Kanban

- eine Kanbansimulation mit SeSAm -

Von
Jakob Acar & Philipp Thiemann

Gliederung des Vortrags

- Motivation
- Einführung in Kanban
- Beschreibung der Produktionssituation
- Umsetzung in SeSAm
- Simulation und Auswertung
- Soll-/Ist-Vergleich auf Modellebene
- Kritische Auseinandersetzung mit SeSAm

Motivation

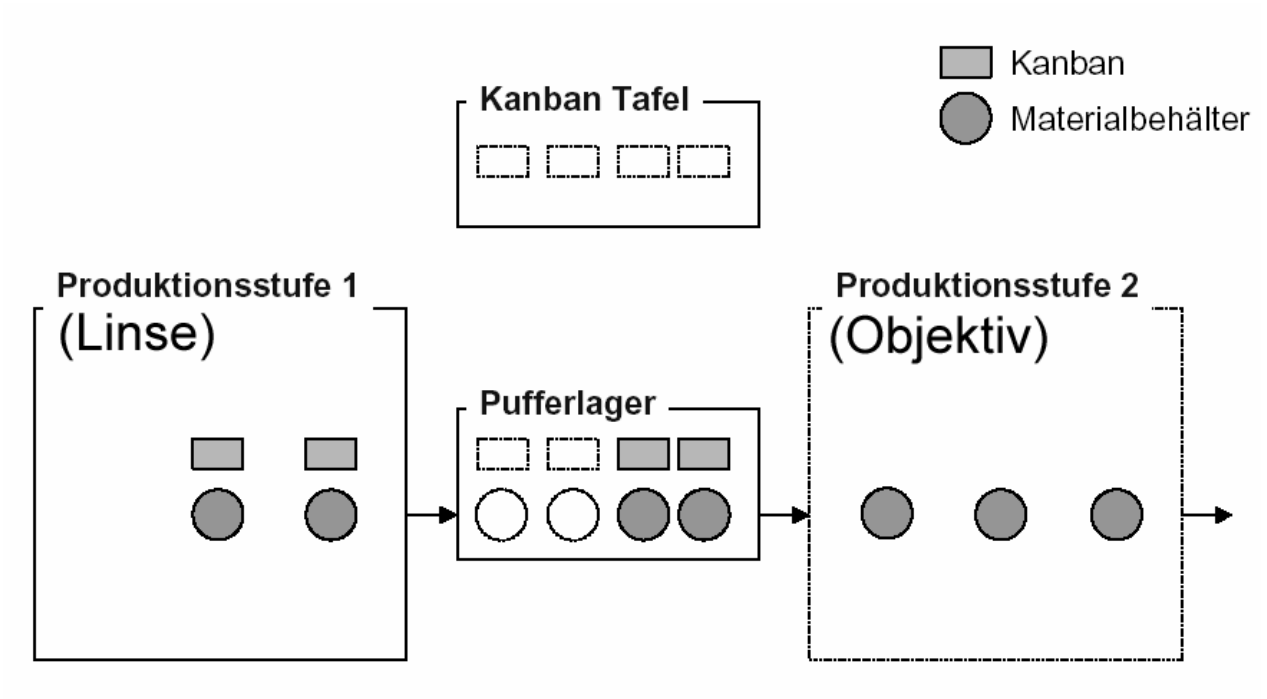
- Einführung einer **Produktionssteuerung** mit Hilfe des Kanban-Prinzips und damit:
  - Minimierung der Lagerbestände
  - Senkung der Durchlaufzeit pro Auftrag
  - Befriedigung aller Nachfragen
- Überprüfung der Formel für die **optimale Anzahl der Kanbans** unabhängig von der anfänglichen Produktionssituation
- **Dynamische Anpassung** der Produktion an die Nachfrage

Einführung in Kanban -allgemein-

- “Kanban” = Karte

- Wurde in den 60er Jahren in Japan entwickelt
- Dezentrales Steuerungskonzept mit selbststeuernden Regelkreisen
- **„Pull“-Konzept** (keine Produktion ohne Nachfrage)

Einführung in Kanban -Prinzip-



Einführung in Kanban -Regeln-

- **Anzahl Kanbans für bestimmtes Material:**

$$\frac{(\text{Ø Bedarf einer Teilperiode} \times \text{Wiederauffüllzeit}) + \text{Sicherheitsfaktor}}{\text{Standardmenge des Behälters}}$$

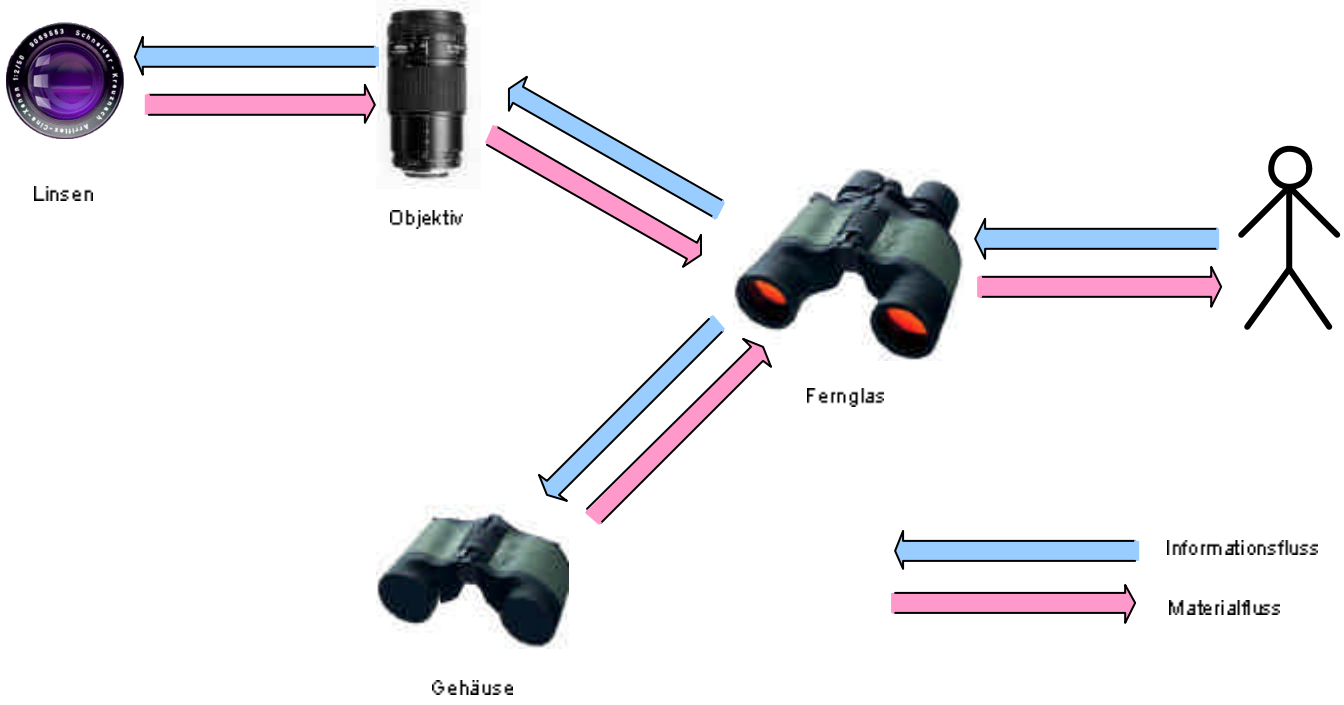
- **Die Materialquelle darf niemals**
  - Mehr Teile als angefordert herstellen
  - Teile vor Eingang der Bestellung herstellen
  - Fehlerhafte Teile abliefern

- **Die Materialsenke darf niemals**
  - Mehr Material anfordern als benötigt
  - Vorzeitig Material anfordern
  - Aus mehreren Behältern mit gleichen Teilen gleichzeitig Material entnehmen

Einführung in Kanban -Voraussetzungen-

- **Verstetigung des Materialflusses**
  - Weitgehend **regelmäßiger** Teilebedarf
  - Keine **Engpässe**
  - Keine **Schwankungen** des Kapazitätsangebotes (ausreichende Kapazitätsreserven, vorsorgliche Wartung und Instandhaltung, flexibler Einsatz von Arbeitskräften)
- **Kurze und sichere Wiederauffüllzeit**
  - Konstantes Kapazitätsangebot
  - Anordnung der Fertigung nach dem Materialfluss

Beschreibung der Produktionssituation 1/2



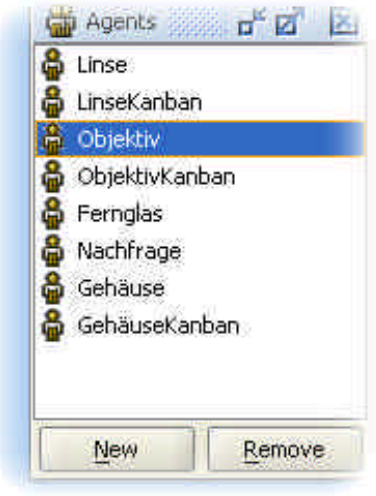
Beschreibung der Produktionssituation 2/2

- Umsetzung des Informationsflusses durch **Warteschlangen:**
  - Bei Teilebedarf trägt sich eine Materialsenke in die Anforderungsliste der Container ein
  - Verfügbare Container nehmen Anforderungen auf und transportieren die Teile zur Produktion
  - Leere Container tragen sich in der Materialbedarfsliste der Materialquelle ein (→Kanban-Tafel)
  - Materialquelle nimmt die Anforderungen der Container als Aufträge auf (→ Entnahme der Kanban von Kanban-Tafel)
  - etc.

→ Nachfrage wird bis zum Produktionsursprung durchgeleitet

Umsetzung in SeSAm

- Welt:
  - Unternehmen
- Agententypen:
  - Produktionsstätten:
    - Linse
    - Objektiv
    - Gehäuse
    - Fernglas
  - Kanbans:
    - LinseKanban
    - ObjektivKanban
    - GehäuseKanban
  - Markt:
    - Nachfrage

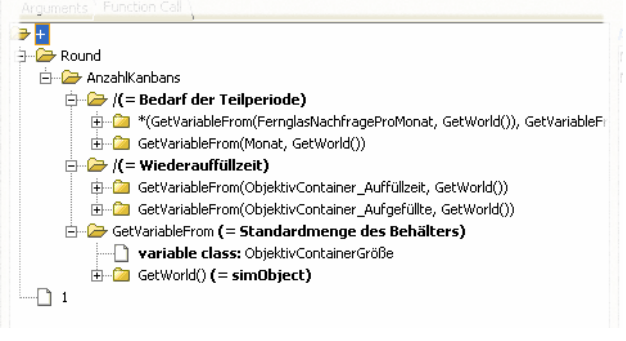


Umsetzung in SeSAm -Unternehmen-

- **Variablen**:
  - alle Warteschlangen (Kanban-Tafeln) für den Produktionsprozess (List <SimObject>)
  - Konstante Variablen für jede Objekt-Klasse (Number<Double>):
    - Produktionskoeffizienten
    - Containergröße
    - Zeitdefinitionen (Tag, Monat, Jahr in Ticks)
    - Produktionsgeschwindigkeiten (Produzierte Teile pro Tick)
  - Maßzahlen
    - zur Bestimmung der Kanban-/ Produktionsstättenanzahl (maximale Produktionsmenge) (Number<Double>)
    - für die grafische Auswertung der Simulationen

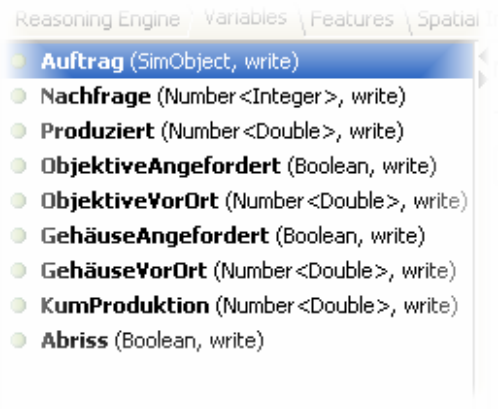
Umsetzung in SeSAm -Unternehmen-

- **Verhalten**:
  - Zufällige Erzeugung von Normalverteilten Nachfragen mit N(15;5)
  - Monatliche Auswertung der Objekt-Anzahl
    - Gespeicherte Kanban-Formeln in den **User Functions** als Soll-Anzahl wird verglichen mit der Ist-Anzahl
    - Fallunterscheidung ob Zerstörung oder Erzeugung von Objekten
  - End-Zustand ist erreicht, wenn keine Veränderung der Kanban-Anzahl zu beobachten ist



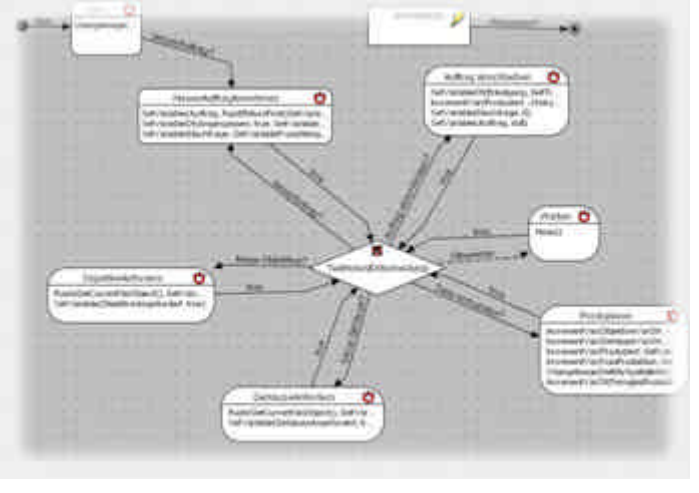
Umsetzung in SeSAm -Produktionsstätten-

- **Variablen**:
  - Auftrag:
    - Verweis auf Nachfrage-Objekte
  - Nachfrage:
    - Nachfragemenge des aktuellen Auftrages
  - Produziert:
    - Produzierte Menge
  - Gehäuse-/ObjektiveAngefordert:
    - Flags für die Anforderung von Containern
  - Gehäuse-/ObjektiveVorOrt:
    - Teile der vorgelagerten Produktionsstätte(n) vor Ort
  - KumProduktion:
    - gesamte Produktion pro Monat
  - Abriss:
    - Signal für Abriss



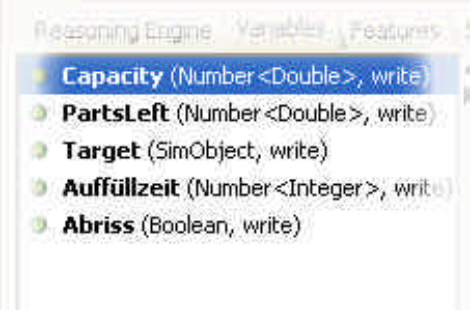
Umsetzung in SeSAm -Produktionsstätten-

- **Verhalten**:
  - Bei Teilemangel kündigt Produktion ihren Bedarf an, indem Sie sich in eine **Anforderungsliste** schreibt
  - Sind Teile vorhanden wird **produziert**, bis der Auftrag erfüllt ist
  - Ist kein Auftrag gesetzt, jedoch befinden sich Aufträge in der Warteschlange, wird ein **neuer Auftrag** angenommen
  - Erfüllte Aufträge werden **abgeschlossen**



Umsetzung in SeSAm -Kanbans-

- **Variablen**:
  - Capacity:
    - Kapazität eines Kanban-Containers
  - PartsLeft:
    - Im Container befindliche Teile
  - Target:
    - Verweis auf Produktionsanlage zur Bestimmung der Position (→MoveTowardsSimObject)
  - Auffüllzeit:
    - Wird in der Zeit inkrementiert, in der ein Container auf seine Auffüllung wartet
  - Abriss:
    - Flag, das signalisiert, dass der Container zerstört werden soll

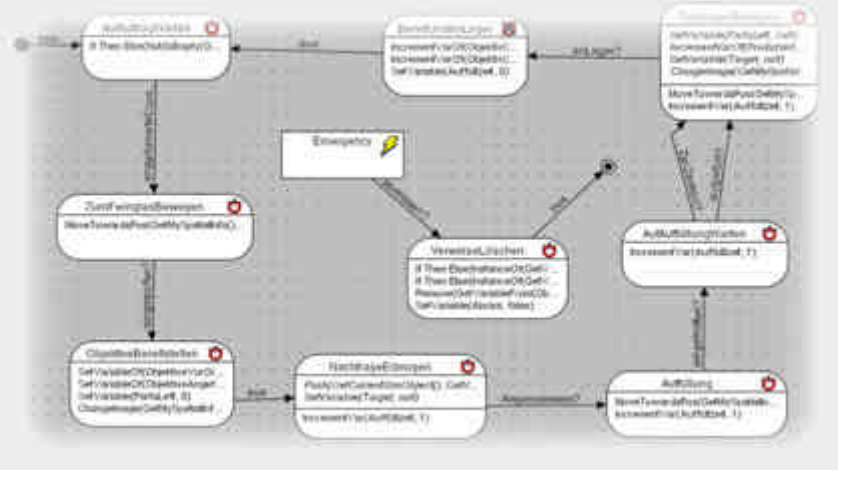


Umsetzung in SeSAm -Kanbans-

- **Verhalten (1/2)**:
  - Kreislauf des Kanban-Containers:
    - Ausgehend vom Startzustand wird eine Materialsenke als Ziel angewählt, sofern diese eine Container-Anforderung in die Warteschlange „gepusht“ hat
    - Der Container bewegt sich auf das Ziel zu und übergibt dem Ziel seine geladenen Teile
    - Anschließend schreibt der Container sich in die Warteschlange der Materialquelle
    - Wenn seine Anforderung von eine Materialquelle angenommen wurde, bewegt sich der Container auf die Produktionsanlage zu (Während der Container auf seine Auffüllung wartet, erhöht sich der Auffüllzeit-Zähler)
    - Wenn genug Teile in der Materialquelle produziert wurden, nimmt der Container diese auf und kehrt zurück ins Lager (Anfangszustand)

Umsetzung in SeSAm -Kanbans-

- **Verhalten (2/2)**:
  - Soll der Container vernichtet werden (Abriss-Flag ist gesetzt), wechselt der Kanban-Agent in den Endzustand
  - Überprüfung erfolgt in jedem Tick durch den **Emergency-Knoten**
  - Zuvor wird das SimObject jedoch aus allen Verweisen entfernt.

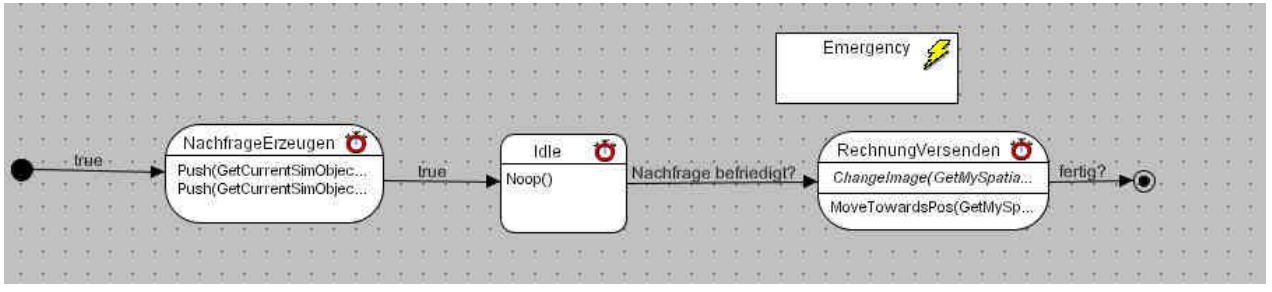


Umsetzung in SeSAm -Nachfrage-

- **Variablen**:
  - Menge:
    - Anzahl der nachgefragten Teile
  - Ankunft
    - Zeitpunkt der Erzeugung
  - Erledigung
    - Zeitpunkt des Abschlusses durch die Fernglas-Produktionsstätte

Umsetzung in SeSAm -Nachfrage-

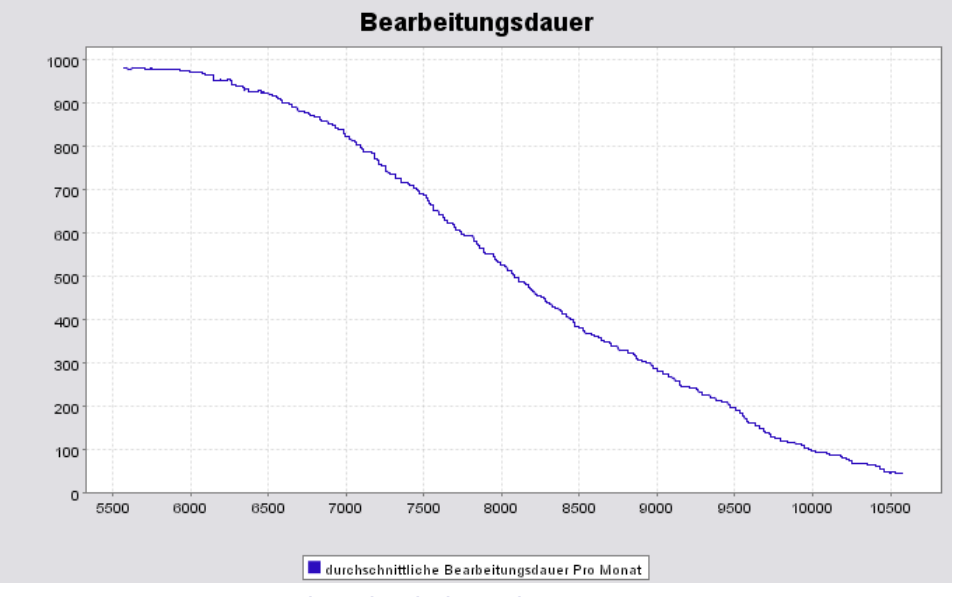
- **Verhalten**:
  - Nach Erzeugung durch die Welt, schreibt sich das Nachfrage-Objekt in die „unbearbeitete Nachfrage“ -Liste
  - Dort verweilt es solange, bis es bearbeitet wurde
  - Danach bewegt es sich aus dem Bildschirmrand (quasi Warenausgang, Rechnungserstellung, …)
  - Zerstörung der Nachfrage



Simulation und Auswertung

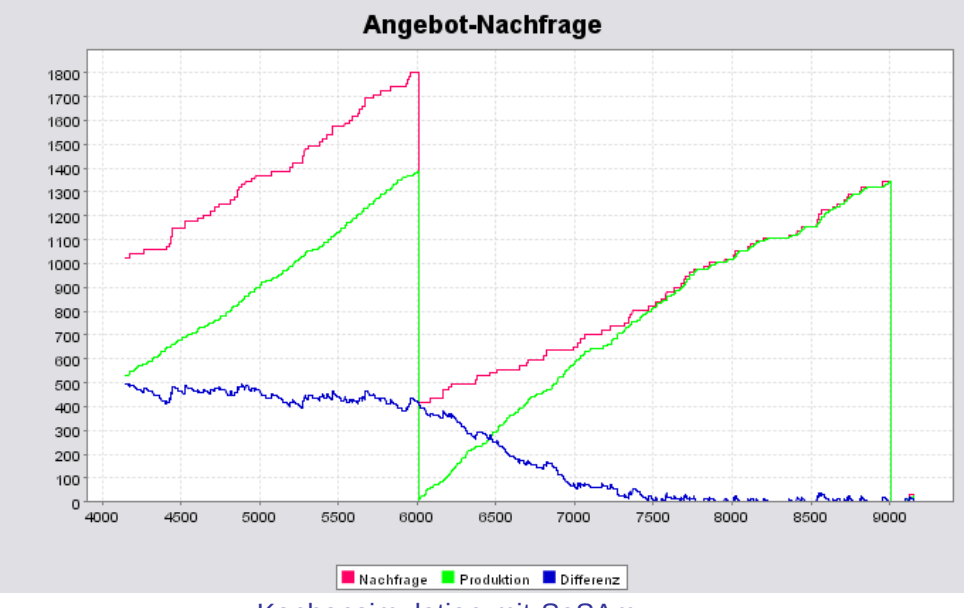
Soll-/Ist-Vergleich auf Modellebene

- Sind die Ziele von Kanban erreicht? →DLZ
  - Bei optimaler Containeranzahl wird die Durchlaufzeit minimal (durchschnittliche Bearbeitungszeit eines Auftrags: ½ **Tag**)



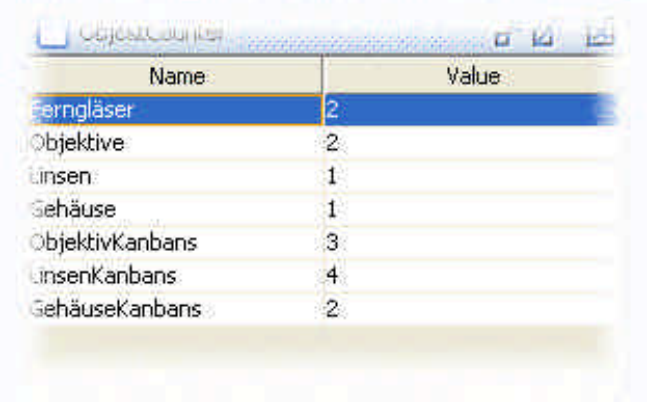
Soll-/Ist-Vergleich auf Modellebene

- Sind die Ziele von Kanban erreicht? →Nachfrage befriedigen
  - Es wird nie mehr als nötig produziert!!!
  - Minimierung der Fehlmenge



Soll-/Ist-Vergleich auf Modellebene

- Sind die Ziele von Kanban erreicht? → Geringe Lagerhaltung
  - Die Anzahl der Kanbans hält sich auf niedrigem Niveau **(→ Steady-State)**
  - Da zwei Objektive benötigt werden, fällt die Anzahl der Container in diesem Produktionszweig etwas höher aus (Linsen analog)



Name	Value
Ferngläser	2
Objektive	2
Linsen	1
Gehäuse	1
ObjektivKanbans	3
LinsenKanbans	4
GehäuseKanbans	2

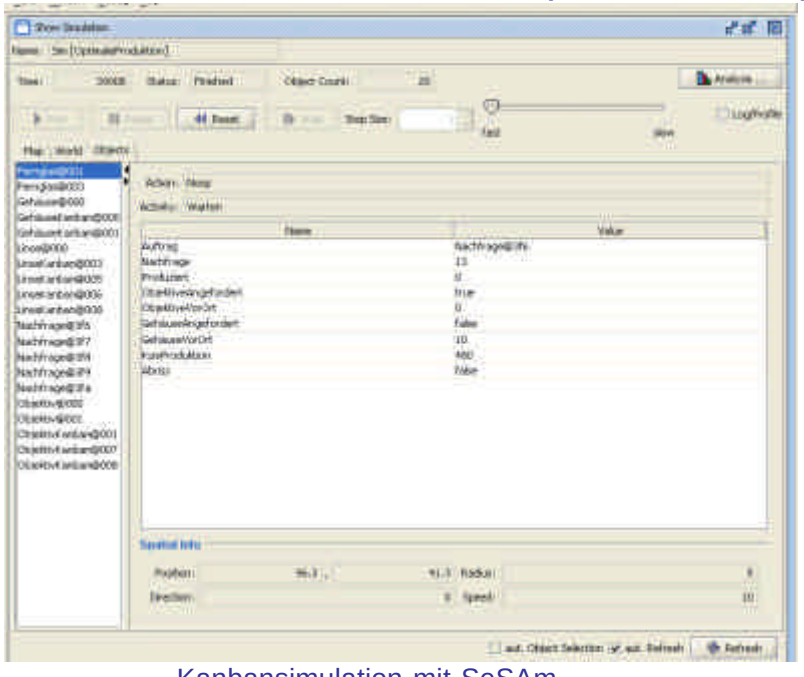
Soll-/Ist-Vergleich auf Modellebene

- Abweichung bei 22 Testläufen ist minimal

	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								

Soll-/Ist-Vergleich auf Modellebene

- **Jetzt:**
  Simulationsdurchlauf mit den ermittelten Optimalwerten (Mittelwerte)



Kritische Auseinandersetzung mit SeSAm

- Einmal Simulation geöffnet, lässt die **Performanz** erheblich nach
- Probleme beim Löschen: Obwohl Referenzen beseitigt wurden
- Probleme beim Kopieren: Trotz *Copy/Paste*-Funktion
- Extrem prozessorlastig, besonders bei **unnötigen grafischen Details** (Grid)
- Analysecharts gehen bei *Experiments* verloren
- Funktionieren der *Hot-Keys* nur teilweise
- Keine gerichteten Nachrichten beim Communication-PlugIn möglich, stattdessen „broadcasten“ vordefinierter Nachrichten (z.B. accept, inform,..)

Kritische Auseinandersetzung mit SeSAm

- **Intuitiver Umgang** mit der Entwicklungsumgebung
- Anschaulichkeit durch **grafische Unterstützung**
- Vielfältige **Analyse**möglichkeiten
- Große Anzahl bereitgestellter **Funktionen** und **Datentypen**
- Gute Idee der Instanziierung einer *Simulation* zu einem *Run,* dadurch mehrere gleiche Simulationsläufe möglich (→ Vergleichbarkeit)
- Einfache Erweiterungsmöglichkeit durch **PlugIns** (Kopieren der *.jar-Datei ins PlugIn Verzeichnis)

Fragen & Diskussion