen und Wahrscheinlichkeiten defi-niert :
     - Variablen : 3,5,8,7,6 mit respektive Wahrscheinlichkeit
     - 0.15,0.25,0.4,0.9,1.0
   - DISC(0.15,3,0.25,5,0.4,8,0.9,7,1.0,6)

c. Simulationslauf:
   Im Run Setup werden unter “Project Parameters“ Projektparameter für den Simulationslauf festgelegt:
   - Warm-up Periode : 0.0 (Einschwingphase)
   - Replication Lengh: 2400 (Dauer eines Simulationslaufes)
   - Time Units : Hours (Zeiteinheit des Simulationslaufes)
   - Base Time Units : Hours

Oft ist die sogenannte Einschwingphase nötig. Das ist der Zeitabschnitt von Beginn der Simulation bis zu dem Punkt, an dem das System alle „Startunregelmässigkeiten“ abgelegt hat. Fängt man z.B mit einem leeren System an, was in der Realität eher selten vorkommt, so muss sich erst ein Normalzustand „einpendeln“. Stellt man im Run Setup unter Warm-Up Periode die Zeitspanne des „Einpendelns“ ein, so werden erst danach Ergebnisse aufgezeichnet. Die Ver-fälschten Werte fliessen nicht in die Endergebnisse ein. Start man die Simulation mit dem *Play Button,* so stoppt sie, wenn die Terminierungsbedingung erfüllt ist oder läuft unendlich lange. Ebenfalls im Run Setup wird als *Terminating Condition* ein Ausdruck eingegeben und Simulation stoppt, wenn er erfüllt ist.

d. Simulationsreport:
   Terminiert die die Simulation, kommt man nach einer Bestätigung sofort zum Simulationsreport. Hier wer-den die Ergebnisse, die während der Simulation gesammelt wurden, angezeigt. Im linken Teilfenster steht dann ein Reports Panel. Dort lässt sich für jeden Modellbestandteil per Doppelklick ein eigenes Reportfenster öffnen. Öffnet man den Verzeichnisbaum des Projektes und Wählt man z.B. „Queue“ aus, erhält man unter *Waiting Time* die durchnittliche Wartezeit eines Retailer Kunden in dieser Simulation. Die mittlere Anzahl von Retailer Kunden in der Schlange erhält man unter „*Number Waiting*“.
   Im Run Setup kann man auch unter *Replication Parameters* den Wert *Number of Replications* >1 setzen. Dies ermöglicht mehrere Simulationsdurchläufe mit dem selben Modell und identischen Parameters. Mit mehre-ren Replikationen simuliert, erhält man im Report Ergebnisse, die sich auf alle Replikationen beziehen und auch Ergebnisse jedes einzelnen Laufs.

   Nota bene: *Mehr als nur ein Simulationslauf ist in jedem Fall für eine System-Analyse sinnvoll, da die Werte eines einzelnen Laufes meist nicht aussagekräftig genug sind. Bei Dieser Bachelorarbeit wurde nicht viel Wer-te auf Simulationsläufe gesetzt, da wir keine konkrete System-Analyse machen sondern nur an die Logik eines Supply chain Simulationsmodelles interessiert sind.*

e. Statistik Erfassungen:
   Folgende Statistik-Werte werden bei der Simulation ermittelt:
   - Serviceleistungen beim Retailer (Customer Service Level)

- Durchnittlicher Lagerbestand beim Retailer (Stock On Hand Inventory)
- Lagerumschlagshäufigkeit (Ordering == 1)
- Lagerhaltungskosten

Mit dem Statistik-Modul in Arena Template Panel (Advanced Process)können verschiedene Statistik-Messungen definiert werden. Die folgende Tabelle beinhaltet die Statistischen Erfassungen von Messwerten für das Retailer-Modul:

Statistic - Advanced Process

	Name	Type	Expression	Report Label	Output File	Frequency Type	Resource Name	Report Label	Output File	Categories
1	A. Stock On Hand	Time-Persistent	Retailer_Inventory	A. Stock On Hand		Value		A. Stock On Hand		0 rows
2	B. Process State	Frequency		B. Process State		State	Service Operator	B. Process State		0 rows
3	C. Ordering O.K	Time-Persistent	Ordering==1	C. Ordering O.K		Value		C. Ordering O.K		0 rows
4	D. Lost Percentage	Output	Lost/Total Customers	D. Lost Percentage		Value		D. Lost Percentage		0 rows
5	F.Retailer Customers Service Level	Output	AErfuellteAuftraege/ANEAuftraege	F.Retailer Customers Service Level		Value		F.Retailer Customers Service Level		0 rows
6	G. Average Ordering Cost	Output	Total Ordering Cost/Days To Run	G. Average Ordering Cost		Value		G. Average Ordering Cost		0 rows

Double-click here to add a new row.

Abbildung 38: Retailer-Module Statistik-Messungen

Die Statistik-Tabelle für das Retailer-Module beinhaltet für die Variable „Retailer_Inventory“ einen Statistik-Wert von Type „Time-Persistent“ mit dem Namen „Stock On Hand“ und einen anderen „Time-Persistent“ Statistik-Wert mit dem Namen „Ordering O.k“ für die Expression „ Ordering OK ==1“. Die Statisk-Messung für „Retailer_Inventory“ werden folgende Parameter gemessen:

- Standardabweichung
- Vertrauensbereich(Konfidenz-Intervall) , sowie
- Minimal und Maximal Wert während eines Simulationslaufes.

Für die Expression „Ordering==1“ wird ausgegeben, wie oft diese Expression Wahr ist. Dies ergibt die Häufigkeit der Nachschubanforderung beim Retailer.
Der Statistik-Wert von Type “Frequency“ mit dem Namen „Process State“, definiert die Häufigkeit vom Zustand „Busy“ oder „down“ beim Verkaufsprozess.
Der Statistik-Wert von Type Output mit dem Namen „Lost Percentage“ definiert der Prozentsatz von den entgangenen Aufträgen beim Retailer. Die Expression „Lost/Total Customers“ besagt, dass am Ende des Simulationslaufes die Variable „Lost“ durch die Variable „Total Customers“ dividiert wird, um den Prozentsatz „Lost Percentage“ zu bestimmen.
Der Statistik-Wert von Type Output mit dem Namen „ Retailer Customer Service Level“ bestimmt den Kundenzufriendenheitsgrad beim Retailer. Die Variable „AErfuellteAuftraege“ wird durch die Variable „ANAuftraege“ dividiert, um den Service Level zu bestimmen. Die Variable „AErfuellteAuftraege“ definiert die Anzahl erfüllte Auftraege und die Variable „ANAuftraege“ die Anzahl nicht erfüllte Auftraege.

f. Ergebnisse der Simulation:

1) **Catagory Overview**

Zusammenfassende Statistiken und Informationen zu Gesamtsystem, Fördermitteln (conveyor), Objekten (entities), Prozessen (processes), Warteschlangen (Queues), Ressourcen (resources), Transportern (transporter) und benutzerdefinierten Variablen (user specified) über alle Replikationen

- *Entity*
  Unter Entity Overview werden die Statistiken für die Entitäten(KundenAuftrag) im Retailer-System angezeigt. Unter *„Number In“ in der Abbildung 24 (Other Overview) ist ersichtlich, dass die Entität „KundenAuftrag“ den Wert (Value) 34382 hat. Unter „Number Out“ hat die Entität „KundenAuftrag“ den Selben Wert (34382). „Number In“ gibt Anzahl Kunden an, die das Retailer-Geschäft betreten haben. Und „Number Out“*

gibt Anzahl Kunden an, die das Retailer-Geschäft verlassen haben. Diese beide Variablen müssen deswegen denselben Wert haben.Während dieser Simulation waren 34384 Kunden beim Retailer.

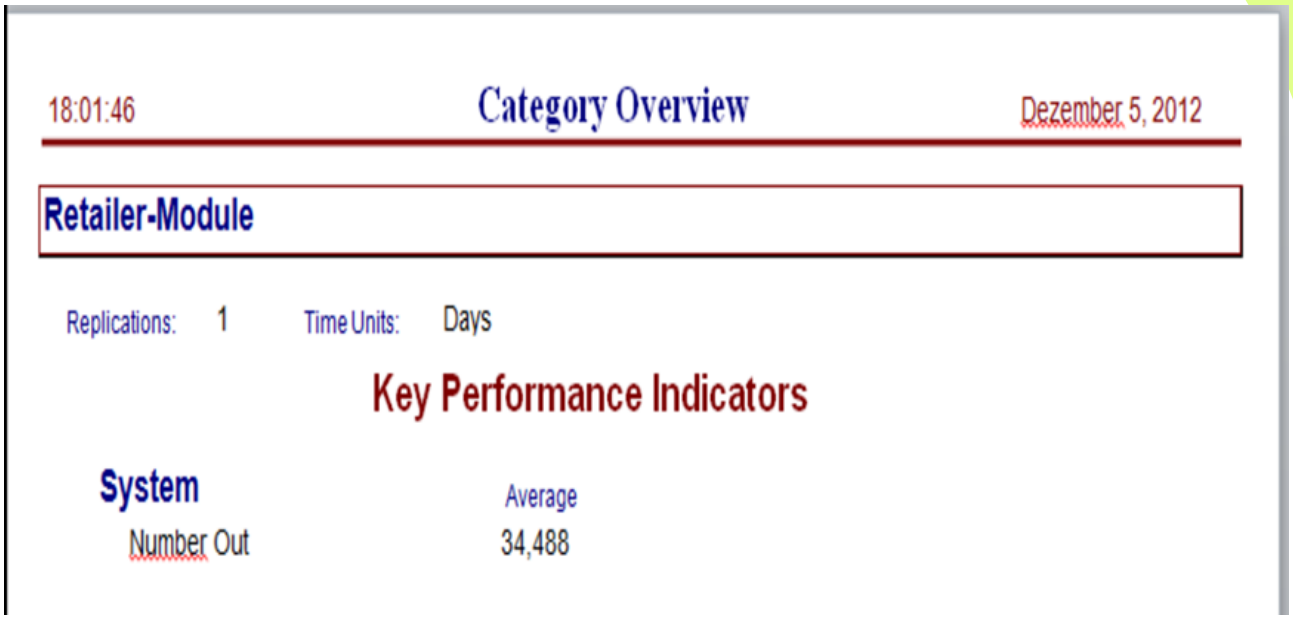


Abbildung 39: Category Overview

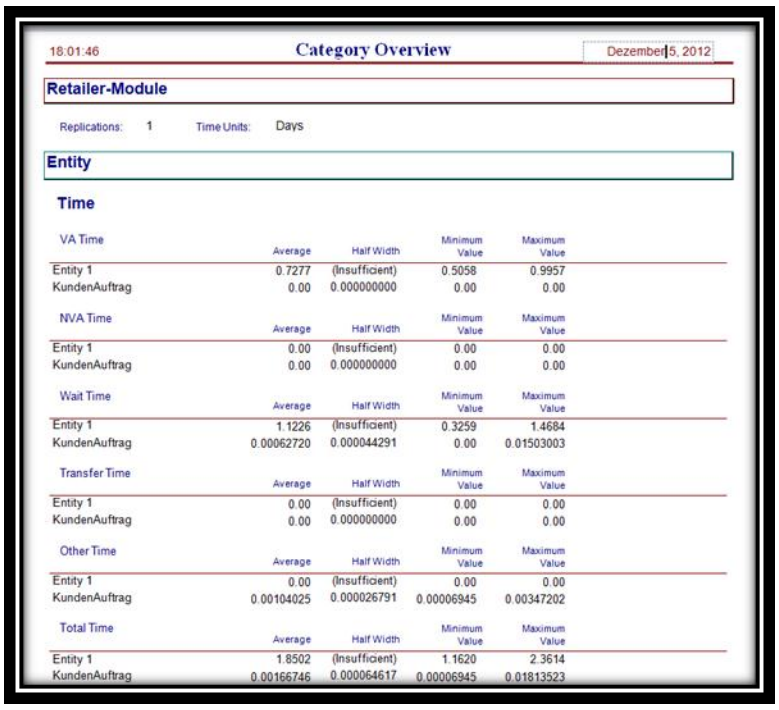


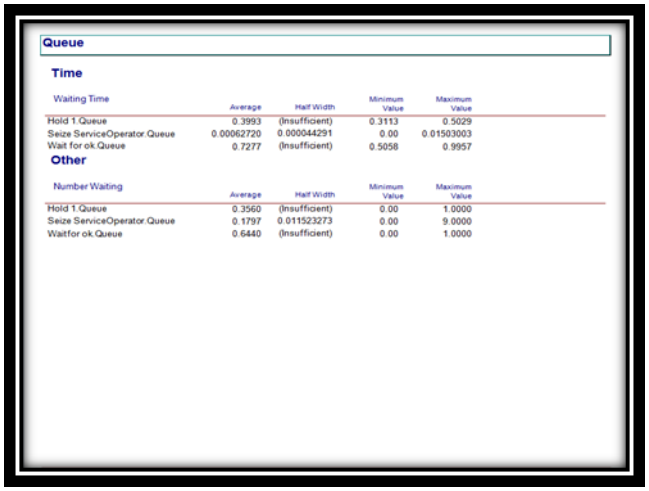
Abbildung 40: Entity Overview

Abbildung 41: Other Overview

- *Queue*

  Unter „Queue“ im Queue Overview Abbildung 27 werden die Wartezeiten

  in den verschiedenen Warteschlangen angezeigt. Unten „Waiting Time“ gibt die Zahl „0.000062“ die mittlere Wartezeit eines Kunden in der Schlange „Size ServiceOperator“ an. Diese simuliert die Wartezeit an der Kasse beim Retailer-Geschäft. Unter „Number Waiting“ gibt die Zahl „0.17“ die mittlere Anzahl Kunden in dieser Schlange an.



Queue

Time

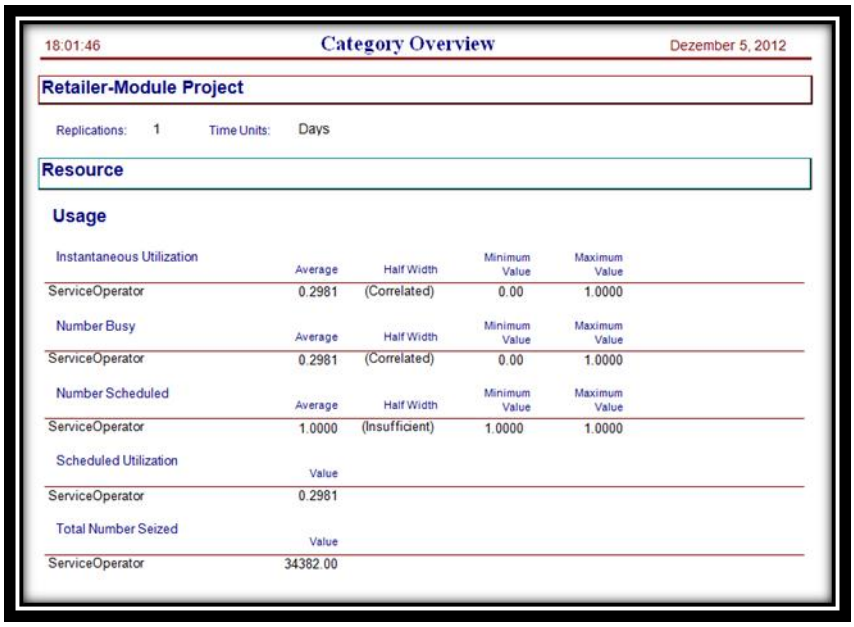
Waiting Time	Average	Half Width	Minimum Value	Maximum Value
Hold 1.Queue	0.3993	(Insufficient)	0.3113	0.5029
Seize ServiceOperator.Queue	0.00062720	0.00004291	0.00	0.01503003
Wait for ok.Queue	0.7277	(Insufficient)	0.5058	0.9957

Other

Number Waiting	Average	Half Width	Minimum Value	Maximum Value
Hold 1.Queue	0.3560	(Insufficient)	0.00	1.0000
Seize ServiceOperator.Queue	0.1797	0.011523273	0.00	9.0000
Waitfor ok.Queue	0.6440	(Insufficient)	0.00	1.0000

Abbildung 42: Queue Overview

- *Resource*
  Unter „Resource Overview“ werden die Statistiken über die Resource im Retailer-System angezeigt. In diesem Beispielsimulation wurde eine Resource (Service Operator)verwendet. „Usage“ gibt die durchnittliche Auslastung des Service Operators(Beschäftigte Angestellte).



18:01:46 Category Overview Dezember 5, 2012

Retailer-Module Project

Replications: 1 Time Units: Days

Resource

Usage

Instantaneous Utilization	Average	Half Width	Minimum Value	Maximum Value
ServiceOperator	0.2981	(Correlated)	0.00	1.0000

Number Busy	Average	Half Width	Minimum Value	Maximum Value
ServiceOperator	0.2981	(Correlated)	0.00	1.0000

Number Scheduled	Average	Half Width	Minimum Value	Maximum Value
ServiceOperator	1.0000	(Insufficient)	1.0000	1.0000

Scheduled Utilization	Value
ServiceOperator	0.2981

Total Number Seized	Value
ServiceOperator	34382.00

Abbildung 43: Resource Overview

- Tally und Output Report
  Im „User Specified“- Teilfenster (Abbildung 29)werden die Ergebnisse für benutzerdefinierten Variablen angezeigt. Unter „Tally“ werden die im Record-Module definierten Statistik-Variablen ausgewertet. Unter „Expression“ sind folgende Tally-Werte zu finden:

- Die durchnittliche Anzahl von entgangene Aufträge pro Kunden beträgt 6.15 für die Expression „Avg Lost per Customer“
- Die durchnittliche Anzahl von Verluste durch entgangene Aufträge beträgt 30.79 für die Expression „Avg Lost Revenue at Retailer“
- Die Durchschnittliche Anzahl Bestellkosten beträgt 1728 für die Expression „ Avg Ordering Cost at Retailer“
- Die Durchnittliche Anzahl Auftragsmenge beträgt 454599.00 für die Expression „ Tally Retailer Demand“

Unter „ Time-Persistent“ ist die Varianz der Variable „RInventory“ (Lagerbestand) angezeigt. Der mittlere Wert für den Lagerbestand gibt die Variable „A.Stock On Hand“ und beträgt 185.94.

Unter „Output“ in dieser Tally- Sektion sind folgende Statistik-Werte zu finden:

- Prozentsatz der entgangenen Aufträge gibt die Variable „C.Lost Percentage“ an und beträgt 57%
- Die Serviceleistung beim Retailer gibt die Variable „Customer Service Level“ an und beträgt 75%

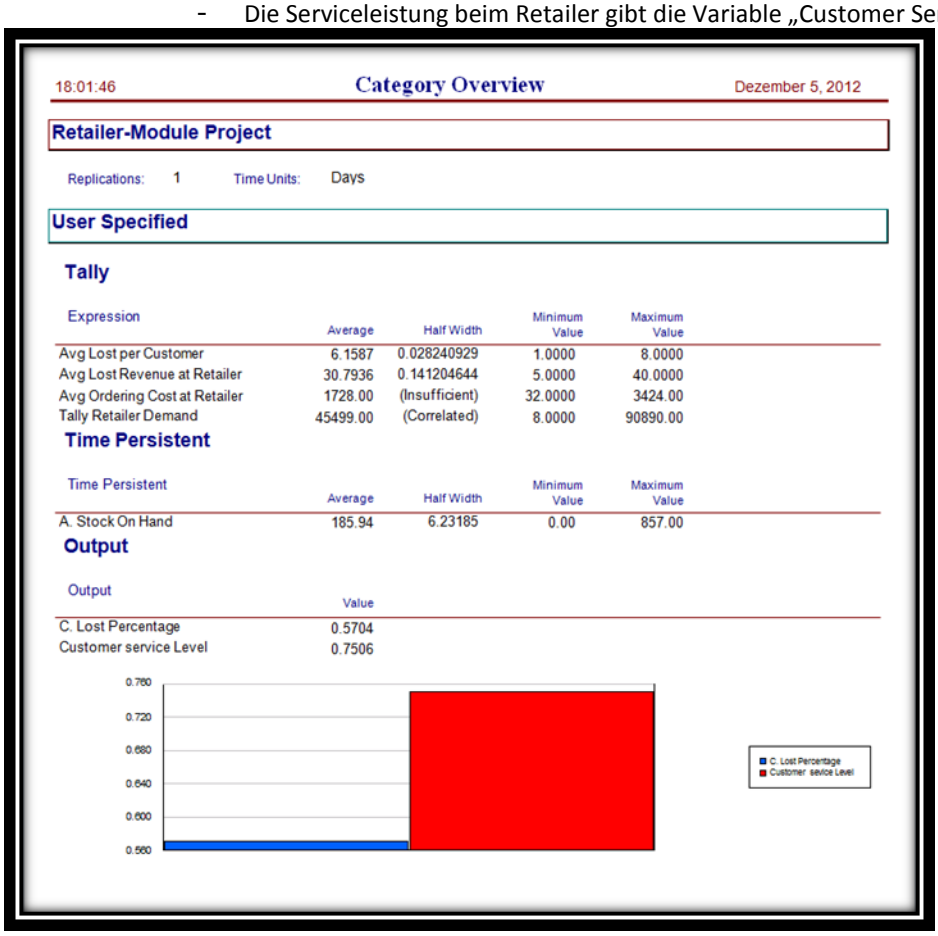


Abbildung 44: Tally und Output Overview

3.2. Wholesaler-Module Key Performance Indicators

3.2.1. Beispielsimulation mit dem Wholesaler Module

a. Szenario :

Der Grosshändler DC-Darou(Wholesaler) ist der direkte Notebook-Taschen- Lieferant vom Retailer. Er erhält die Bestellungen von Retailer. Er arbeitet wie der Retailer mit einem (r, R) Inventory. Mit r = Meldebestand, R = Maximalbestand. In seinem Lager kann er maximum 500 Artikeln halten. Er arbeitet mit einem Startbestand von 250 Artikeln. Eine Lagerbestandführung wird jeden Tag beim Wholesaler durchgeführt. Wenn der Lagerbestand den Meldebestand von 150 Artikeln erreicht, wird eine Nachschubmenge von (R –I(t)) Artikeln beim Lieferanten angefordert.

- I(t) = Lagerbestand beim Wholesaler im Zeitpunkt (t).
- I(0) = 250 initialisiert den Startbestand am Beginn der Simulation
  Retailer betreten das Wholesaler-Geschäft mit einer bestimme Nachfrage. Zunächst wird die vom Kunden nachgefragte Menge, Falls ein entsprechender Lagerbestand vorhanden ist, ausgeliefert. Anderenfalls wird die Fehlmenge als Ruckstand registriert. Für die Rückstände werden Lagerzinsen (Unit Shortage Cost) berechnet. Diese Kosten werden bilden zusammen mit den Bestellkosten beim Lieferanten die Gesamtkosten beim Retailer. Sobald genug Bestand vorhanden ist, werden die KundenAufträge erfüllt.

b. Ankunftsprozess :

- Die Kunden kommen zum Wholesaler mit einer exponentiell verteilten Zwischen Ankunftszeit an. („Random(Expo)“ mit Value: „0.5“).
- Die Auftragsmenge ist als diskrete Verteilung mit folgende Variablen und Wahrscheinlichkeiten definiert :
  - Variablen : 3,5,8,7,6 mit respektive Wahrscheinlichkeit
  - 0.15,0.25,0.4,0.9,1.0
- DISC(0.15,3,0.25,5,0.4,8,0.9,7,1.0,6)

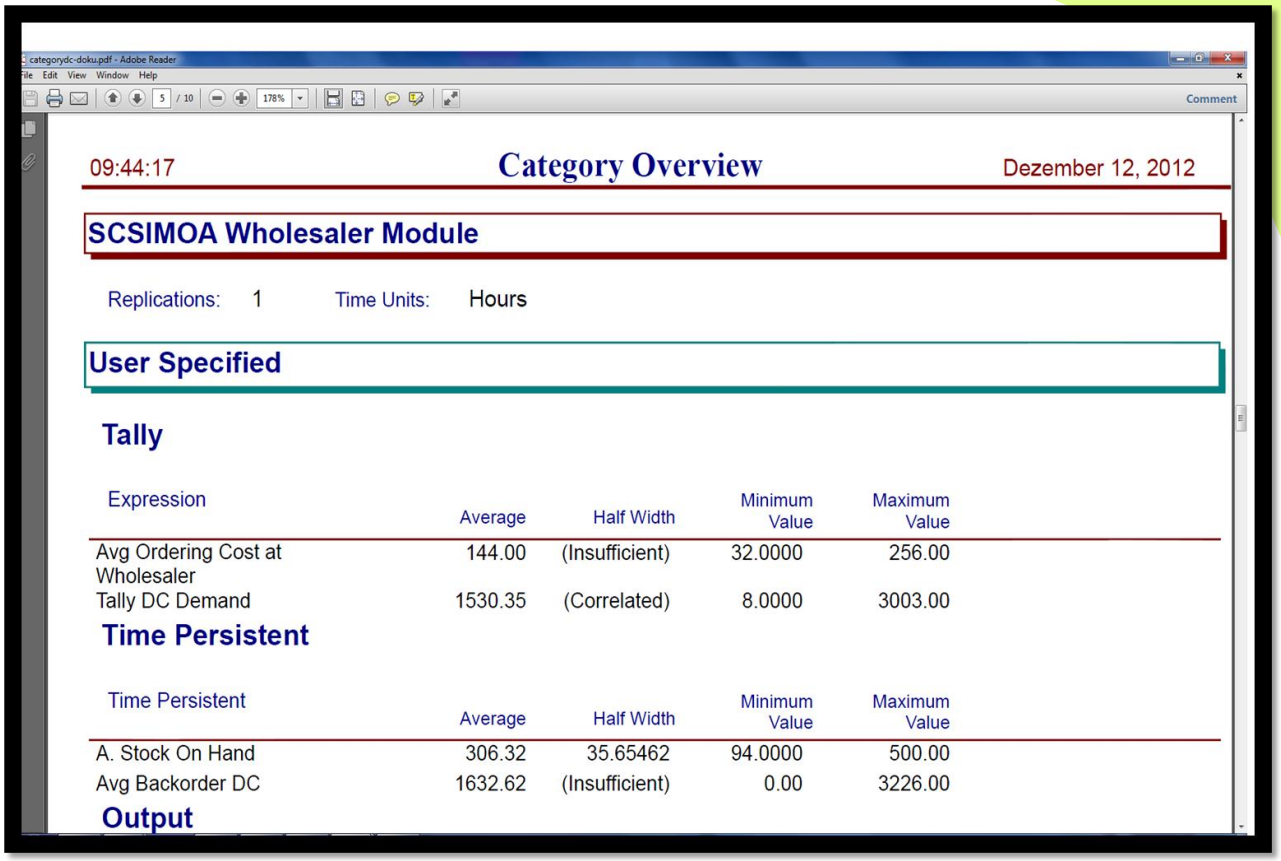
c. Simulationslauf:
Im Run Setup werden unter “Project Parameters“ Projektparameter für den Simulationslauf festgelegt:
  - Warm-up Period : 0.0 (Einschwingphase)
  - Replication Lengh: 2400 (Dauer eines Simulationslaufes)
  - Time Units : Hours (Zeiteinheit des Simulationslaufes)
  - Base Time Units : Hours

d. Statistik Erfassungen:
Folgende Statistik-Werte werden bei der Simulation ermittelt:
- Serviceleistungen beim Wholesaler (Customer Service Level)
- Durchnittlicher Lagerbestand beim Wholesaler (Stock On Hand Inventory)
- Lagerumschlagshäufigkeit (Ordering == 1)
- Lagerhaltungskosten
- Durchschnittliche Anzahl Rückstände (Backorders) beim Wholesaler

Statistic - Advanced Process

	Name	Type	Expression	Frequency Type	Resource Name	Collection Period	Report Label	Output File	Report Label	Output File	Categories
1	A. Stock On Hand	Time-Persistent	DC_Inventory	Value		Entire Replication	A. Stock On Hand		A. Stock On Hand		0 rows
2	B. Process State	Frequency		State	Service Operator	Entire Replication	B. Process State		B. Process State		0 rows
3	C. Ordering Ok	Time-Persistent	DC_Inventory <= DC_InventoryMin	Value		Entire Replication	C. Ordering Ok		C. Ordering Ok		0 rows
4	D. Average Backorders	Time-Persistent	Backorders_DC	Value		Entire Replication	D. Average Backorders		D. Average Backorders		0 rows
5	E. Customer Service Level	Output	Backorders_DC / DC_Customers	Value		Entire Replication	E. Customer Service Level		E. Customer Service Level		0 rows

Abbildung 45: Wholesaler-Module Statistik-Messungen



09:44:17 **Category Overview** Dezember 12, 2012

SCSIMOA Wholesaler Module

Replications: 1 Time Units: Hours

User Specified

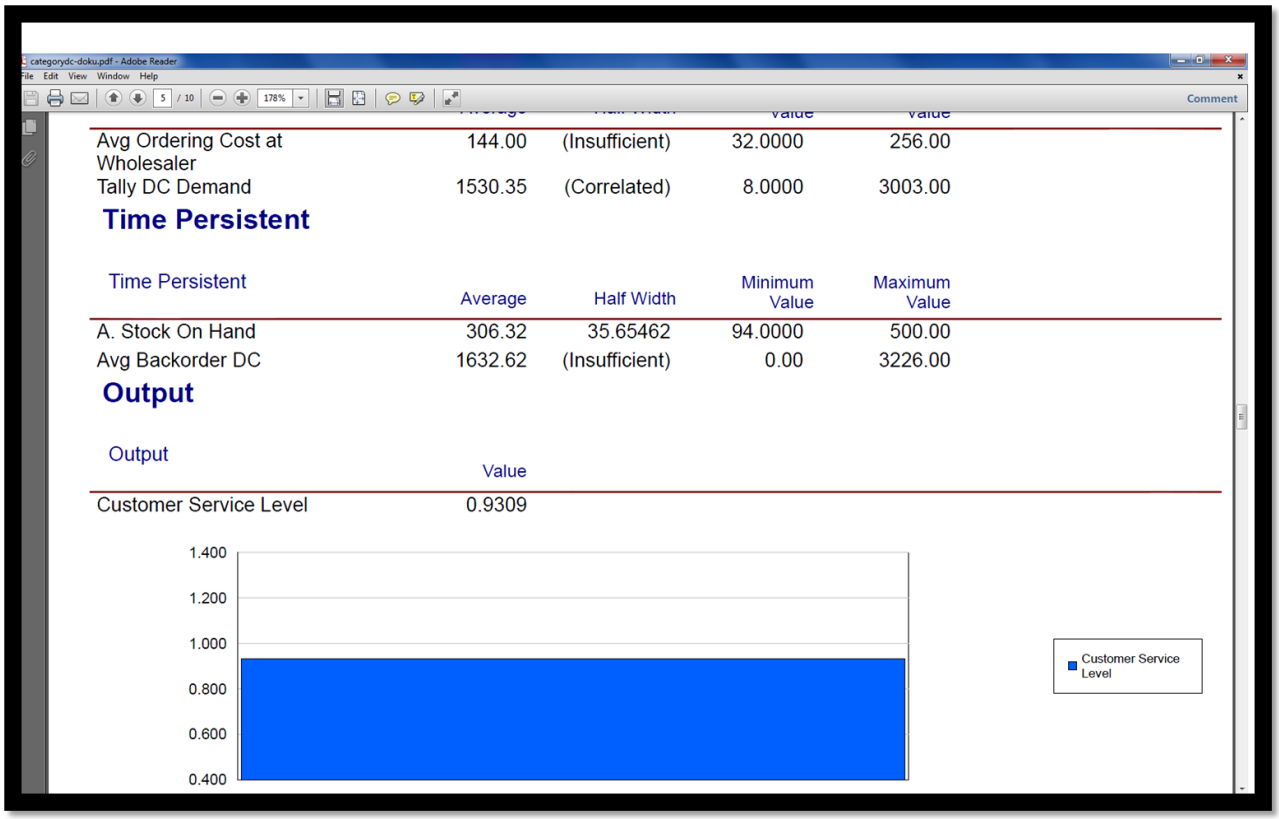
Tally

Expression	Average	Half Width	Minimum Value	Maximum Value
Avg Ordering Cost at Wholesaler	144.00	(Insufficient)	32.0000	256.00
Tally DC Demand	1530.35	(Correlated)	8.0000	3003.00

Time Persistent

Time Persistent	Average	Half Width	Minimum Value	Maximum Value
A. Stock On Hand	306.32	35.65462	94.0000	500.00
Avg Backorder DC	1632.62	(Insufficient)	0.00	3226.00

Output



Avg Ordering Cost at Wholesaler 144.00 (Insufficient) 32.0000 256.00
Tally DC Demand 1530.35 (Correlated) 8.0000 3003.00

Time Persistent

A. Stock On Hand 306.32 35.65462 94.0000 500.00
Avg Backorder DC 1632.62 (Insufficient) 0.00 3226.00

Output

Output	Value
Customer Service Level	0.9309

Abbildung 46: Wholesaler Statistik-Messungen

Der Simulationsreport zeigt in der Benutzerspezifische Sektion folgende Werte an:

- Customer Service Level beim Wholesaler beträgt 93% in der “Output ”Sektion
- Der mittlere Wert für den Lagerbestand beim Wholesaler liegt bei 306 und für Rückstände (Backorders) liegt bei 163 in der „Time-Persistent“ Sektion
- Die „Tally“-Sektion gibt die Statistik-Werte im Record-Module an. Die Variable „Avg Ordering Cost at Wholesaler“ beträgt 144 und gibt den mittleren Wert für die Bestellkosten an. Die Variable „Tally DC Demand“ beträgt 1530 und gibt den mittleren Wert für die Auftragsmenge beim Wholesaler.

3.3. Factory-Module Key Performance Indicators

3.3.1. Beispielsimulation mit dem Factory Module

e. Szenario :

Factory-Touba stellt in Wagen Notebook-Taschen her und arbeitet zusammen mit verschiedenen Grosshändlern in der Region. Die Factory-Touba produziert maximum 500 Notebook-Taschen und arbeite mit einem Startbestand von 250 Taschen. Wenn der Meldebestand r = 150 Taschen beim Verkauf erreicht wird, startet automatisch die Produktion bis wieder der Lagerbestand den Maximalbestand R = 500 erreicht. Die Factory-Touba arbeitet auch mit einem (r, R) Inventory-Modell. Für die Produktion werden Komponenten benötigt und müssen auch bei einem Lieferant bestellt werden. Bei der Produktion wird immer ein Set von 5 Taschen hergestellt. Für die Verpackung werden 10 bis 20 Minuten benötigt.
Die Nachfrage von Wholesaler wird immer vom aktuellen Lagerbestand entnommen. Falls nicht genug Bestand vorhanden, wird eine Nachschubmenge(R- I(t)) an Komponente für die Produktion angefordert. Für Rückstände werden registriert und nachgeliefert sobald genug Bestand vorhanden ist.

- I(t) = Lagerbestand beim Factory-Touba im Zeitpunkt (t).
- I(0) = 250 initialisiert den Startbestand am Beginn der Simulation

f. Ankunftsprozess :

- Die Wholesaler kommen zum Factory-Touba mit einer exponentiell verteilten Zwischen Ankunftszeit an. („Random(Expo)“ mit Value: „0.5“).
- Die Auftragsmenge ist als diskrete Verteilung mit folgende Variablen und Wahrscheinlichkeiten definiert :
  - Variablen : 3,5,8,7,6 mit respektive Wahrscheinlichkeit
  - 0.15,0.25,0.4,0.9,1.0

- DISC(0.15,3,0.25,5,0.4,8,0.9,7,1.0,6)

g. Simulationslauf:
Im Run Setup werden unter “Project Parameters“ Projektparameter für den Simulationslauf festgelegt:
  - Warm-up Period : 0.0 (Einschwingphase)
  - Replication Lengh: 2400 (Dauer eines Simulationslaufes)
  - Time Units : Hours (Zeiteinheit des Simulationslaufes)
  - Base Time Units : Hours

h. Statistik Erfassungen:
Folgende Statistik-Werte werden bei der Simulation ermittelt:

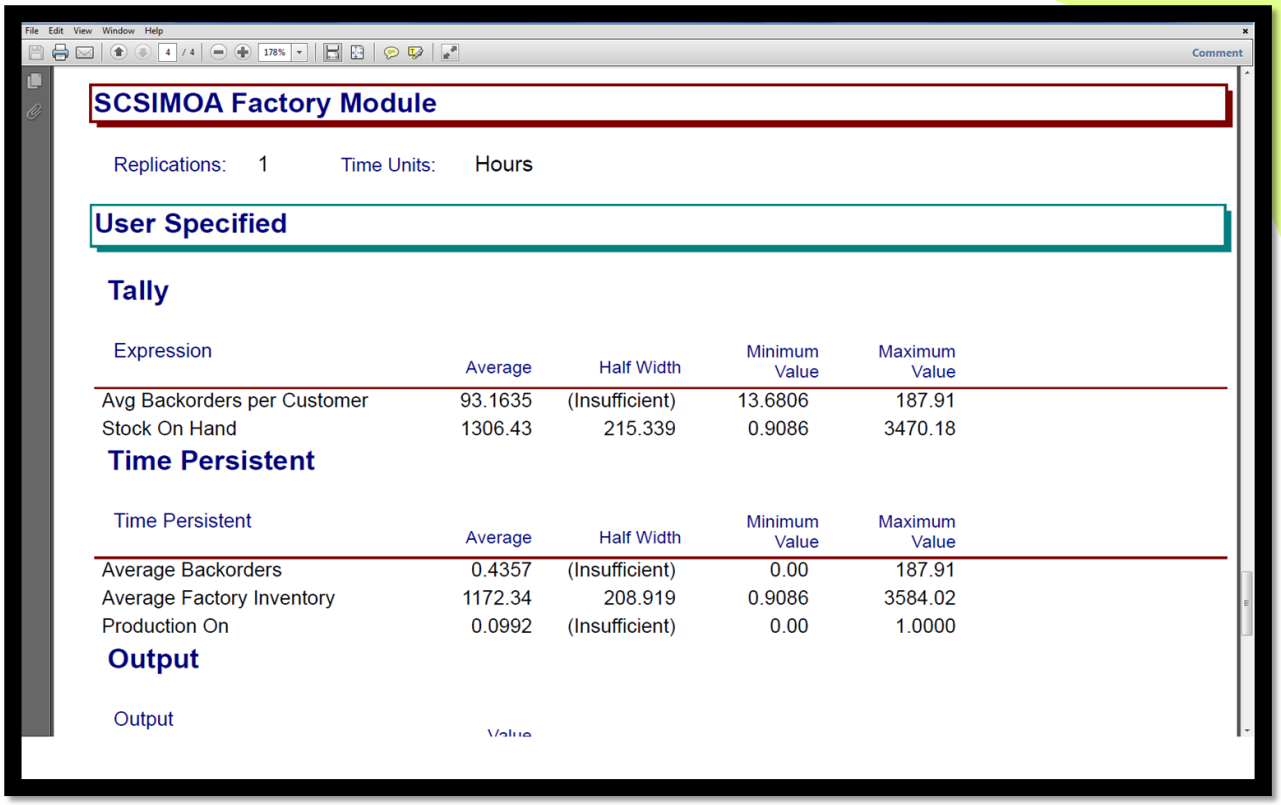
- Serviceleistungen beim Factory (Factory Customers Service)
- Durchnittlicher Lagerbestand beim Factory (Stock On Hand)
- Lagerumschlagshäufigkeit (Production_Plant == 1)
- Lagerhaltungskosten
- Durchschnittliche Anzahl Rückstände beim Factory (Factory_Backorder)

Statistic - Advanced Process

	Name	Type	Expression	Report Label	Output File	Frequency Type	Resource Name	Report Label	Output File	Categories
1	A. Stock On Hand	Time-Persistent	Factory_Inventory	A. Stock On Hand		Value		A. Stock On Hand		0 rows
2	B. Process State	Frequency		B. Process State		State	Service Operator	B. Process State		0 rows
3	C. Plant Utilization	Time-Persistent	Production_Plant==1	C. Plant Utilization		Value		C. Plant Utilization		0 rows
4	D. Factory Avg. Backorders	Time-Persistent	Factory_Backorder	D. Factory Avg. Backorders		Value		D. Factory Avg. Backorders		0 rows
5	F. Factory Customers Service	Output	AErfuellteAuftraege/ANEAuftraege	F. Factory Customers Service		Value		F. Factory Customers Service		0 rows
6	G. Average Ordering Cost	Output	Total Ordering Cost/Days To Run	G. Average Ordering Cost	...	Value		G. Average Ordering Cost		0 rows

Double click here to add a new row

Abbildung 47: Factory-Module Statistik-Messungen



SCSIMOA Factory Module

Replications: 1 Time Units: Hours

User Specified

Tally

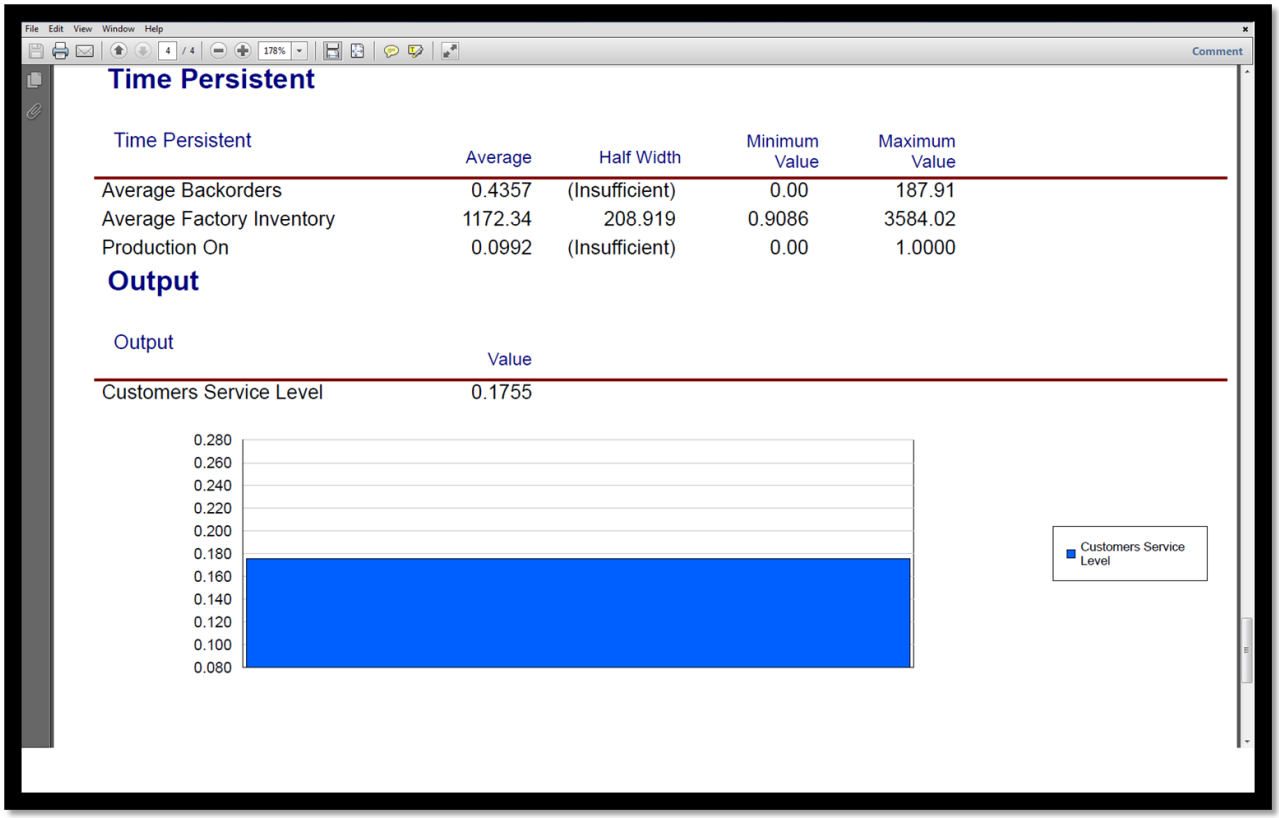
Expression	Average	Half Width	Minimum Value	Maximum Value
Avg Backorders per Customer	93.1635	(Insufficient)	13.6806	187.91
Stock On Hand	1306.43	215.339	0.9086	3470.18

Time Persistent

Time Persistent	Average	Half Width	Minimum Value	Maximum Value
Average Backorders	0.4357	(Insufficient)	0.00	187.91
Average Factory Inventory	1172.34	208.919	0.9086	3584.02
Production On	0.0992	(Insufficient)	0.00	1.0000

Output

Output	Value



Time Persistent

Time Persistent	Average	Half Width	Minimum Value	Maximum Value
Average Backorders	0.4357	(Insufficient)	0.00	187.91
Average Factory Inventory	1172.34	208.919	0.9086	3584.02
Production On	0.0992	(Insufficient)	0.00	1.0000

Output

Output	Value
Customers Service Level	0.1755

Abbildung 48: Factory Statistik Messungen

Der Simulationsreport zeigt in der Benutzerspezifische Sektion folgende Werte an:

- Customer Service Level beim Factory-Touba beträgt 17% in der "Output "Sektion
- Der mittlere Wert für den Lagerbestand beim Factory-Touba liegt bei 1306 und für Rückstände (Backorders) liegt bei 43% in der „Time-Persistent" Sektion
- Die „Tally"-Sektion gibt die Statistik-Werte im Record-Module an. Die Variable „Avg Backorders per Customer" beträgt 93%.

3.4. Supplier-Module Key Performance Indicators

3.4.1. Beispielsimulation mit dem Supplier Module

i. Szenario :

Supply-TB stellt in Winden Komponenten für Notebook-Taschen her und arbeitet zusammen mit verschiedenen Factories in der Region. Die Supply-TB produziert maximum 500 Komponenten und arbeite mit einem Startbestand von 250 Komponenten. Wenn der Meldebestand r = 150 Taschen beim Verkauf erreicht wird, startet automatisch die Produktion bis wieder der Lagerbestand den Maximalbestand R = 500 erreicht. Die Factory-Touba arbeitet auch mit einem (r, R) Inventory-Modell. Für die Produktion ist immer genug Roh-Material vorhanden. Bei der Produktion wird immer ein Set von 5 Taschen hergestellt. Für die Verpackung werden 10 bis 20 Minuten benötigt.Der Supply-TB steht am Ende der Lieferkette.
Die Nachfrage von Factory-Touba wird immer vom aktuellen Lagerbestand entnommen. Falls nicht genug Bestand vorhanden, werden die unerfüllten Aufträge als entgangene Aufträge registriert. Es gibt keine Teillieferung oder Rückstände beim Supply-TB.

- I(t) = Lagerbestand beim Factory-Touba im Zeitpunkt (t).
- I(0) = 250 initialisiert den Startbestand am Beginn der Simulation

j. Ankunftsprozess :

- Die Kunden kommen zum Supply-TB mit einer exponentiell verteilten Zwischen Ankunftszeit an. („Random(Expo)" mit Value: „0.5").
- Die Auftragsmenge ist als diskrete Verteilung mit folgende Variablen und Wahrscheinlichkeiten definiert :
  - Variablen : 3,5,8,7,6 mit respektive Wahrscheinlichkeit
  - 0.15,0.25,0.4,0.9,1.0
- DISC(0.15,3,0.25,5,0.4,8,0.9,7,1.0,6)

k. Simulationslauf:
Im Run Setup werden unter "Project Parameters" Projektparameter für den Simulationslauf festgelegt:
  - Warm-up Period : 0.0 (Einschwingphase)
  - Replication Lengh: 2400 (Dauer eines Simulationslaufes)
  - Time Units : Hours (Zeiteinheit des Simulationslaufes)
  - Base Time Units : Hours

I. Statistik Erfassungen:
Folgende Statistik-Werte werden bei der Simulation ermittelt:

- Serviceleistungen beim Supply-TB (Supplier Service Level Customers Service)
- Durchnittlicher Lagerbestand beim Supply-TB(Supplier Average Inventory)
- Lagerumschlagshäufigkeit (Production == 1)
- Lagerhaltungskosten
- Prozentsatz von entgangenen Aufträge

Statistic - Advanced Process

	Name	Type	Expression	Report Label	Output File	Frequency Type	Resource Name	Report Label	Output File	Categories
1	A. Supplier Average Inventory	Time-Persistent	Supplier_Inventory	A. Supplier Average Inventory		Value		A. Supplier Average Inventory		0 rows
2	B. Process State	Frequency		B. Process State		State	Service Operator	B. Process State		0 rows
3	C. Production On	Time-Persistent	Production==1	C. Production On		Value		C. Production On		0 rows
4	D. Lost Percentage	Output	Lost/Total Customers	D. Lost Percentage		Value		D. Lost Percentage		0 rows
5	F. Supplier Customers Service Level	Output	AErfuellteAuftraege/ANEAuftraege	F. Supplier Customers Service Level		Value		F. Supplier Customers Service Level		0 rows

Abbildung 49: Supplier-Module Statistik-Messungen

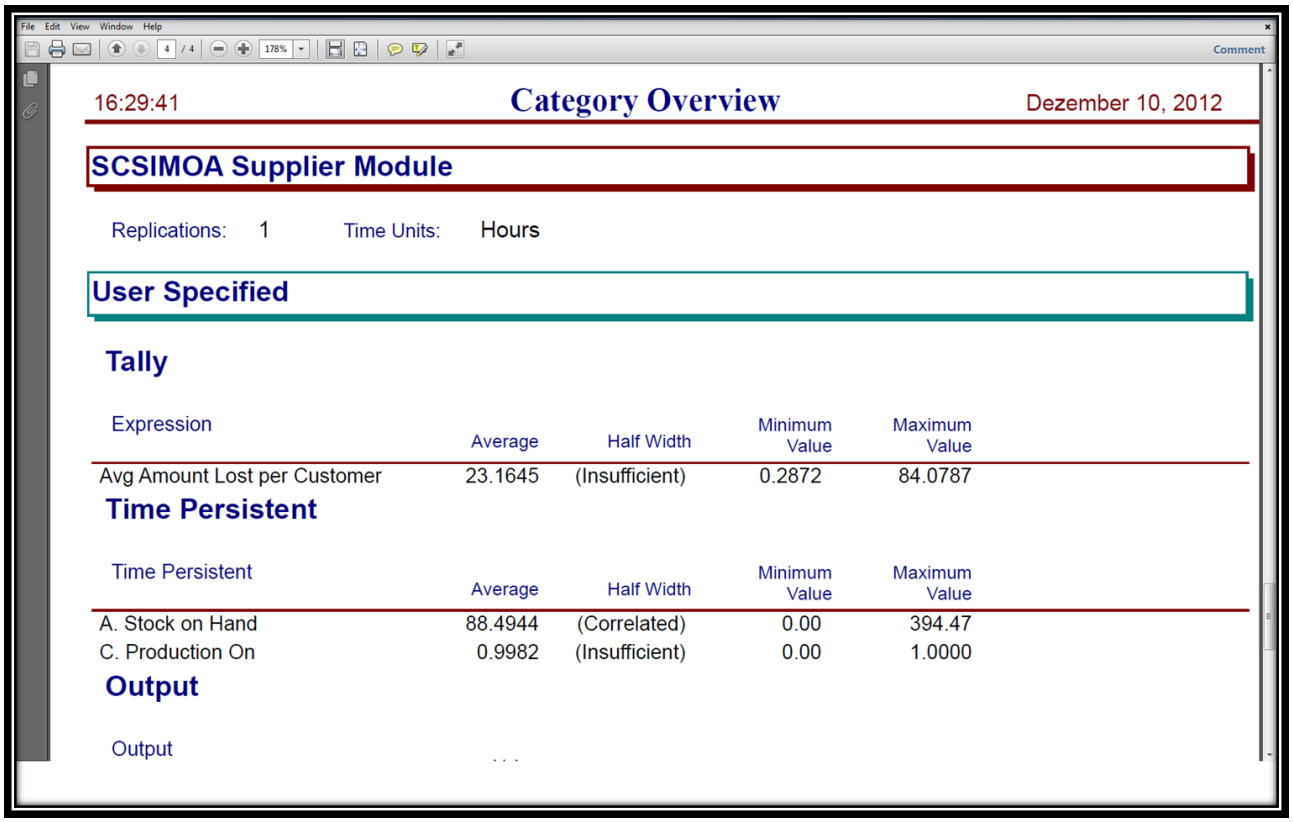


Abbildung 50: Supplier User Specified Messungen

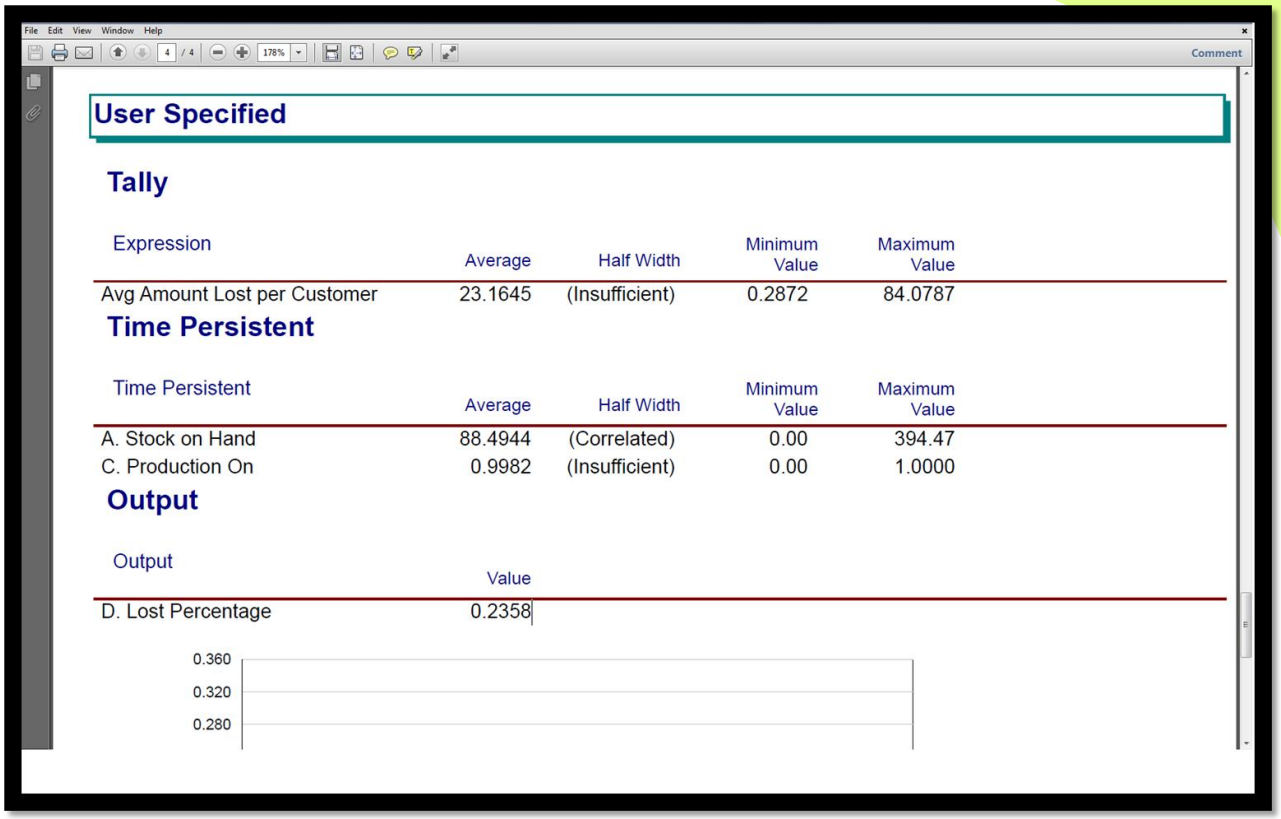
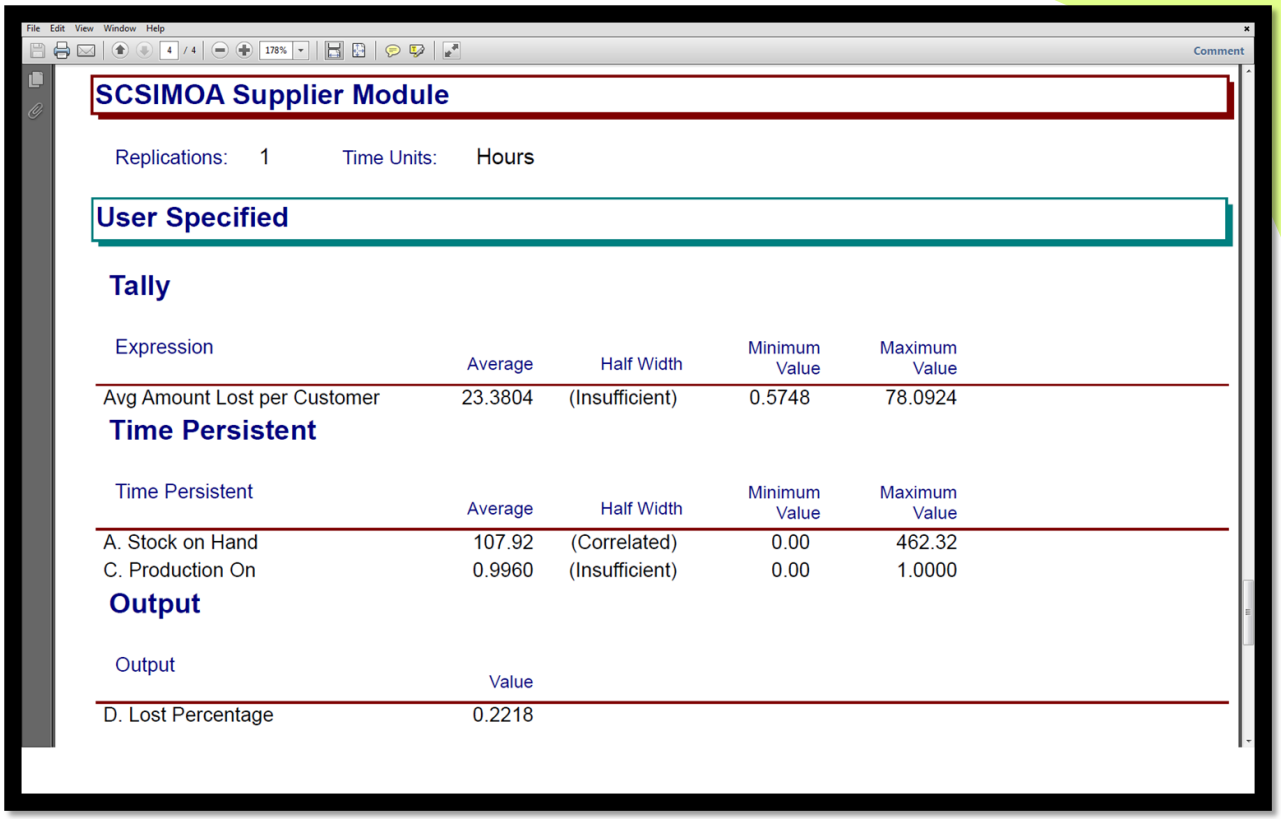


Abbildung 51: Supplier Tally Messungen

Der Simulationsreport zeigt in der Benutzerspezifische Sektion folgende Werte an:

- Customer Service Level beim Supply-TB beträgt 76.5% in der "Output"Sektion, da wir 23.5% an entgangene Aufträge haben.
- Der mittlere Wert für den Lagerbestand beim Supply-TB liegt bei 88 in der „Time-Persistent" Sektion
- Die „Tally"-Sektion gibt die Statistik-Werte im Record-Module an. Die Variable „Avg Amount Lost per Customer" beträgt 23%. Dies gibt Anzahl entgangene Aufträge pro Kunden an.

Nach einer Analyse der Ergebnisse der Simulation, sind wir interessiert den Customer Service Level beim Supply-TB zu verbessern. Da wir mit einem Inventory-Oriented System arbeiten, können wir Serviceleistungen erhöhen indem wir den Startbestand beim Supply-TB erhöhen. Wir simulieren das System mit dem Starbestand I(t) = 350 statt I(t = 250). Der neue Simulationsreport (Abbildung 32) zeigt eine Änderung für den Wert von der Variable „D. Lost Percentage" in der Output-Sektion an. Der Prozentsatz von entgangenen Aufträgen ist von 23.5% auf 22.1% gesunken. Daraus folgt eine Erhöhung von der Serviceleistung von 76.5% auf 77.9%.



SCSIMOA Supplier Module

Replications: 1 Time Units: Hours

User Specified

Tally

Expression	Average	Half Width	Minimum Value	Maximum Value
Avg Amount Lost per Customer	23.3804	(Insufficient)	0.5748	78.0924

Time Persistent

Time Persistent	Average	Half Width	Minimum Value	Maximum Value
A. Stock on Hand	107.92	(Correlated)	0.00	462.32
C. Production On	0.9960	(Insufficient)	0.00	1.0000

Output

Output	Value
D. Lost Percentage	0.2218

Abbildung 52: Supplier Serviceleistungen verbessert

4. SCSOA Supply Chain Modell

Um ein Supply Chain Simulationsmodell mit der SCSOA-Bibliothek aufzubauen, müssen die alle vier Module in Arena importiert werden. Die Module können mit dem Befehl File > Template Panel > Attach. Das Ziel der Bachelorarbeit ist damit erreicht.

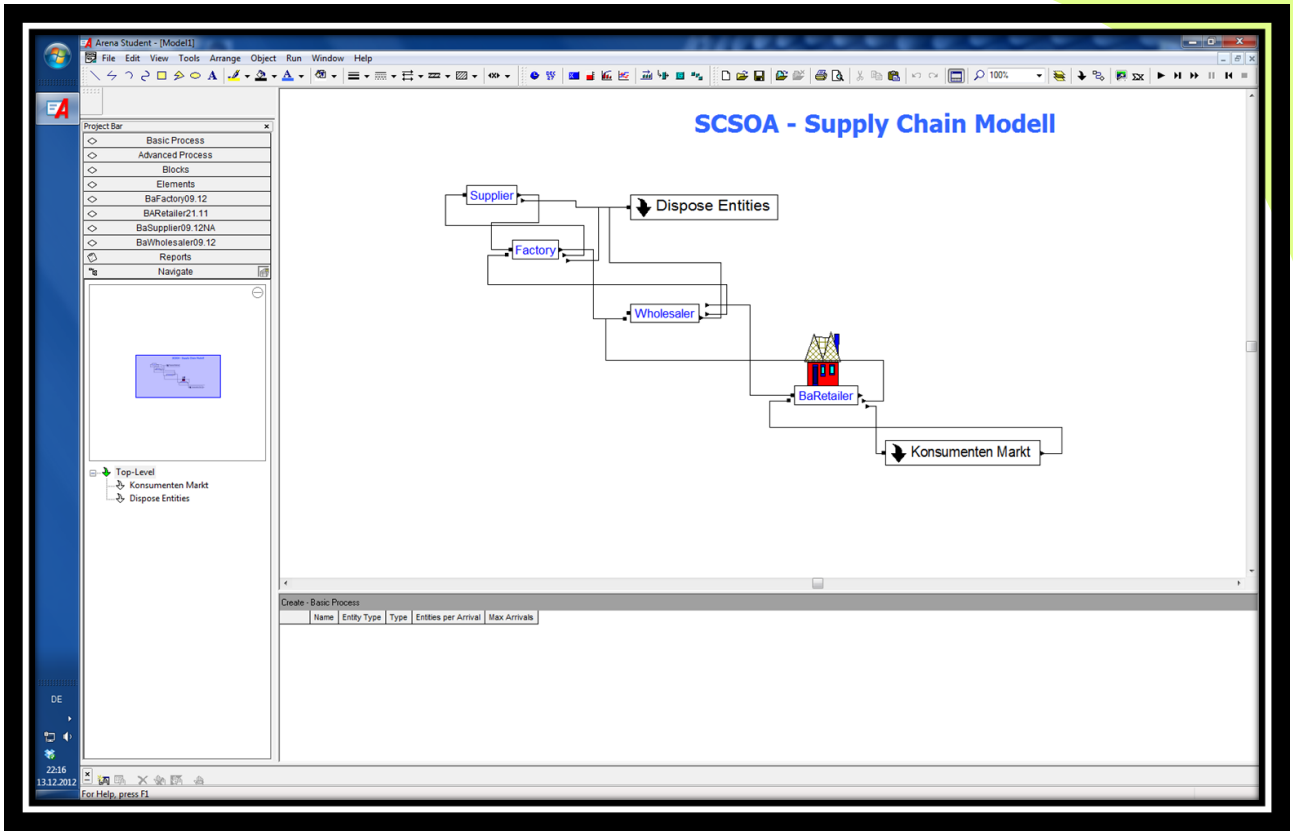


Abbildung 53: SCSOA Supply Chain Modell

5. Schlussfolgerung

Mit dem Arena Simulationstool können sehr komplexe Systeme nachgebildet und analysiert werden. Die Möglichkeit eine eigene Bibliothek in Arena zu entwickeln, kann die Aufgabe eines Systeme-Analytikers in viele Hinsichten erleichtern. Die in diese vorliegende Arbeit entwickelte Bibliothek soll eine der Vielen Funktionalitäten von Arena zeigen.