Thema:

Eine Kanbansimulation mit SeSAm

Ausarbeitung
im Rahmen des Seminars Agenten in simulierten Umgebungen

im Fachgebiet Informatik
am Institut für Informatik

Themensteller/
Betreuer: Dr. Dietmar Lammers

vorgelegt von:
Jakob Acar
Margaretenstraße 15
48145 Münster
jakobus@uni-muenster.de

Philipp Thiemann
Stiftsstraße 19
48301 Nottuln
phithi@uni-muenster.de

Abgabetermin: 2005-01-09

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Motivation für eine Kanbansimulation

Die zugrunde liegende Simulationsumgebung SeSAm (Shell für Simulierte Agentensysteme) zeichnet sich dadurch aus, dass Multiagentensysteme erfolgreich umgesetzt werden können.[1] Aus dieser Tatsache heraus liegt es nahe eine Simulation zu realisieren, die mehrere Entitäten beinhaltet, welche miteinander agieren. Die Vielzahl der Entitäten sollte durch die Simulation möglichst realitätsnah in Einklang gebracht werden können. Bei der Überlegung in welchem Bereich eine Vielzahl von Entitäten vorhanden ist, die koordiniert werden muss und gleichzeitig verschiedene Merkmale hat, lag die Idee nahe, beeinflusst durch diverse Vorlesungen im Studium, im Bereich der Produktion ein Produktionsverfahren zu simulieren.[2] Die bereits mehrfach angesprochenen Entitäten werden in der Simulation als Agenten in einem Multiagentensystem koordiniert. Die in einem Produktionsverfahren involvierten Entitäten (Agenten) sind beispielsweise Kunden, Arbeiter, Betriebsmitteln (Maschinen) und Werkstoffe.[3] Allen Produktionsverfahren ist gemein, dass sie diese Entitäten beinhalten.

Das Produktionsverfahren, welches hier als Grundlage dient, ist das Kanban-Produktionsverfahren. Wieso Kanban aus einer Vielzahl von Produktionsverfahren ausgewählt wurde ist zunächst mit einem Hinweis auf die zugrunde liegende Simulationsumgebung SeSAm zu erklären. Durch das Arbeiten mit SeSAm wird ein Aspekt sehr schnell deutlich und zwar die gute Anschaulichkeit der Simulationen durch die grafische Unterstützung der Simulationsumgebung. Da das Kanban-Produktionsverfahren ebenfalls durch seine Anschaulichkeit, was im Weiteren durch Schaubilder deutlich wird, besticht, lag es nahe das Kanban-Produktionsverfahren zu simulieren.

---

1 Vgl. Klügl (2001), S. 141.
2 Vgl. Vorlesung: Produktion, Vorlesung: Produktionsplanung und -steuerung.
3 Werkstoffe setzen sich aus Rohstoffen, Hilfsstoffen und Betriebsstoffen zusammen.

2 Kanban als logistikdeterminierte Produktionssteuerung

2.1 Einführung in Kanban

Im Zusammenhang mit dem Kanban-Produktionsverfahren kommt schnell die Frage auf, was Kanban überhaupt bedeutet. Der Begriff Kanban stammt aus dem japanischen und bedeutet Karte, Schild oder Zettel.

Abb. 1: Beispiel einer Kanban

Abb. 1 zeigt eine solche Karte. Sie enthält Informationen über die Teilenummer und den Namen des Teils, die Behälterart, die Standardfüllmenge je Behälter, die Herkunft der Teile (Quelle), die Adresse der Teile (Senke) und die Registriernummer der Karte. Das Verfahren der Produktionssteuerung wurde in den sechziger Jahren von Toyota in Japan entwickelt. Der Grund dessen Entstehung war ein erhöhter Marktdruck, dem es entgegenzuwirken galt. Zu jener Zeit war es kennzeichnend für japanische Unternehmen, dass sie eine geringe Kapitalausstattung hatten. Insofern lag die Idee nahe, Kapital, welches in Form von Materialien in Lagern gebunden ist, zu reduzieren und somit über zusätzliches Kapital zu verfügen. Das Ziel war, durch die Senkung der Lagerbestände, so wenig Kapital wie möglich im Lager zu binden. Betrachtet wurden nicht nur große Hauptlager sondern auch Zwischenlager.[4]

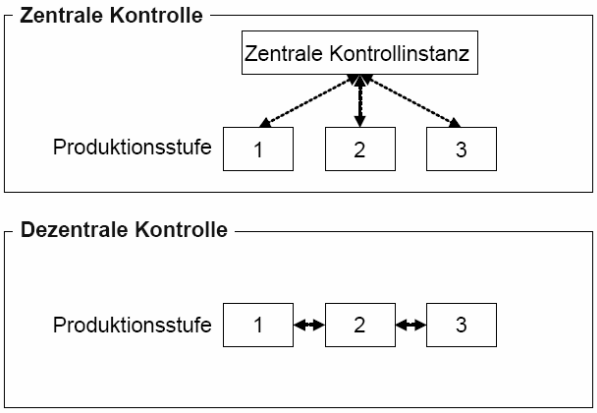

Abb. 2: Zentrale vs. dezentrale Kontrolle

[4] Vgl. Kurbel (1999), S. 174.

Der Hauptgedanke von Kanban ist es, ein System zu installieren, dass aus selbststeuernden Regelkreisen besteht, welche dazu benutzt werden, die Koordination zwischen den Fertigungsstufen, zu übernehmen. Abb. 2 verdeutlicht die Eliminierung einer zentralen Kontrollinstanz beim Kanban-Produktionsverfahren, da dieses sich, als dezentrales Steuerungskonzept, durch die bereits beschriebenen Regelkreise selbst steuert.

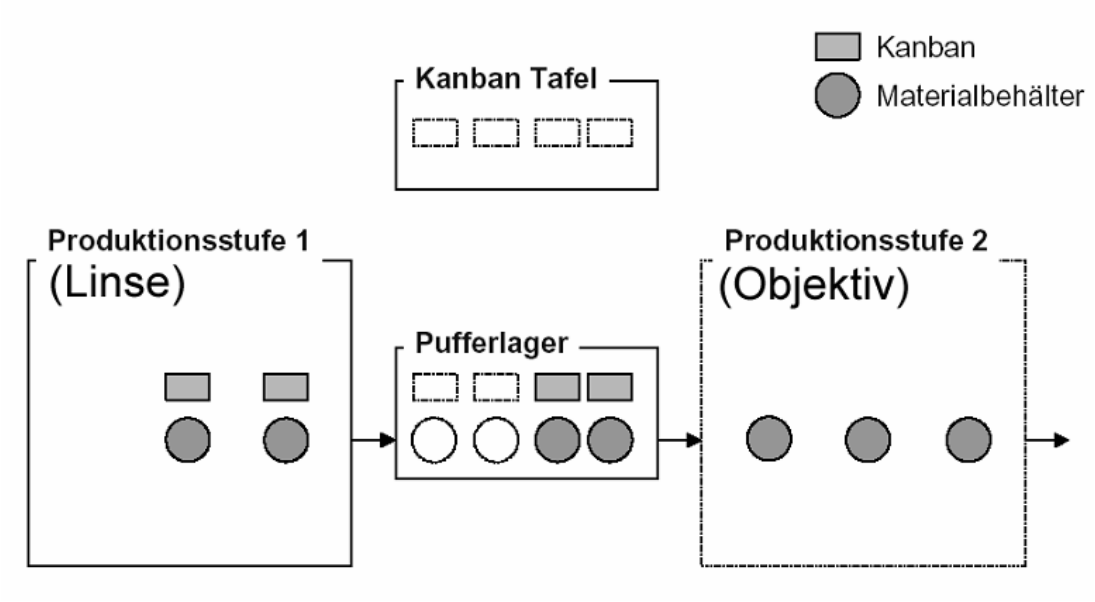

Abb. 3: Das Kanban-Prinzip

Abb. 3 zeigt einen schematischen Überblick über das Kanban Prinzip.[5] Das Kanban-Produktionsverfahren gehört zur Familie der Produktionsverfahren der Auftragsfertiger, d.h. es wird auf Bestellung der Kunden produziert und nicht wie bei Lagerfertigern auf Grundlage von Prognosen.

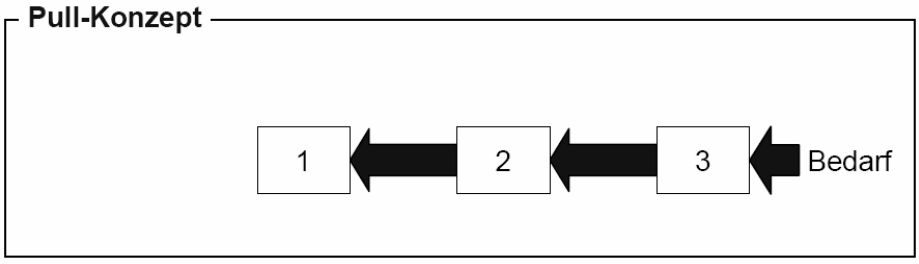

Abb. 4: Das Pull-Konzept

Daher handelt es sich bei Kanban um ein Produktionsverfahren, das nach dem in Abb. 4 dargestellten Pull-Prinzip arbeitet.

---

[5] In Anlehnung an: Kurbel (1999), S. 174.

Zwischen zwei Produktionsstufen befindet sich jeweils ein Pufferlager mit einem definierten Mindestbestand. Eine vorgelagerte Produktionsstufe wird als Materialquelle bezeichnet und eine nachgelagerte als Materialsenke. Bei einer mehrstufigen Produktion besteht die Möglichkeit, dass eine Produktionsstufe Materialquelle und Materialsenke zugleich ist. Materialquelle und Materialsenke bilden, mit dem Pufferlager als Verbindung, die bereits erwähnten selbststeuernden Regelkreise. Zu Beginn produzieren die Materialquellen so viel, dass die Pufferlager gefüllt sind. In den Pufferlagern befinden sich anschließend die Materialbehälter, welche eine fixe Anzahl an Teilen beinhalten, und die dazu gehörigen Kanbans. In Falle der Entnahme eines Behälters aus dem Pufferlager, wandern der Materialbehälter zu der Materialsenke und die Karte auf die Kanban-Tafel. Wenn eine Materialquelle freie Kapazitäten besitzt, fordert sie eine Karte von der Kanban-Tafel. Sobald die Karte in der Materialquelle angekommen ist, beginnt die Produktion neuer Teile. In der Zwischenzeit befindet sich der Materialbehälter in der Materialsenke solange er nicht leer ist. Ist er geleert, wandert er in die Materialquelle, in der sich seine zugehörige Karte befindet, und wird wieder befüllt. Anschließend wandert der Materialbehälter wie zu Beginn mit der Karte in das Pufferlager und wartet darauf von einer Materialsenke angefordert zu werden. Besonders deutlich wird hierdurch der entgegengesetzte Fluss der Informationen und des Materials. Während der Informationsfluss durch Bestellungen am Fertigwarenlager startet und womöglich bis zum Rohstofflager durchgereicht wird, verhält es sich beim Materialfluss entgegengesetzt. Hier startet der Materialfluss offensichtlich im Rohstofflager und mündet in der letzten Materialsenke, d.h. dem Fertigwarenlager.

2.1.1 Ziele

Die Ziele, die mit dem Einsatz des Kanban-Produktionsverfahren verfolgt werden, sind zum einen die Minimierung der Lagerbestände, die Senkung der Durchlaufzeit pro Auftrag und schließlich die Befriedigung aller Nachfragen. Das Hauptziel, dass mit dem Einsatz des Kanban-Produktionsverfahrens verfolgt wird, ist eine optimale Konfiguration von Kanbans bzw. Behältern zu realisieren, die dazu führen das die Kapitalbindung gesenkt wird. Hierbei ist das Ziel die Minimierung der Anzahl der Karten. Dafür wird die in der Literatur[6] verwendete Kanban-Formel benutzt:

$$\text{Kanbans} = \frac{(\text{Bedarf je Tag} \cdot \text{Wiederbeschaffungszeit je Los}) + \text{Sicherheitsfaktor}}{\text{Standardanzahl der Teile je Behälter}}$$

Der Bedarf je Tag ermittelt sich aus den Nachfragen. Die Wiederbeschaffungszeit ist als die Zeit definiert, die ein Behälter benötigt, um nach einer Entnahme aus dem Pufferlager

[6] Vgl. Adam (1998), S. 630.

wieder gefüllt im Pufferlager einzutreffen. Der Sicherheitsfaktor muss, je größer zufällige Schwankungen sind, umso größer gewählt werden. Er sollte allerdings möglichst gering sein. Die Standardanzahl der Teile im Behälter ist vorgegeben und ändert sich nicht.

2.1.2 Regeln

Die Materialsenke darf unter keinen Umständen mehr Material anfordern als sie benötigt und ebenso auf keinen Fall Material, welches benötigt wird, vorzeitig anfordern. Nur so kann realisiert werden, dass sich die Produktion dynamisch an die Nachfrage anpasst und indirekt die Lagerbestände gering bleiben. Des Weiteren wird jeweils immer nur aus einem Behälter entnommen, d.h. es wird nicht aus mehreren Behältern simultan entnommen. Umgekehrt darf eine Materialquelle nicht mehr Teile herstellen als tatsächlich angefordert sind und es darf auch nicht produziert werden, wenn keine Bestellung vorliegt. Um den Materialfluss möglichst nicht zu unterbrechen, ist davon auszugehen, dass die Teile, welche sich in den Behältern befinden, 100 % fehlerfrei sind, somit muss die Qualitätsüberwachung es bewerkstelligen, dass fehlerhafte Teile im Vorfeld entdeckt und eliminiert werden[7].

2.1.3 Voraussetzungen

Um das Kanban Prinzip erfolgreich umsetzen zu können ist es erforderlich bestimmte Rahmenbedingungen zu schaffen. Das Kanban-Produktionsverfahren setzt einen weitgehend regelmäßigen Teilebedarf in allen selbststeuernden Regelkreisen[8] voraus. Es muss gewährleistet sein, dass genügend Betriebsmittel[9] vorhanden sind, die möglichst geringen Schwankungen unterliegen, was ihr Produktionsangebot betrifft. Maßnahmen die dazu führen, dass das Produktionsangebot geringe Schwankungen aufweist, sind eine vorsorgliche Durchführung der Wartung und Instandsetzung und die Schaffung von Kapazitätsreserven. Eine weitere sehr wichtige Voraussetzung ist, zu garantieren, dass die Wiederauffüllzeit möglichst kurz ist und im Zeitablauf sehr geringen Streuungen unterliegt. Die Voraussetzung dieser Größe, welche Bestandteil der Kanban-Formel[10] ist, ist wiederum nur dann zu erreichen, wenn ein konstantes Produktionsangebot bzw. Kapazitätsangebot vorhanden ist. Positiv wirkt sich ebenfalls eine Anordnung der Fertigung nach dem Materialfluss aus.

---

7 Vgl. Adam (1998), S. 631.

8 Materialquellen und Materialsenken bilden verknüpft durch die jeweiligen Pufferlager, selbststeuernde Regelkreise.

9 z.B. Maschinen.

10 Vgl. Abschnitt 2.1.1.

2.2 Beschreibung unseres Szenarios

In unserem Beispiel werden mittels des Kanban-Produktionsverfahrens Ferngläser produziert, welche einer Nachfrage unterliegen, die einer Normalverteilung mit den Parametern 15 als Erwartungswert und 5 als Varianz genügt. Ein Fernglas besteht aus einem Gehäuse und zwei Objektiven, welche jeweils aus drei Linsen zusammengesetzt sind. Wir gehen davon aus, dass von den Teilen, die zur Fertigung der Linsen und Gehäuse benötigt werden, unendlich viele vorhanden sind. Es kann davon ausgegangen werden, dass diese extern und „Just-in-Time“ beschafft werden.

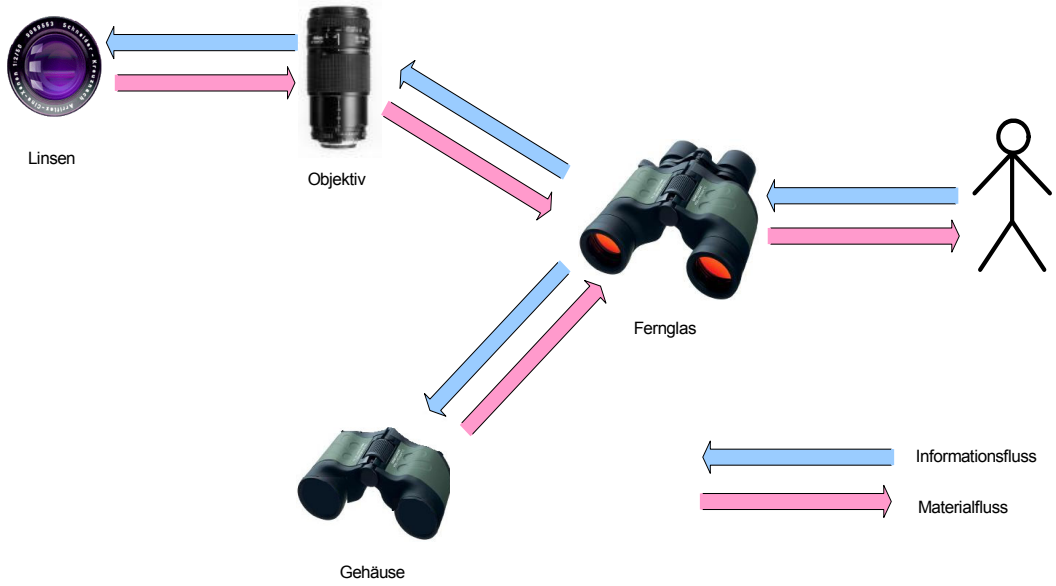

Abb. 5: Fernglasproduktion mit Kanban

Aus Abb. 5 wird ersichtlich, dass der Materialfluss nach den Fertigungsstufen angeordnet ist, um so eine geringere Wiederauffüllzeit zu realisieren. Der Informationsfluss verläuft wie bereits geschildert entgegengesetzt und realisiert so eine dynamische Produktion, die auf der Nachfrage basiert. Die Pufferlager sind in der Abbildung nicht gesondert gekennzeichnet, sie befinden sich zwischen den Produktionsstätten. Um ein konstantes Angebot der Produktion gewährleisten zu können ist es in unserem Modell so realisiert, dass die Anzahl Betriebsmittel, abhängig von der Nachfrage, in diskreten Zeitabständen angepasst werden bis eine Konfiguration erreicht wird, die keine Anpassung mehr erfordert. In diesem Zusammenhang ist darauf hinzuweisen, dass Kosten in unserem Modell ausgeblendet sind, da die sinnvolle Bestimmung von Kosten eine praktische Untersuchung in Unternehmen voraussetzen würde (z.B. Konventionalstrafen[11]).

[11] Sanktionierung des Nichteinhaltens bestehender Verträge bzw. deren verspätete Erfüllung.

3 Umsetzung in SeSAm

Für die Umsetzung unserer Kanbansimulation haben wir das *Unternehmen* als Weltagenten implementiert, da aus Sicht der Produktion das Unternehmen die Umwelt darstellt. Des Weiteren sind Agentenklassen vorhanden. Zur groben Unterscheidung lassen sich diese in *Produktionsstätten*, *Kanban*(-Behälter) und die Markt*nachfrage* einteilen. Gemäß der bereits vorgestellten Produktionssituation, sind die Produktionsstätten für die einzelnen Produktionsaggregate (Linsen, Gehäuse, …) und für den Teiletransport zwischen diesen die Kanbanbehälter zu unterscheiden.[12]

3.1 Der Weltagent

Dem Weltagenten kommt, wie dem Unternehmen in der Realität, eine koordinierende Rolle zu. Das Unternehmen hält eine Reihe von Warteschlangen (*List<SimObject>*) bereit und bietet den Agenten somit ein Koordinationsgerüst, auf das sie im Rahmen ihrer Aktivitäten zugreifen können. Im Übrigen hält der Weltagent eine Reihe von Parametern vor, die für die Konfiguration der Produktion zuständig sind. Dazu zählen die Behältergröße, die Anzahl der Teile, die zur Produktion eines nachgelagerten Teils notwendig sind, oder die Anzahl der Teile, die in einem Zeittick von einer Produktionsanlage hergestellt werden. Zuletzt werden einige Variablen bereitgestellt, die zur Anpassung der Produktion oder auch zur Auswertung der Simulation benötigt werden. Beispielhaft lassen sich hierbei die maximale Produktionsmenge einer Anlagenklasse oder die kumulierte Marktnachfrage nennen.

Das Unternehmen erzeugt mit *RandomBoolean(0,025)* neue Aufträge mit einer normalverteilten Nachfragemenge ($N(15;5)$). Alle 3000 Ticks überprüft der Weltagent die Anzahl der Produktionsagenten. Zu unterscheiden ist hierbei die Überprüfung der Kanbans und die Überprüfung der Produktionsstätten:

Überprüfung der Anzahl der Kanbanagenten

Mit Hilfe der SeSAm-eigenen *User Functions* haben wir die in Kapitel 2.1.1 erwähnte „Kanbanformel“ hinterlegt. Diese legt die Verrechnung der Parameter fest. Angepasst für jeden Kanbantyp haben wir außerdem jeweils eine User Function mit den konkreten Parameterzuordnungen erstellt. Diese Funktionen geben bei Ausführung den Sollwert der Kanbananzahl zurück.

---

[12] Alternativ ließe sich auch ein einheitlicher Agententyp definieren, dem nach der Initialisierung seine Bezugspunkte zuzuweisen wären. An dieser Stelle sei darauf verwiesen, dass dies eine überaus gesteigerte Komplexität des Systems zur Folge hätte.

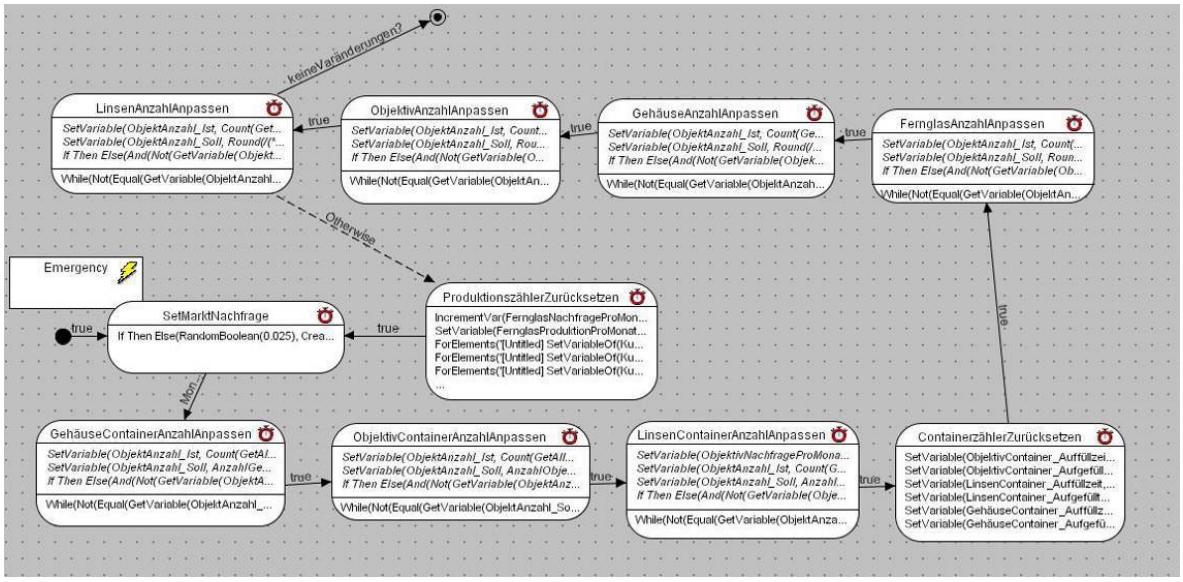

Abb. 6: Reasoning-Engine der Welt

Dabei berechnet sich der Bedarf pro Zeittick aus der Nachfrage in dem Intervall, geteilt durch die Dauer, seit der letzten Überprüfung, in Ticks. Die Wiederauffüllzeit berechnet sich aus eigens in der Welt mitgeführten Variablen, die von den Kanbanagenten geschrieben werden, während diese aufgefüllt werden. Wie bereits erwähnt, ist auch die Behältergröße als Variable in dem Weltagenten vorhanden. Zusätzlich haben wir einen Behälter als Sicherheitspuffer einberechnet.

Die Sollanzahl wird verglichen mit der tatsächlichen Anzahl von Kanbanagenten einer Klasse. Anhand des Vergleichs wird entschieden, ob weitere Agenten des Typs erzeugt werden (Positionierung im Lager mit *BeamTo<Position>*) oder vorhandene zerstört werden müssen. (Dazu wird dem jeweiligen Kanbanagenten ein entsprechender Booleanwert auf *true* gesetzt, der die weitere Zerstörung veranlasst.) Ergibt sich bei allen Kanbanagenten keine Veränderung der Anzahl, wird der Simulationslauf beendet, indem die Welt in ihren Endzustand wechselt.

Überprüfung der Anzahl der Produktionsagenten

Eine Prämisse der kanbangesteuerten Produktion war, dass für ausreichendes Kapazitätsangebot gesorgt wird. Um dies zu gewährleisten, wird der Quotient aus der gesamten Nachfrage nach dem zu produzierenden Teil und der maximalen Produktionsmenge einer Klasse von Produktionsanlagen gebildet. Dieser Quotient ergibt die benötigte Anzahl von Produktionsanlagen, um die Nachfrage in Hinblick auf die Kapazität befriedigen zu können und, um im Weiteren die Sinnhaftigkeit der Kanbanformel beurteilen zu können. Analog erfolgen hier die Überprüfung mit der tatsächlichen Anzahl und die Anpassung. Lediglich die Position wird innerhalb eines bestimmten Radius, um die Standardposition der

Produktionsstättenklasse herum, zufällig gewählt. Dazu wird mittels einer Zufallszahl von -3 bis 3 die Abweichung von den Standardkoordinaten des Produktionstyps bestimmt.

3.2 Die Produktionsstätte

Eine Produktionsstätte verfügt über einen Verweis auf einen Auftrag vom Typ *SimObject*. Darüber hinaus enthält ein Produktionsagent eine Variable, die die aktuelle *Nachfrage*menge wiedergibt, sowie eine Variable mit der Anzahl der bereits *produziert*en Teile. Die Anzahl der für die Produktion benötigten und vor Ort befindlichen Teile wird durch eine *TeileVorOrt*-Variable festgehalten. Da eine Produktionsanlage nur aus einem Behälter entnehmen darf, soll sie auch nur einen anfordern können. Dies gewährleistet ein Booleanwert, der festhält, ob bereits ein *Behälter angefordert* wurde, der noch nicht angekommen ist. Die bereits angesprochene *Abriss*-Variable dient der Zerstörung eines Agenten. Eine Variable die die *gesamte Produktion* im Aktualisierungsintervall kumuliert hat den Zweck, die bereits erwähnte maximale Produktionsmenge eines Anlagentyps zu bestimmen.

Bezüglich seiner Aktionen hat der Produktionsagent fünf Verhaltensweisen:

- **Aufträge annehmen**: Wenn kein Verweis auf einen Auftrag besteht, entnimmt der Produktionsagent das erste Element aus der Warteschlange für seinen Anlagentyp, sofern vorhanden und übernimmt dieses als seinen Auftrag. Im Falle eines Behälterauffüllauftrags wird außerdem der Behälter über seine Auffüllanlage benachrichtigt.

- **Aufträge abschließen**: Übersteigt die Anzahl der produzierten Teile die Nachfragemenge, wird der Auftrag abgeschlossen. Darunter ist für die Endnachfrage das Setzen eines Erledigungszeitpunktes und für die Behälter das Auffüllen der Teile zu verstehen.

- **Produzieren**: Sind ein Auftrag und Teile zur Produktion vorhanden, werden die Teile vor Ort um den Produktionskoeffizienten verringert und die Anzahl der produzierten Teile um eins erhöht.

- **Teile anfordern**: Befinden sich keine Teile mehr vor Ort, jedoch besteht ein zu bearbeitender Auftrag, schreibt sich die Produktionsanlage in die Warteliste der vorgelagerten Kanbanbehälter. Um nicht erneut anzufordern – die Zeit bis zum Eintreffen eines Behälters kann bei nicht optimaler Behälteranzahl sehr lange dauern – wird der besagte Booleanwert *TeileAngefordert* auf *true* gesetzt.

- **Leerlauf**: sonst

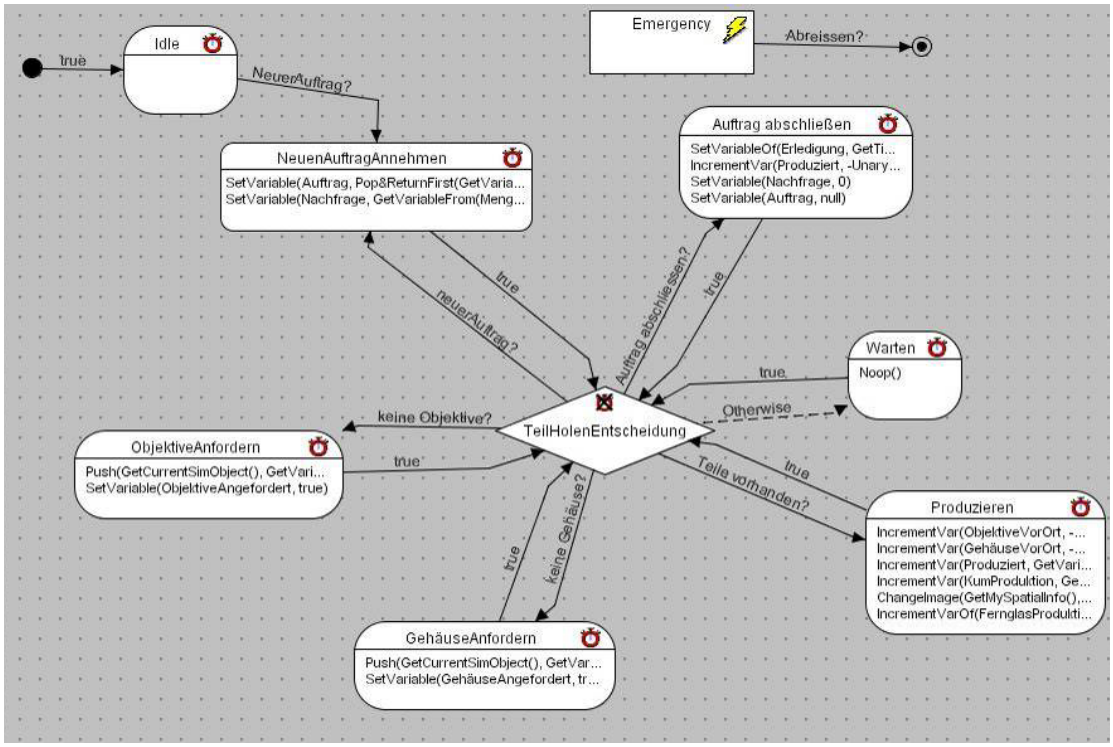

Abb. 7: Reasoning-Engine des Produktionsagenten

Um jederzeit in jede Aktivität wechseln zu können, sind alle Aktivitäten mit einem Entscheidungsknoten verbunden. Dieser konsumiert keine Zeit und nach jeder ausgeführten Aktivität wird dieser Knoten durchlaufen, so dass in jedem Tick eine Überprüfung des Verhaltens erfolgt.

Ist der Abriss-Wert von der Welt auf *true* gesetzt worden, wird dies von dem Emergency-Knoten registriert. Dieser prüft vor der weiteren Ausführung des Agenten die von ihm ausgehende Regel in jedem Zeitstep ab. Bei Erfüllung der Bedingung begibt sich der Agent in den Endzustand.

Die Produktionsstätten sind allerdings nicht so einheitlich, wie es den Anschein hat. Neben den unterschiedlichen Bezugspunkten, die sich auch bei Erzeugung der *Situations* anwählen lassen könnten, weisen vor allem die Anfangsproduktionen (Linsen, Gehäuse) unterschiede zu den übrigen Produktionsstätten auf. Da der benötigte Warentransport zu ihnen nicht mit Hilfe von Kanbanbehältern erfolgt, wird davon ausgegangen, dass die benötigten Teile „Just-in-Time“ geliefert werden. Daher unterstellen wir eine hinreichend große Zahl an vorhandenen Teilen. Für die Fernglasproduktion ist anzumerken, dass das Produkt aus zwei Teilen erstellt wird. Aus Sicht des OO-Programmierparadigma entspricht unsere Lösung nicht dem Prinzip der Vererbung. Eine Realisierung dieser Art blieb uns jedoch wegen der fehlenden API-Dokumentation versagt.

3.3 Der Kanban(-Behälter)

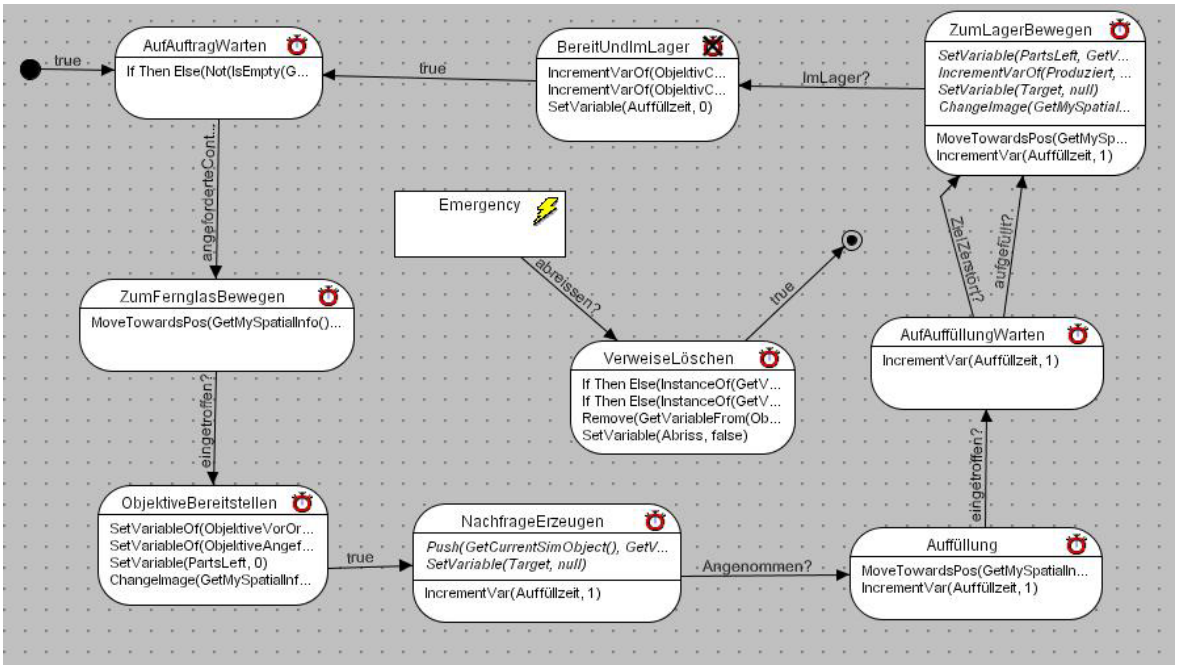

Abb. 8: Reasoning-Engine des Kanbanagenten

Zu diesem Agententyp sei vorab zu erwähnen, dass von uns ein einheitlicher Agent verwendet wird, jedoch im Allgemeinen zwischen Kanban und Behälter unterschieden wird. Außerdem muss in der Praxis auch nicht eine feste 1:1-Verbindung zwischen Kanban und Behälter existieren. Derlei realistische Annahmen hätten jedoch das System sehr viel komplizierter und wartungsunfreundlicher gemacht.

Kanbanagenten verfügen über eine *Kapazität*, sowie eine tatsächliche *Anzahl* von befindlichen Teilen. Die *Auffüllzeit* wird über die Ticks, in denen der Behälter von Entnahme bis Auffüllung wartet, summiert. Ein *Verweis* auf ein SimObject signalisiert dem Behälter zu welcher Produktionsanlage er sich für die Teileentnahme/-auffüllung bewegen muss. Wiederum führt ein *Abriss*-Booleanwert den Agenten in seinen Endzustand.

Der Kanbanbehälter beschreitet einen Kreislauf über seine Aktivitäten. Dies liegt daran, dass dieser Agententyp die Implementierung der Regelkreise darstellt. Ausgehend vom Wartezustand im Pufferlager (Die Position wird im Unternehmen gespeichert [siehe Kapitel 3.1]), prüft ein Kanbanagent, ob sich Anforderungen (Verweise vom Typ *SimObject* auf die anfordernde Produktionsstätte) in der Behälterwarteschlange befinden. Diese Überprüfung erfolgt so lange, bis dies der Fall ist. Daraufhin entfernt der Behälter das erste Objekt aus der Warteliste und setzt dieses als seinen persönliches Ziel. Der Kanbanagent bewegt sich zu seinem Ziel und teilt der Produktionsstätte mit, dass er eingetroffen ist. Nach der Übergabe der Teile aktualisiert er sich selbst, indem er seine tatsächliche Anzahl auf 0 setzt. Der soeben entleerte Behälter meldet nun seinerseits den Bedarf nach seiner Auffül-

lung. In der Theorie erfolgte dies durch die Kanban-Tafel. In unserem Fall schreibt der Kanbanagent sich in die Warteliste für die Produktionsstätte seiner Teile. Von diesem Zeitpunkt an wird die Auffüllzeit des Behälters in Ticks festgehalten. Ist sein Auftrag angenommen worden und ihm mitgeteilt worden, welche Produktionsstätte für seine Auffüllung zuständig ist, bewegt er sich auf dieses Ziel zu. Hat seine Bezugsproduktionsstätte genügend Teile produziert, wird der Behälter aufgefüllt und dieser erhält als Bezugspunkt das für ihn zuständige Pufferlager. In dem Pufferlager angekommen, teilt der Behälter dem Unternehmen mit, wie lange seine Auffüllung gedauert hat, um von hier aus erneut auf Anforderungen zu warten. Besonders zu erwähnen ist, dass ein Behälter dessen Bezugspunkt mit Hilfe des Abriss-Flags vernichtet wurde (isDead<SimObject> = = true), als aufgefüllt ins Lager zurückkehrt. Diese Annahme verhindert eine Verkomplizierung der Prozesse und fällt aufgrund des seltenen Vorkommens nicht ins Gewicht.

Gravierender wäre die radikale Löschung eines Kanbanagenten, da als Folgereaktion eine Produktionsanlage, durch das Warten auf den zerstörten Behälter, ewig blockiert werden könnte. Daher löscht sich ein Kanbanagent vor seiner Zerstörung aus allen Verweisen, sowie aus der Anforderungsliste für seine Auffüllung. Die Abprüfung und Zerstörung erfolgt analog zu den Produktionsagenten über die Emergency-Aktivität. Für die Löschung der Verweise ist eine zusätzliche Aktivität zwischengeschaltet.

3.4 Die Marktnachfrage

Der Nachfrageagent ist vergleichsweise einfach konstruiert, wird jedoch durch das Unternehmen auf Dauer in sehr großen Mengen erzeugt. Bereits erwähnt wurde die Nachfrage*menge* von N(15;5). Ein *Anfang*s- und ein *End*wert dienen der Zeitmessung in Ticks, um eine Auswertung der Bearbeitungszeiten (Differenz der Zeitpunkte) vornehmen zu können.

Bei seiner Erzeugung schreibt sich der Nachfrageagent in die Gesamtnachfrageliste und in die unbearbeitete Nachfrage-Liste. Die unbearbeiteten Aufträge werden von der Fernglasproduktionsstätte abgearbeitet. Ist die Abarbeitung eines Auftrages erfolgt und wurde der Endzeitpunkt des Nachfrageagenten von der Produktionsstätte geschrieben, zerstört sich der Agent selbst. Über die Gesamtnachfrageliste lassen sich nachher Analysen erzeugen.

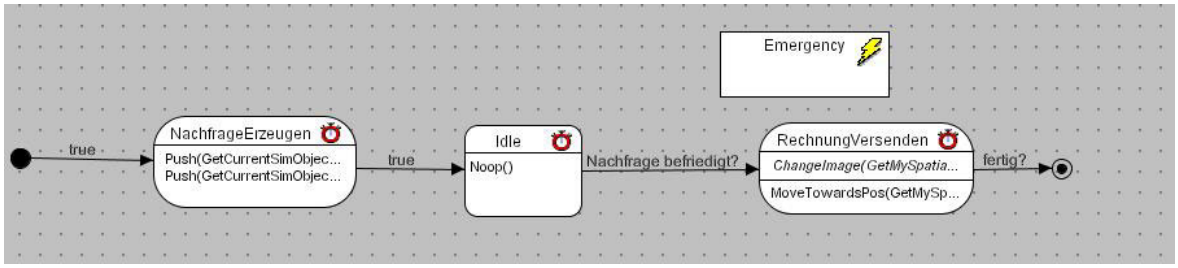

Abb. 9: Reasoning-Engine des Nachfrageagenten

4 Beurteilung der Modellqualität

In diesem Abschnitt werden wir zeigen, dass die theoretischen Ziele, die in der Literatur vorgegeben sind, durch unsere Simulation abgebildet werden können. Das Ziel der Minimierung der Anzahl der Kanbans bzw. Behälter ist dadurch erreicht, dass durch die Einbettung der Kanban-Formel in das Modell, die Dynamik zu einem sinnvollen Ergebnis führt.

In der Simulation wird deutlich, dass die in der Theorie propagierten selbststeuernden Regelkreise tatsächlich dazu führen, dass eine optimale Konfiguration erreicht wird, die nach einer gewissen Anlaufzeit und unter den bestehenden Prämissen, im Zeitverlauf konstant bleibt. Das geschieht durch die Tatsache, dass die Kanban-Formel in das Modell integriert ist, wodurch das Hauptziel, eine minimale Anzahl von Kanbans zu erhalten, durch das Modell in der optimalen Konfiguration realisiert wird. Ebenso ist als Zusatz in unserem Modell anzumerken, dass durch die dynamische Anpassung der Betriebsmittel auch hier im Endzustand ein Optimum der Anzahl von Betriebsmitteln erreicht ist. Die weiteren Ziele, die in Abschnitt 2.1.1 definiert wurden, werden im Weiteren behandelt.

4.1 Senkung der Durchlaufzeit

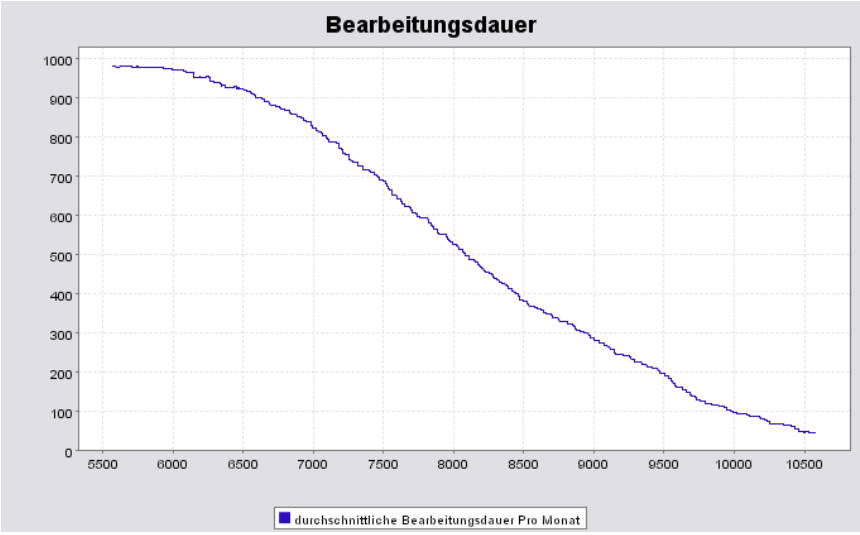

Abb. 10: Senkung der Durchlaufzeit

Abb. 10 zeigt, dass im Endzustand bzw. im Optimum eine durchschnittliche Bearbeitungsdauer einer Kundennachfrage von ca. 50 Ticks erreicht ist. Unter Berücksichtigung der alleinigen Produktionsdauer von 15 Ticks im Mittel, ist dies durchaus ein gutes Ergebnis. In Anbetracht der getätigten Verringerung der Bearbeitungsdauern kann hier durchaus von einem hervorragenden Ergebnis gesprochen werden.

4.2 Befriedigung aller Nachfragen

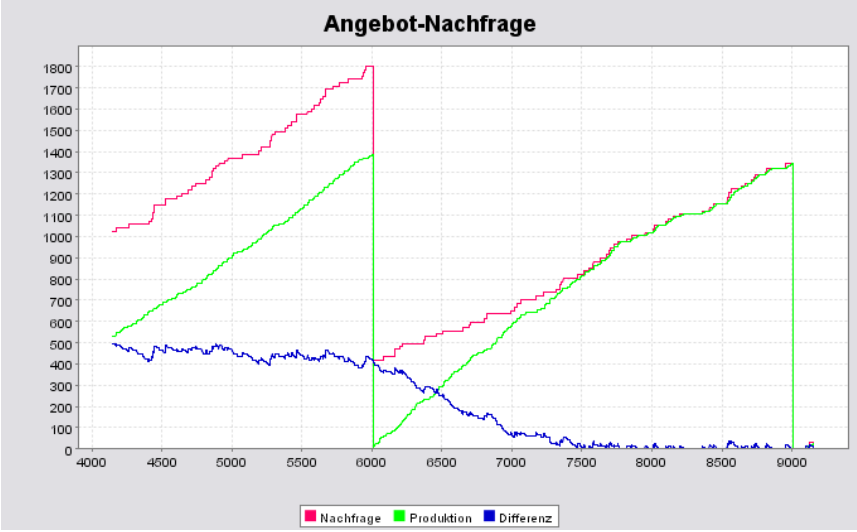

Abb. 11: Befriedigung der Nachfrage

Abb. 11 zeigt, dass im Endzustand bzw. im Optimum die Produktion ohne Schwierigkeiten mit der Nachfrage mithalten kann, so dass sich die Differenzen um den Wert Null bewegen. Die Fehlmenge ist minimiert. Des Weiteren wird hier ganz deutlich, dass die Regel, nicht mehr zu produzieren als benötigt wird, auch eingehalten wird, weil die Kurve der Produktion durchaus steiler verlaufen könnte, es jedoch nicht tut, sondern sich knapp unter der Nachfragekurve befindet.

4.3 Minimierung der Lagerbestände

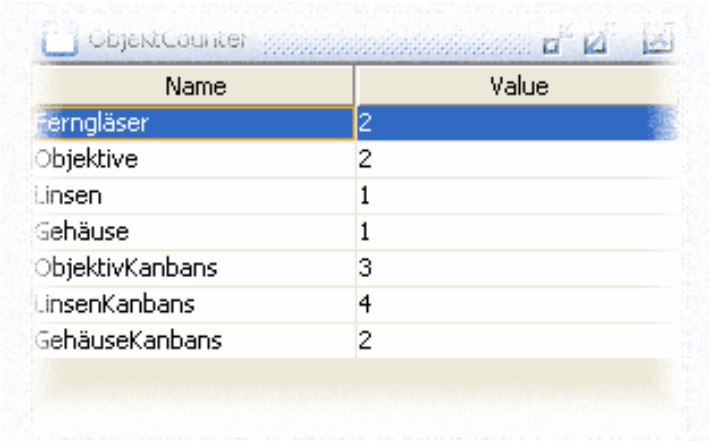

Abb. 12: Anzahl Kanbans im Modell

Abb. 12 zeigt, dass sich im Endzustand bzw. im Optimum eine Konfiguration ergibt, die sehr niedrige Lagerbestände aufweist. Aus unseren Erfahrungen mit verschiedenen Situationen lässt sich sagen, dass das Entfernen eines einzigen Behälters bereits eine erhebliche Verschlechterung der formulierten Ziele erkennen lässt.

4.4 Statistische Auswertung

	A	B	C	D	E	F	G	H
1	Time	Ferngläser	Objektive	Linsen	Gehäuse	ObjektivKanbans	LinsenKanbans	GehäuseKanbans
2	36000	2	2	1	1	3	4	2
3	36000	2	2	1	1	3	4	2
4	36000	2	2	1	1	3	4	2
5	39000	2	2	1	1	3	4	2
6	39000	2	2	1	1	3	4	2
7	39000	2	2	1	1	3	4	2
8	39000	2	2	1	1	3	4	2
9	39000	2	2	1	1	4	5	2
10	42000	2	2	1	1	3	3	2
11	42000	2	2	1	1	4	4	2
12	42000	2	2	1	1	4	5	2
13	45000	2	2	1	1	3	4	2
14	45000	2	2	1	1	3	5	2
15	48000	2	2	1	1	2	4	2
16	51000	2	2	1	1	3	5	2
17	60000	2	2	1	1	4	4	2
18	63000	2	2	1	1	5	4	2
19	63000	2	2	1	1	3	5	2
20	72000	2	2	1	1	3	4	2
21	87000	2	2	1	1	4	4	2
22	123000	2	2	1	1	3	5	2
23								
24								
25	Mittelwert	2	2	1	1	3,285714286	4,238095238	2
26	Standardabweichung	0	0	0	0	0,643650304	0,538958431	0
27								
28								

Abb. 13: Durchführung diverser Testläufe

Im Endzustand ist ein „steady-state“ erreicht, der sich nicht mehr ändern wird, obwohl die Anpassungsmechanismen weiterhin im Modell aktiviert sind, d.h. weiterhin wird überprüft ob die Anzahl der Kanbans optimal ist bzw. ob die Anzahl der Betriebsmittel optimal ist, jedoch wie bereits geschildert ohne weitere Anpassungen. Obwohl die Behälter für Linsen und Objektive größer als für die Gehäuse sind, zeigt sich, dass von diesen Produktionsmitteln mehr Behälter benötigt werden. Dies lässt sich damit erklären, dass es sich bei Objektiven um eine mehrstufige Produktion handelt, in der die Linsen selbst erst bei Bedarf produziert werden. Notwendigerweise müssen daher mehr Behälter vorhanden sein, um die Verzögerungen auszugleichen. Abb. 13 zeigt, dass die angestrebte Standardabweichung der Produktionsstätten bzw. der Kanbans von maximal eins deutlich unterschritten ist, bei 22 Testläufen. Daher war es wenig erstaunlich, dass diese stabil laufende Simulation mit der Eingabe der Optimalwerte als Startkonfiguration zu keinen Veränderungen der Anzahlen führte.

5 Eignung des Systems

Während des Arbeitens mit SeSAm sind uns einige Aspekte aufgefallen, die wir in positive und negative Merkmale unterteilt haben und im Weiteren Vorstellen werden.

Negativ anzumerken ist:

- Wird eine Simulation im Hintergrund gestartet, aus dem Grunde einen zügigen Simulationslauf zu realisieren, um ebenso zügig an dessen Resultate zu gelangen, so lässt die Performanz eines Simulationslaufes erheblich nach, wenn die Simulation einmal visuell gestartet wurde.

- Beim Löschen von Variablen treten erhebliche Schwierigkeiten auf, weil angebliche Referenzen bestehen. Jedoch selbst bei beseitigen dieser Referenzen ist das Löschen teilweise nicht möglich.

- Obwohl das Feature Copy/Paste angeboten wird, ist dieses nur mit Vorsicht zu benutzen, da Verweise teilweise leer eingefügt werden.

- Die Simulationsumgebung ist äußerst prozessorlastig. Das kann jedoch dadurch vermindert werden, in dem grafische Details wie das Grid (Gitternetz) deaktiviert werden. Ein erheblicher Unterschied wird hierbei deutlich.

- Bei der Durchführung von Experiments gehen die Analysecharts verloren.

- Das komfortable Nutzen von Hot Keys., beispielsweise Strg-S zum Speichern funktioniert nur eingeschränkt. Durch Zufall haben wir herausgefunden, dass ein bestimmter Frame aktiviert sein muss.

- Unsere Idee, die Agenten durch Nachrichtenaustausch[13] direkt miteinander kommunizieren zu lassen ging leider nicht, da keine gerichteten Nachrichten möglich sind. Es können vordefinierte Nachrichten (accept, inform,...) benutzt werden, welche an alle Agenten verbreitet werden.

Positiv anzumerken ist:

- Die Simulationsumgebung SeSAm bietet einen intuitiven Umgang durch eine übersichtliche Oberfläche.

---

[13] Vgl. Communication PlugIn. http://ki.informatik.uni-wuerzburg.de/~sesam/download/plugin/CommunicationPlugin.jar. Abrufdatum: 2004-11-05.

- Simulationen werden durch die grafische Unterstützung anschaulich dargestellt.
- Es können die unterschiedlichsten Analysen erstellt werden. Hervorzuheben ist die Aufzeichnung der Analyseergebnisse in einer Zeitreihe (*Series Chart*), die für die Beurteilung der Simulation im Zeitverlauf äußerst nützlich ist.
- Die Simulationsumgebung bietet eine Vielzahl von Funktionen und Datentypen, die häufig unmittelbar intuitiv verständlich sind. Des Weiteren besteht die Möglichkeit selbst Funktionen zu definieren.
- Besonders nennenswert ist die Idee, dass „Simulations" zu „Runs" instanziiert werden, wodurch mehrere gleiche Testläufe durchgeführt werden können. Dadurch können Vergleiche vorgenommen werden.
- PlugIns können leicht in das Model integriert werden. Die *.jar Dateien müssen bloß in das Verzeichnis PlugIn kopiert werden.

Alles in allem bietet die Simulationsumgebung SeSAm die Möglichkeit die verschiedensten Ideen zu realisieren. Nach einer gewissen Anlaufzeit werden immer mehr Funktionen bekannt so dass die Effektivität im Zeitverlauf stetig steigt.

Mit einer gewissen Routine werden auch die als negativ angemerkten Aspekte gerne hingenommen, denn mit SeSAm lassen sich erstaunlich komplexe Sachverhalte abbilden.

Literaturverzeichnis

Adam, D.: Produktions-Management. 9. Aufl., Wiesbaden 1998.

Klügl, F.: Multiagentensimulation. Konzepte, Werkzeuge, Anwendungen. München 2001.

Kurbel, K.: Produktionsplanung und -steuerung. 4. Aufl., München/Wien 1999.