transentis consulting

Das Beer Game simulieren

Dr. Oliver Grasl
Berlin, 17.11.2021

Inhaltsverzeichnis

1. Einführung in das Beer Distribution Game 4
   1.1. Was macht das Beer Game so interessant? 4
   1.2. Erinnerung: Struktur der Lieferkette des Beer Games 5
   1.3. Entwurf eines Bestands- und Flussmodells 5
   1.4. Die Modul-Struktur 5
   1.5. Die Bestellentscheidung 8
   1.6. Zusammenfassung 9

Das Beer Game simulieren

Ein Bestands- und Flussmodell des Beer Games, das zur Analyse der Spieldynamik verwendet werden kann

Das Beer Game wurde ursprünglich in den späten 1950er-Jahren von Jay Forrester am MIT entwickelt, um die Konzepte dynamischer Systeme vorzustellen. In diesem Beitrag wird ein Bestands- und Flussmodell des Beer Games entwickelt, das für die Analyse der Spieldynamik verwendet werden kann. Das hier entwickelte Modell wird auch als Grundlage für unsere Online-Version des Beer Games verwendet.

#simulation #business transformation #system dynamics #advanced analytics #BPTK #Beer Distribution Game

1. Einführung in das Beer Distribution Game

Das Beer Game wurde ursprünglich in den späten 1950er-Jahren von Jay Forrester am MIT entwickelt, um die Konzepte dynamischer Systeme vorzustellen.

Um die Jahrhundertwende habe ich das erste Mal vom Beer Game gelesen, in Peter Senges wunderbarem Buch *Die fünfte Disziplin.*

Seither habe ich das Spiel in diversen Workshops eingesetzt und es hat für mich nicht an Faszination eingebüßt.

1.1. Was macht das Beer Game so interessant?

Für mich ist der springende Punkt, dass das Spiel recht einfache Regeln hat, aber dennoch viele Aspekte des Managements eines komplexen „Business Ökosystems“ veranschaulicht.

Wenn man das Spiel mit mehreren Spielern spielt, entweder online oder mit einer Brettspielversion, kommen eine ganze Reihe „weicher Faktoren“ hinzu, wie z. B. das Treffen von Entscheidungen, unter (Zeit-)Druck und die (mangelnde) Kommunikation zwischen den Spielern, was eine zusätzliche Dynamik erzeugt.

Hier ist meine Liste der Lieblingsmerkmale des Beer Games:

- Das Beer Game veranschaulicht die Dynamik von Lieferketten, insbesondere den Peitscheneffekt.
- Das Spiel zeigt auf sehr greifbare Weise, wie schwierig es ist, dynamische Systeme zu steuern, selbst wenn man glaubt, sie zu verstehen, und wie frustrierend es sein kann, wenn man nicht die Kontrolle über alle Variablen des Systems hat.
- Es ist eine wunderbare Veranschaulichung des modellbasierten wissenschaftlichen Denkens - die Reduzierung einer komplexen Realität auf ein konzeptionell viel einfacheres Modell und die anschließende Verwendung dieses Modells, um über die Realität nachzudenken.
- Das Beer Game ist auch ein großartiges Studienobjekt, um zu sehen, wie die computergestützte Modellierung helfen kann, komplexe Systeme zu verstehen. Ich war schon immer ein Fan von Raymond Queneaus brillanten Exercises In Style, die dieselbe Geschichte in 99 verschiedenen Stilen erzählen - unser Open Source Github-Repository zeigt nun drei verschiedene Möglichkeiten, ein Simulationsmodell des Beer Games zu erstellen, wobei System Dynamics XMILE, die System Dynamics DSL und die agentenbasierte Modellierung verwendet werden.
- Ich habe auch festgestellt, dass das Spiel ein nützliches Testfeld ist, um zu sehen, wie Techniken des maschinellen Lernens, wie z. B. Reinforcement Learning, eingesetzt werden können, um Agenten zu trainieren, das Beer Game autonom zu spielen, wie im Notebook Training AI to Play the Beer Distribution Gamedokumentiert.

Ich habe bereits ausführlich über das Beer Game geschrieben, insbesondere Understanding The Beer Distribution Game, in dem die Spielstrategien anhand eines Simulationsmodells, das mit System Dynamics erstellt wurde, detailliert analysiert werden.

In diesem Beitrag wird die Dynamik, die dem Beer Game zugrunde liegt, gut erklärt, nicht aber die Struktur des Bestands-und-Flussmodells selbst. Dieser Beitrag füllt diese Lücke.

Hinweis: Unser GitHub-Repository enthält Jupyter-Notebooks und Python-Code, die das hier dargestellte Bestands- und Flussmodell implementieren.

Wenn Sie das Beer Game noch nie gespielt haben, sollten Sie es ausprobieren, bevor Sie diesen Beitrag lesen. Sie können das Spiel unter beergame.transentis.com aufrufen.

1.2. Erinnerung: Struktur der Lieferkette des Beer Games

Die dem Beer Game zugrundeliegende Lieferkette führt von der Brauerei, die das Bier herstellt, über den Verteiler und den Großhändler zum Wiederverkäufer, der das Bier an seine Kunden, die Endverbraucher, verkauft.

Die nachstehende Skizze veranschaulicht die Lieferkette und den Informationsfluss entlang dieser Kette:

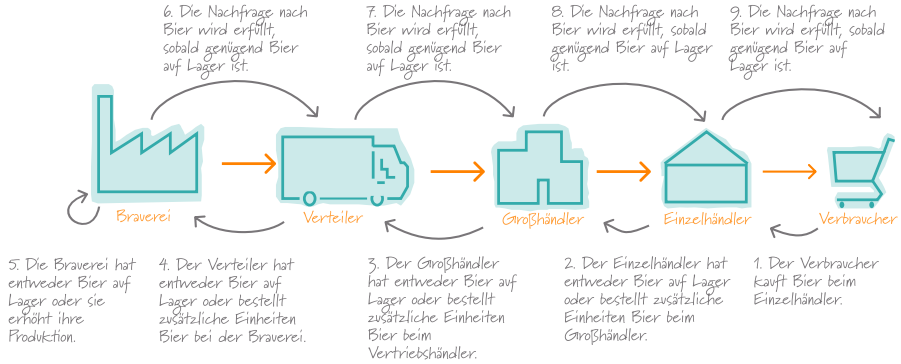


Abbildung 1: Die Bier Supply Chain

1.3. Entwurf eines Bestands- und Flussmodells

Wenn Sie ein Simulationsmodell erstellen, sollten Sie als Erstes versuchen, die Struktur des Modells auf der obersten Ebene zu bestimmen.

1.4. Die Modul-Struktur

In der Regel bezeichnen wir die übergeordneten Bausteine eines Modells als Module, und jedes Modul enthält dann andere Module oder - im Falle von System Dynamics Modellen - die eigentlichen Bestands- und Flussmodelle.

Die wichtigste Erkenntnis beim Aufbau eines Bestands- und Flussmodells für das Beer Game ist, dass das Spiel absichtlich so konzipiert ist, dass die Brauerei, der Verteiler, der Großhändler und der Einzelhändler tatsächlich derselben Dynamik unterliegen - mit einer kleinen Ausnahme: Die Brauerei bestellt kein Bier bei einem Lieferanten, sondern stellt das Bier selbst her. Die Bierherstellung dauert jedoch genauso lange wie die Bestellung und die Erfüllung eines Auftrags in der übrigen Lieferkette, sodass das dynamische Verhalten dasselbe ist.

Dank dieser Erkenntnis können wir dieses gemeinsame Verhalten in eine einzige Struktur abstrahieren, die wir dann beim Aufbau des eigentlichen Simulationsmodells entweder wiederverwenden oder nachbilden können.

Benötigen wir weitere Module? Ja, die brauchen wir:

- Der Sinn eines Simulationsmodells besteht darin, dass wir die Leistung der einzelnen Spieler und der Lieferkette selbst überwachen wollen. Da wir nicht nur die einzelnen Spieler, sondern auch die Gesamtleistung betrachten, ist es sinnvoll, die Überwachung der Lieferkette in einen separaten Teil des Modells auszulagern.
- Um sicherzustellen, dass alle Teile unserer Lieferkette dieselben Entscheidungsgrundsätze und Einstellungen verwenden, z. B. für den Bierpreis und die Zeit, die für die Bestellung oder Herstellung von Bier benötigt wird, extrahieren wir die Einstellungen in ein separates Modul.

Damit ergibt sich die in der nachstehenden Abbildung dargestellte Modellstruktur.

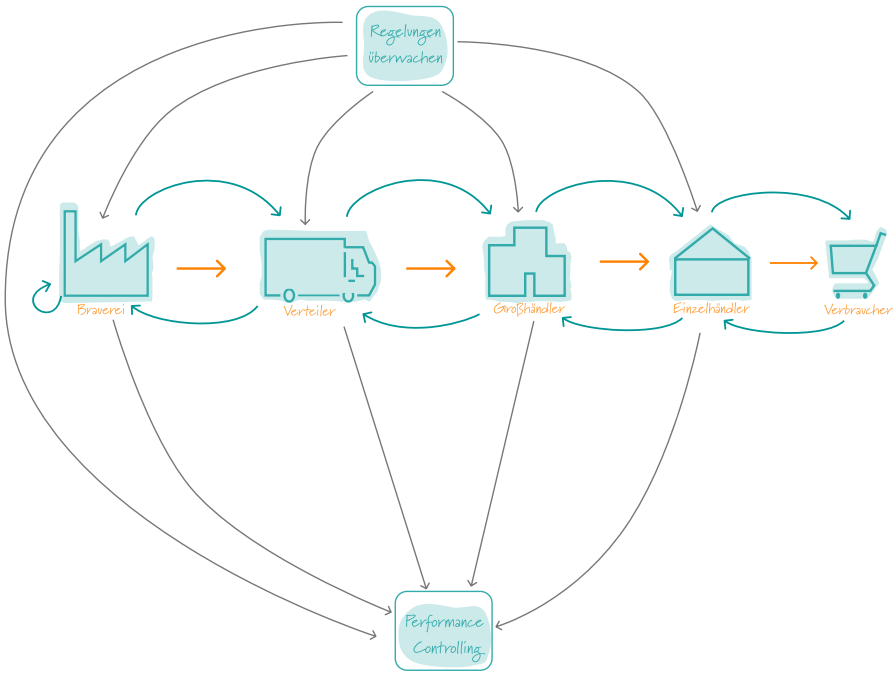


Abbildung 2: Beer Distribution Game Stock And Flow Blog Post Image 1 DE

Nachdem wir nun die Gesamtstruktur des Modells kennen, wollen wir einen Blick auf die Bestände und Flüssen werfen, die wir für die Modellierung der Lieferkettenmodule benötigen.

Jedes Modul in der Lieferkette hat einen Lieferanten und einen Kunden. Tun wir so, als wären wir ein Akteur in der Lieferkette, und gehen wir den Prozess des Lieferkettenmanagements aus dieser Perspektive durch:

1. Zuerst ruft Ihr Kunde Sie an und gibt eine Bestellung auf. Sie vermerken dies in einer Liste names `Orders received`.
2. Sie überprüfen dann Ihren `Inventory`, um zu sehen, ob Sie den Kundenwunsch erfüllen können.
3. Sie geben Ihrem Kunden, was Sie können, und vermerken dies in Ihrer Liste der erfolgten `Deliveries`
4. Sie geben eine Bestellung bei Ihrem Lieferanten auf, um das von Ihrem Kunden bestellte Bier zu beziehen. Um den Überblick über Ihre Bestellungen zu behalten, führen Sie eine Liste mit dem Namen `Open orders`.

Damit sind die Bestände definiert, die Sie benötigen, um zu verfolgen, was in Ihrem Teil der Lieferkette vor sich geht.

Beachten Sie, dass sich das Bier selbst in einem von drei Zuständen befinden kann: Es ist entweder auf Bestellung (d. h. noch bei Ihrem Lieferanten oder auf dem Weg dorthin), es kann sich in Ihrem Bestand befinden, oder Sie haben es geliefert und es ist auf dem Weg zu Ihrem Kunden (oder es wurde bereits vom Kunden entgegengenommen).

Wir können dies als eine Kette von Beständen und Flüssen modellieren, wie im folgenden Diagramm dargestellt.

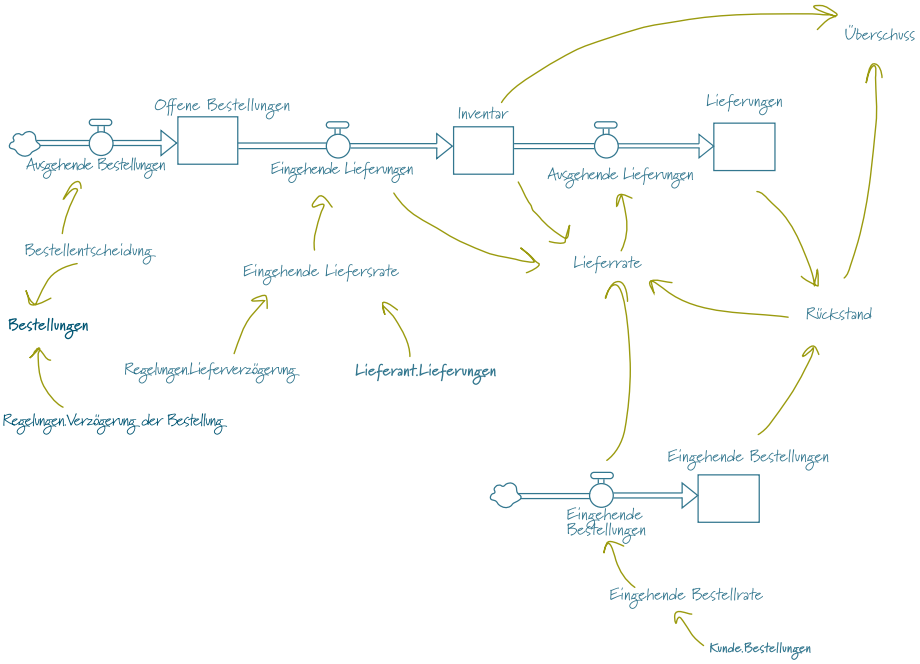


Abbildung 3: Beer Distribution Game Stock And Flow Blog Post Image 2 DE

Das Diagramm zeigt auch zwei Indikatoren, mit denen wir die Leistung unserer Lieferkette messen können: `backorder` und `surplus`.

Der `backorder` ist einfach die Differenz zwischen eingegangenen Bestellungen und erfolgten Lieferungen. Im besten Fall ist der Rückstand gleich 0, d.h. Sie können die Wünsche Ihrer Kunden immer erfüllen:

```
Backorder = Orders_received - Deliveries_made
```

Der `surplus` ist die Differenz zwischen dem Inventar und dem Rückstand. Der Überschuss gibt an, wie viel Bier Sie haben, um die eingehenden Kundenanfragen zu decken. Er sollte immer eine positive Zahl sein, aber nicht zu groß, um die Inventarkosten niedrig zu halten.

```
Surplus = Inventory - Backorder
```

Nachdem wir nun die Bestände und die Indikatoren kennen, können wir einen Blick auf die Stromgrößen und die damit verbundenen Raten werfen.

Fangen wir mit den `incoming orders` an. Dieser ist gleich der `incoming order rate`, die ihrerseits nur gleich dem Auftrag ist, der von Ihrem Kunden gesendet wird:

```
Incoming_Orders = Incoming_Order_Rate = Customer.Sending_Orders
```

Und nun wollen wir die Bierlieferkette durchlaufen:

Die Anzahl der ausgehenden Bestellungen entspricht der Anzahl der von Ihnen getätigten Bestellungen.

```
Outgoing_Orders = Order_Decision
```

Die Schlüsselfrage des gesamten Spiels ist natürlich, wie Sie diese Bestellentscheidung treffen - weil sie so wichtig ist, verschieben wir die detaillierte Diskussion der Bestellentscheidung auf den nächsten Abschnitt.

Die eingehenden Lieferungen hängen von der eingehenden Lieferrate ab, und diese wiederum entspricht den ausgehenden Lieferungen, die von Ihrem Lieferanten verschickt werden, verzögert um die Zeit, die es dauert, bis die Waren bei Ihnen ankommen.

In unserem Spiel ist diese Verzögerung gleich einer Woche, im Modell wird die Verzögerung durch die Lieferverzögerung in den politischen Einstellungen bestimmt. Dies führt zu der folgenden Gleichung:

```
Incoming_Deliveries = Incoming_Delivery_Rate =
Delay(Supplier.outgoing_deliveries,Policy_Settings.Delivery_Delay,100.0)
```

Der einzige verbleibende Fluss sind die *ausgehenden Lieferungen*, die gleich der `outgoing_delivery_rate` sind. Hier ist die Strategie ganz einfach: Wir wollen immer so viel liefern, wie wir können, d.h. die Summe aus dem Auftragsrückstand und dem aktuellen Auftragseingang. Aber wir können nicht mehr liefern, als der aktuelle Bestand und die gerade eintreffenden Bestellungen. Dies führt zu den folgenden Gleichungen:

```
Outgoing_Deliveries= Outgoing_Delivery_Rate =
Min(Backorder+Incoming_orders,Inventory+Incoming_Deliveries)
```

1.5. Die Bestellentscheidung

Kommen wir nun zurück zur Auftragsentscheidung. In unserem Modell haben wir zwei Bestellentscheidungen implementiert, eine „naive“, die Rückstände und die Lieferkette ignoriert, und eine „ausgeklügelte“, die diese berücksichtigt. Die unterschiedlichen Auswirkungen dieser Entscheidung werden in meinem Blogbeitrag ausführlich erörtert - daher werde ich hier nur eine kurze Zusammenfassung geben.

Bei der naiven Bestellentscheidung bestellen Sie im Wesentlichen alles, was der Kunde gerade bestellt, plus den gesamten Rückstand, plus die Differenz zwischen Ihrem aktuellen Bestand und Ihrem Zielbestand:

```
Naive_Order_Decision =  Backorder + Incoming_Order_Rate +
Policy_Settings.Target_Inventory - Inventory
```

Eine ausgefeiltere Auftragsentscheidung ignoriert den Rückstand, bezieht aber die Versorgungslinie (d. h. die offenen Aufträge) mit ein. Außerdem werden Schocks in der Lieferkette vermieden und der Inventar langsam angepasst. Dies führt zu der folgenden Gleichung:

```
Sophisticated_Order_Decision = Incoming_Order_Rate +
(Policy_Settings.Target_Inventory - Inventory + Target_Supply_Line - Open_Orders)/
Policy_Settings.Inventory_Adjustment_Time
```

Die Zielversorgungslinie muss sowohl die erwartete Bestellung als auch die Lieferverzögerungen berücksichtigen:

```
Target_Supply_Line = Incoming_Order_Rate *(Policy_Settings.Delivery_Delay +
Policy_Settings.Order_Delay)
```

Das Diagramm fasst die Faktoren zusammen, die für die Bestellentscheidung relevant sind:

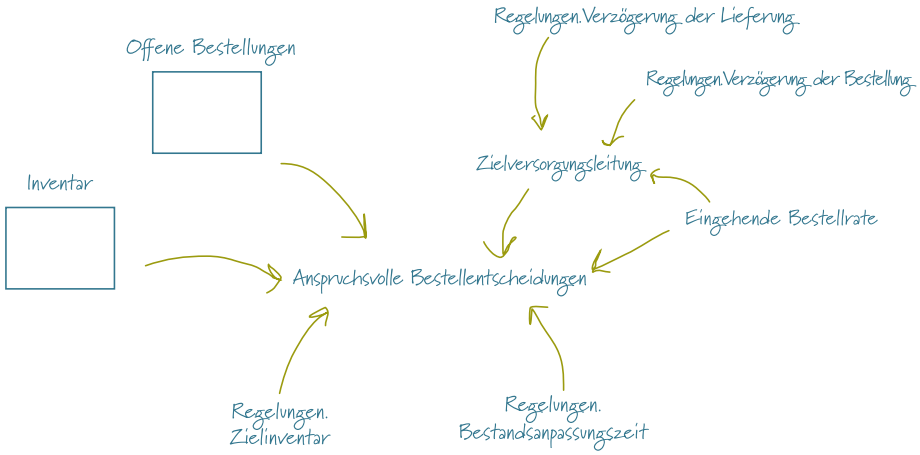


Abbildung 4: Beer Distribution Game Stock And Flow Structure image 3 DE

Hier ist ein kleines Dashboard, mit dem Sie die Auswirkungen der Bestellentscheidung auf die Lieferkette testen können.

Embedded content: https://app.metapad.ai/embed/16cf74a7-e642-4277-b22a-8b5576bca422/eaadcfee-480b-4d66-a961-42965de61205

1.6. Zusammenfassung

In unserem GitHub-Repository finden Sie zwei Implementierungen des Bestands- und Flussmodells, die hier abgebildet sind - eine Implementierung in Python unter Verwendung der BPTK SD DSL und eine Implementierung als Stella-Modell.

Die SD-DSL-Version wird im Begleit-Notebook zu diesem Blogbeitrag ausführlich besprochen.

© www.transentis.com, 1997-2026 transentis consulting GmbH & Co. KG