Universität Zürich[UZH]

IFI INFORMATICS AND SUSTAINABILITY GROUP

Design und Implementierung eines agentenbasierten Modells des Schweizer Energieholzmarktes

Masterarbeit von Stefan Holm

05-709-415

Kilchberg

1. Juli 2011

Betreuung: Bernhard Steubing (EMPA), Fabian Kostadinov (WSL)

Betreuender Professor: Prof. Dr. Lorenz Hilty (EMPA / UZH)

EMPA Materials Science & Technology

WSL

Zusammenfassung

In der vorliegenden Masterarbeit wurde auf Basis eines konzeptuellen Modells des Schweizer Energieholzmarktes ein agentenbasiertes Simulationsprogramm erstellt. Das Simulationsprogramm wurde in Java programmiert. Anstatt die Daten über Agenten sowie die Daten über ihre Interaktionen direkt als Java-Objekte im Arbeitsspeicher zu halten, wurde eine Ontologie-Datenbank für die Speicherung dieser Daten genutzt. Ziel war es, damit bessere Auswertungsmöglichkeiten zu erhalten, indem einerseits durch das Vorhandensein von sämtlichen Interaktionsdaten jederzeit alle Schritte in der Simulation nachvollzogen werden können, andererseits durch die Möglichkeit mittels SPARQL-Queries gezielt einzelne Informationen auszulesen. Zusätzlich sollte mit Hilfe eines Reasoners durch Inferenz-Regeln aus den vorhandenen Daten logische Schlüsse gezogen werden können. Es konnte gezeigt werden, dass durch den Einsatz einer Ontologie-Datenbank tatsächlich Vorteile bezüglich Auswertungsmöglichkeiten entstehen. Auch kann der Java-Code übersichtlicher gehalten werden, da er nur noch Funktionen für die Mutation der Agentendaten enthält, sich jedoch nicht um die Verwaltung der Agentendaten in Java-Klassen kümmern muss. Demgegenüber steht der Nachteil einer schlechteren Performance. Ein Simulationsdurchgang mit 1000 Agenten über einen Simulationszeitraum von 25 Jahren auf einem High-Performance-Cluster dauert zwischen zwei und zehn Stunden, abhängig von den gewählten Einstellungen. Dadurch gestaltete sich die Verifikation und Validierung des Simulationsprogramms schwierig. Konkrete Aussagen über den Schweizer Energieholzmarkt konnten daher mit der aktuellen Version noch keine getroffen werden. Es wurde jedoch anhand eines exemplarischen Beispiels gezeigt, wie mit Hilfe des Simulationsprogramms, ausgehend von einer bestimmten Fragestellung, Aussagen über den Schweizer Energieholzmarkt getroffen werden könnten.

Abstract

In this master thesis, an agent-based simulation software based on a conceptual model of the Swiss woodfuel market has been programmed. The simulation software was implemented in Java. Instead of holding all information about the agents directly as Java objects in memory, an ontology database was used to store the data. The objective of this approach was to get more sophisticated possibilities for the evaluation of the simulations, as on the one hand all the data about the interactions between the agents are available, on the other hand it is possible to read out very specific information with SPARQL-Queries. Additionally, logical consequences should be inferred from the ontology database using a reasoner and specifying inference rules. It could be demonstrated that using an ontology-database in fact improves the possibilities of evaluating the simulation. Moreover, the source code stayed clearer since it only contains methods for the mutation of agent data, but doesn't hold the data itself. However, using the ontology databases decreases the performance of the simulation: simulating 25 years with 1000 agents on a high performance cluster took between two and ten hours, depending on the settings used. This makes the processes of verification and validation even more complex. Hence, with the present version of the simulation program it was not possible to make precise statements about the Swiss woodfuel market. Even though it was demonstrated with the help of an example how the simulation program can be used to make statements about the Swiss woodfuel market, based on a specific question.

Inhaltsverzeichnis

1. Einleitung **1**
- 1.1. Forschungsfragen und Ziele der Arbeit . . . 1
- 1.2. Einordnung der Masterarbeit . . . 2
- 1.3. Gliederung . . . 3

2. Theoretische Grundlagen **4**
- 2.1. Agentenbasierte Modellierung . . . 4
- 2.2. MAIA . . . 5
  - 2.2.1. IAD . . . 6
  - 2.2.2. OperA . . . 6
  - 2.2.3. MAIA . . . 7
- 2.3. Ontologie . . . 7
- 2.4. AHP . . . 8
- 2.5. Der Schweizer Energieholzmarkt . . . 10

3. Konzeptuelles Modell des Schweizer Energieholzmarktes **13**
- 3.1. Action Arena . . . 13
- 3.2. Akteure . . . 13
  - 3.2.1. Entscheidungsverhalten der Akteure . . . 15
- 3.3. Action Situations . . . 16
  - 3.3.1. District Heating Market . . . 16
  - 3.3.2. Market Entry and Exit . . . 18
  - 3.3.3. Industrial Roundwood Market und Woodfuel Market . . . 18
  - 3.3.4. Harvest and Thinning Market . . . 20

4. Architektur-Design und Implementierung **21**
- 4.1. Layer . . . 22
- 4.2. Architektur . . . 22
- 4.3. Implementierung . . . 23
  - 4.3.1. Generelle Funktionsweise . . . 23
  - 4.3.2. Multithreading . . . 25
  - 4.3.3. CSV-Dateien mit externen Preisen . . . 27
  - 4.3.4. Parameter-File . . . 27
  - 4.3.5. Agent-Upscaling . . . 28
  - 4.3.6. Random-Seeds . . . 29
  - 4.3.7. Evaluator . . . 30
- 4.4. GUI . . . 30
  - 4.4.1. Visualisierung . . . 31
  - 4.4.2. Konsolen-Modus . . . 34

4.5. Verwendete Frameworks . . . 34
4.5.1. Jena . . . 35
4.5.2. Prefuse . . . 35
4.5.3. JFreeChart . . . 35
4.5.4. JUnit . . . 35
4.5.5. Javadoc . . . 35

5. Simulation **36**
5.1. Annahmen . . . 36
5.1.1. AHP . . . 36
5.1.2. Weitere Kriterien des Entscheidungsverhaltens . . . 38
5.2. Verifikation und Validierung . . . 38
5.3. Simulation auf dem Cluster . . . 40
5.4. Sensitivitätsanalyse . . . 41
5.5. Szenarien . . . 42

6. Diskussion **44**
6.1. Eignung der Verwendung einer Ontologie-Datenbank . . . 44
6.2. Eignung des generischen Frameworks für zukünftige Arbeiten . . . 45

7. Schlussfolgerungen **47**

8. Ausblick **49**

Literaturverzeichnis **50**

Anhang **53**

A. Annahmen **53**

B. Ontologie **59**

C. Wichtigste Markteigenschaften **60**

D. Programmargumente **61**

1. Einleitung

Als Energieholz wird Holz bezeichnet, welches zur Energiegewinnung genutzt wird, indem es beispielsweise in Form von Holzschnitzeln verbrannt wird, um Wärme zu generieren. Energieholz ist eine erneuerbare Energiequelle und insofern CO_2-neutral, als dass bei der Verbrennung des Holzes nur so viel CO_2 freigesetzt werden kann, wie das Holz während des Wachstums aufgenommen hat. Energieholz hat ein grosses Potential als Alternative zu fossilen Energieträgern. Dieses Potential wird jedoch selbst in Industrieländern erst teilweise genutzt. Studien zufolge wird im Moment in der Schweiz nur ca. die Hälfte bis zwei Drittel des möglichen Potentials an Energieholz genutzt [5],[11],[17],[18]. Die vorliegende Masterarbeit soll die Auswirkungen des Verhaltens der Akteure auf die Potentiale des Energieholzes in der Schweiz genauer analysieren, indem der Schweizer Energieholzmarkt mittels eines Simulationsprogramms simuliert wird. Der Fokus der Arbeit liegt dabei auf dem Zusammenhang zwischen dem Entscheidungsverhalten der in den Energieholzmarkt involvierten Akteure und der Verfügbarkeit von Energieholz, d.h. welche Menge an Energieholz jährlich genutzt werden könnte. Die Analyse erfolgt aufgrund verschiedener Szenarien, in denen die einzelnen Akteure jeweils unterschiedliche Entscheidungen treffen, indem sie einzelne Entscheidungskriterien unterschiedlich gewichten.

Die Masterarbeit baut auf einem konzeptuellen Modell des Schweizer Energieholzmarktes auf, welches an der EMPA und der WSL im Rahmen der Doktorarbeit von Steubing et al.[16] entwickelt wurde. Im Zuge dieser Masterarbeit soll dieses konzeptuelle Modell in ein geeignetes Simulationsprogramm umgesetzt werden, damit mittels Simulationen die Einflüsse des Akteurverhaltens auf den Energieholzmarkt bemessen werden können. Die Simulation wird anhand von Daten des Kantons Aargau durchgeführt, soweit diese verfügbar sind.

1.1. Forschungsfragen und Ziele der Arbeit

Primäres Ziel der vorliegenden Arbeit ist das Design und die Implementierung eines Simulationsprogramms, welches das entwickelte konzeptuelle Modell des Schweizer Energieholzmarktes simulieren kann. Konkret soll das implementierte Simulationsprogramm folgende Eigenschaften aufweisen:

- An das konzeptuelle Modell angelehnte Architektur
- Eingabedaten:
  - Rollen
  - Akteure
  - Szenarien
- Ausgabedaten:
  - Verfügbare Menge Energieholz in Abhängigkeit der Zeit (als Diagramm)

  - Interaktionsdaten (in einer Ontologie-Datenbank)

- Flexibel erweiterbare Architektur, beispielsweise durch zusätzliche Akteure und Rollen
- Effiziente Algorithmen, welche auch für eine grosse Anzahl an interagierenden Akteuren skalieren

Diese Ziele werden anschliessend durch Verifikation und Validierung des Simulationsprogramms überprüft. Die Überprüfung soll dabei durch Vergleich von Simulationsergebnissen mit historischen Daten und durch Experteninterviews erfolgen. Die Entwicklung des Simulationsprogramms und die Validierung erfolgen in einem iterativen Prozess. Das Simulationsprogramm wird in der Programmiersprache Java implementiert.

Sobald Verifikation und Validierung zeigen, dass das Simulationsprogramm die Anforderungen erfüllt, sollen in einem nächsten Schritt verschiedene Szenarien gebildet, diese simuliert und anschliessend die Ergebnisse der verschiedenen Simulationen analysiert werden. Durch diese Analysen sollen anschliessend Aussagen getroffen werden über:

- Die Energieholzverfügbarkeit in Abhängigkeit verschiedener Szenarien
- Die wichtigsten Einflussfaktoren auf die Energieholzverfügbarkeit (z.B. Ölpreis, Grad der Profitorientierung der Förster / Konsumenten etc.)

Ziel ist es, die gesamte Architektur in zwei klar getrennte Schichten zu unterteilen. Die untere Schicht soll ein Framework sein, welche für die Simulation von Märkten mit ähnlichen Eigenschaften wie der Schweizer Energieholzmarkt ebenfalls verwendet werden kann. Die obere Schicht beinhaltet alles, was spezifisch zum Schweizerischen Energieholzmarkt gehört. Die Trennung der Architektur in zwei Schichten soll die Wiederverwendbarkeit der Software für ähnliche Zwecke ermöglichen.

1.2. Einordnung der Masterarbeit

In diesem Abschnitt wird die vorliegende Masterarbeit in die bisherigen wissenschaftlichen Arbeiten eingeordnet, und insbesondere auch den Bezug zum Modell des Schweizer Energieholzmarktes von Steubing et al.[16] und des MAIA-Modells[1] gezeigt.

Im Modell von Steubing et al. wird der Schweizer Energieholzmarkt auf ein konzeptuelles Modell abgebildet. Die Abbildung auf das konzeptuelle Modell erfolgt mit Hilfe des MAIA-Frameworks, welches in Kapitel 2.2 erklärt wird. In der vorliegenden Masterarbeit wurde nun versucht, dieses konzeptuelle Modell des Schweizer Energieholzmarktes, welches auf dem MAIA-Framework basiert, in ein implementiertes Modell umzusetzen. Zielsetzung war jedoch nicht nur eine eins-zu-eins Implementierung des konzeptuellen Modells; es sollte vielmehr versucht werden, bei der Implementierung des spezifischen Modells des Schweizer Energieholzmarktes wiederum generische Komponenten zu finden, und somit eine Vorlage für zukünftige Arbeiten zu kreieren, welche agentenbasierte Modelle von Märkten mit ähnlichen Eigenschaften abbilden können. Die Einordnung der vorliegenden Arbeit in die bisherigen wissenschaftlichen Arbeiten ist in Abbildung 1.1 graphisch dargestellt.

[1] MAIA: A methodology for agent-based modelling using institutional analysis[3]), siehe Kapitel 2.2

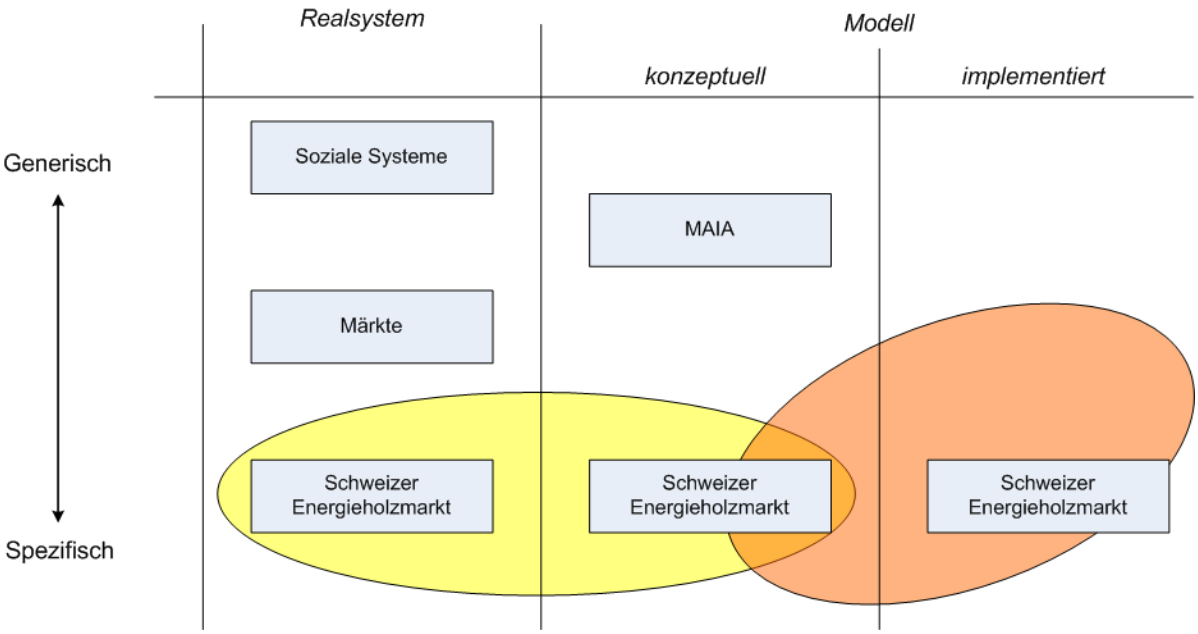


Abbildung 1.1.: Einordnung der vorliegenden Masterarbeit in bisherige wissenschaftliche Arbeiten. Die gelb markierte Fläche zeigt die Arbeit von Steubing et al., in welcher der Schweizer Energieholzmarkt in ein konzeptuelles Modell abgebildet wurde. In dieser Masterarbeit wurde versucht, dieses konzeptuelle Modell des Schweizer Energieholzmarktes in ein implementiertes, simulationsfähiges Modell, d.h. in ein Simulationsprogramm, umzusetzen. Während des Prozesses der Umsetzung des konzeptuellen Modells in das implementierte Modell ergaben sich Erkenntnisse, die sich wiederum auf das konzeptuelle Modell auswirkten. Dies ist in der Abbildung durch die Überschneidung der orangen und der gelben Fläche ersichtlich. Die orange Fläche geht mehr in Richtung Generezität als die gelbe, da in der vorliegenden Arbeit auch generische Konzepte gefunden werden sollten, welche sich für künftige Arbeiten eignen (z.B. Abbildung des französischen Papierholzmarktes).

1.3. Gliederung

In **Kapitel 2** folgt ein Überblick über die der Masterarbeit zu Grunde liegenden theoretischen Grundlagen. Anschliessend wird in **Kapitel 3** das konzeptuelle Modell des Schweizer Energieholzmarktes vorgestellt, welches von Steubing et al. im Rahmen einer Dissertationsarbeit entwickelt wurde. In **Kapitel 4** wird auf die Details der Softwarearchitektur des implementierten Simulationsprogramms eingegangen, während in **Kapitel 5** auf die mit Hilfe des Simulationsprogramms durchgeführten Simulationen eingegangen wird. In **Kapitel 6** folgt schliesslich eine Diskussion der Ergebnisse. Nach den Schlussfolgerungen in **Kapitel 7** erfolgt ein Ausblick in **Kapitel 8**.

2. Theoretische Grundlagen

Das bereits entwickelte konzeptuelle Modell des Schweizer Energieholzmarktes basiert auf der Methode der agentenbasierten Modellierung (ABM). Eine kurze Einführung ins Thema ABM wird im nachfolgenden Unterkapitel gegeben. Anschliessend folgt eine Erläuterung des MAIA-Frameworks, welches als Hilfsmittel für die Erstellung des konzeptuellen Modell des Schweizer Energieholzmarktes diente. Im den nächsten beiden Unterkapiteln dieses Kapitels wird zuerst ein kurzer Überblick über das Thema Ontologie gegeben, danach wird AHP, ein Verfahren zur Entscheidungsfindung, erklärt. Im letzten Unterkapitel folgt schliesslich eine kurze Übersicht über den Schweizer Energieholzmarkt.

2.1. Agentenbasierte Modellierung

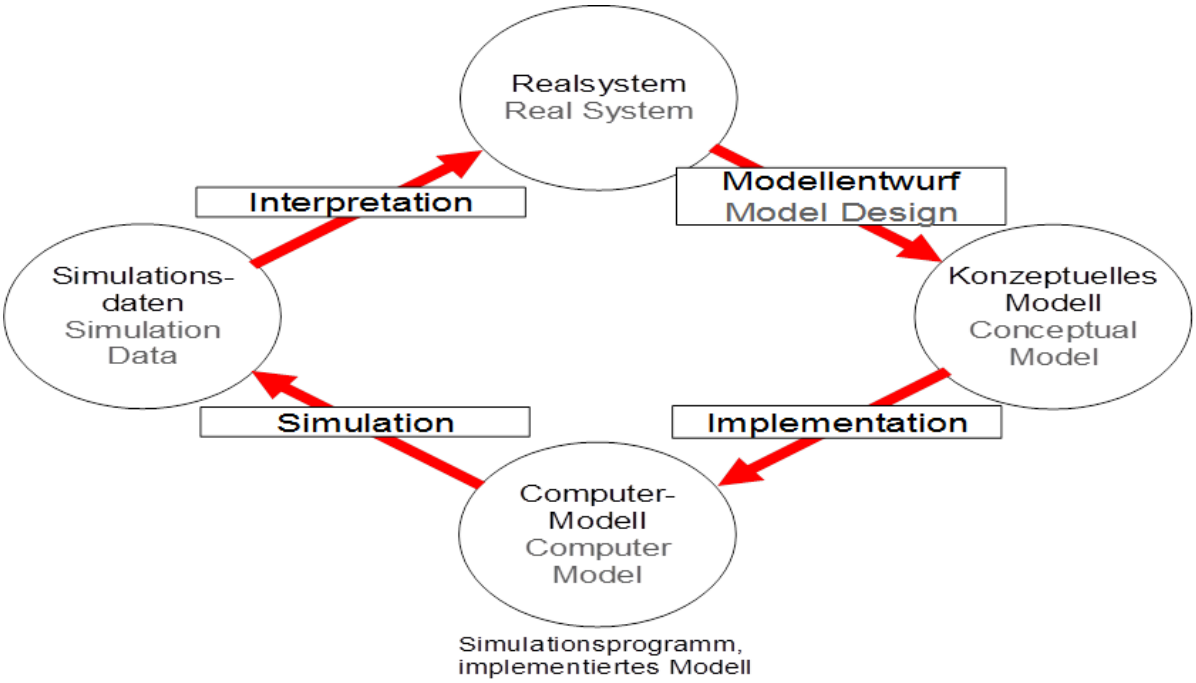


Abbildung 2.1.: Modellierungszyklus in der agentenbasierten Modellierung (nach [10]).

Bei der agentenbasierten Modellierung (ABM) wird das Verhalten einer Vielzahl von einzelnen Akteuren simuliert, um das daraus resultierende Makroverhalten analysieren zu können. Dies setzt voraus, dass beispielsweise Wissen darüber vorhanden ist, wie ein Akteur Entscheidungen trifft und welche Informationen seine Entscheidungen beeinflussen [6]. Abbildung 2.1 zeigt den typischen Modellierungszyklus in der ABM: Ausgangspunkt ist ein *Realsystem*, welches modelliert werden soll, damit im Anschluss Aussagen über dieses Realsystem getroffen

werden können. In der Phase des *Modellentwurfs* entsteht aus dem Realsystem ein *konzeptuelles Modell*. Das konzeptuelle Modell bezeichnet hierbei ein Abbild des Realsystems, das gewissen Einschränkungen unterliegt. Die Art und der Grad der Einschränkungen ist durch den späteren Verwendungszweck des Systems bestimmt. Das entstandene konzeptuelle Modell muss im nächsten Schritt, der *Implementation*, in ein *Computermodell* umgesetzt werden. Oft gibt es - dies wurde auch in der vorliegenden Arbeit festgestellt - in dem Schritt der Implementierung bereits Rückkopplungen auf das konzeptuelle Modell. Insbesondere Lücken und Ungenauigkeiten im konzeptuellen Modell fallen während der Implementierung auf, so dass auch die Implementierung bereits zu einer Verfeinerung des konzeptuellen Modells beitragen kann. Sobald das Computermodell[1] fertig implementiert wurde, werden *Simulationen* durchgeführt. Als Ergebnis dieser Simulation entstehen *Simulationsdaten*. Werden diese schliesslich *interpretiert*, können Rückschlüsse auf das Realsystem gezogen werden.

Es bleibt zu erwähnen, dass das beschriebene Vorgehen zur agentenbasierten Modellierung ein Modellierungs*zyklus* ist, in welchem gewonnene Erkenntnisse beispielsweise aus der Interpretation der Simulationsdaten wieder in den Modellentwurf einfliessen können, damit dieses verfeinert werden kann. Wichtig in diesem Zusammenhang ist jedoch die Validierung und die Verifikation, sowohl des konzeptuellen Modells wie auch des Computermodells. Diese beiden Begriffe stehen nahe beieinander, werden jedoch unterschiedlich definiert:

- **Validierung**. Die Validierung bezeichnet den Vorgang, bei welchem geprüft wird, ob ein bestimmtes System die spezifizierten Anforderungen erfüllt. In Bezug auf die vorliegende Arbeit ist das System valid, wenn es den Schweizer Energieholzmarkt für den vorgegebenen Verwendungszweck korrekt simuliert. Wenn es jedoch den französischen Weinmarkt simuliert, ist es - obwohl es möglicherweise verifiziert und somit formal korrekt ist - für den hier spezifizierten Zweck nicht valid. Bei der Validierung wird geprüft, ob man *das Richtige* richtig tut.
- **Verifikation**. Bei der Verifikation wird die formale Korrektheit eines Systems geprüft. Es wird geprüft, ob man das Richtige *richtig* tut.

Sowohl Validierung wie auch Verifikation sind nötig, um die Konfidenz des Modells zu bekräftigen, bzw. zu erhöhen (siehe Abbildung 2.2)

2.2. MAIA

MAIA (A Methodology for Agent-based Modelling using Institutional Analysis[3])) ist ein Framework, welches die Analyse von sozialen Strukturen ermöglichen soll, um damit die Grundlagen für das Design und die Entwicklung eines agentenbasierten Modells zu schaffen. MAIA basiert auf zwei Frameworks: dem *Institutional Analysis and Development Framework* (IAD) und dem *OperA Framework*. Diese beiden Frameworks werden in den nachfolgenden beiden Unterkapiteln kurz erklärt. Anschliessend wird in Kapitel 2.2.3 der Zusammenschluss dieser beiden Konzepte erklärt, und somit der Kreis zum MAIA-Framework geschlossen.

[1] Das Computermodell wird im Zusammenhang mit ABM auch *Simulationsprogramm* oder *implementiertes Programm* genannt. Im den nachfolgenden Kapiteln der Arbeit wird der Begriff Simulationsprogramm verwendet.

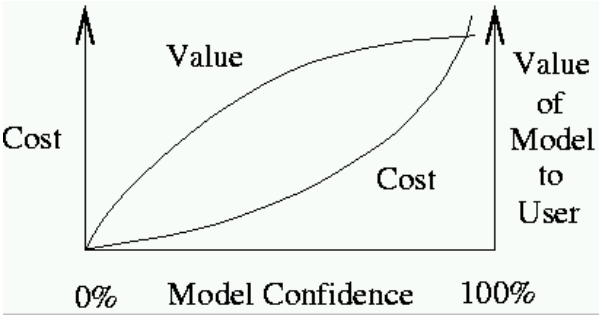


Abbildung 2.2.: Konfidenz eines Modells (nach [15]).

2.2.1. IAD

Das IAD-Framework stammt aus den Sozialwissenschaften und beschreibt ein soziales System, wie beispielsweise einen Markt, auf der Basis von Institutionen. Eine Institution beschreibt ein System aus Regeln (z.B. Verträge, Normen, Sitten), welches das Handeln von Individuen beschreibt. Das IAD-Modell von Ostrom[9] definiert verschiedene Elemente; das zentrale dabei ist die Action Arena, in welcher Individuen miteinander interagieren, um beispielsweise Güter auszutauschen. Diese Action Arena besteht aus Teilnehmern mit individuellen Eigenschaften, sowie der Action Situation, dem Ort an welchem die Teilnehmer interagieren. Die Interaktionen der Teilnehmer in der Action Arena führen zu Interaktionsmustern und Ergebnissen, welche aufgrund von bestimmten Evaluationskriterien analysiert werden können. Die Ergebnisse beeinflussen schliesslich die physische Welt, die Gesellschaft und ihre Prinzipien, welche zusammen wiederum die Action Arena beeinflussen (Rückkopplung). Diese Struktur ist in Abbildung 2.3 dargestellt.

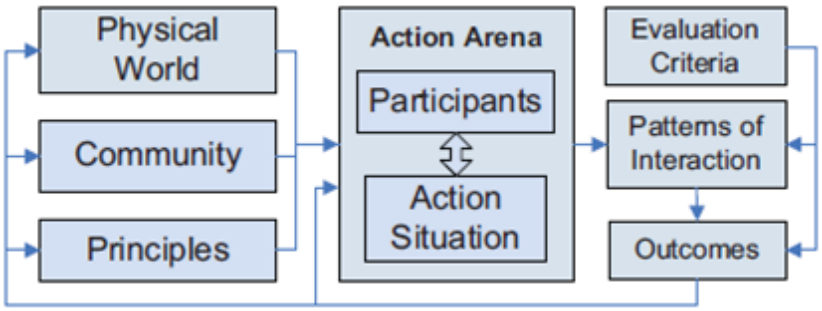


Abbildung 2.3.: IAD-Framework [3]

2.2.2. OperA

Das OperA Framework ermöglicht die Definition von offenen Organisationen und unterscheidet dabei explizit zwischen den Zielen der Organisation und den darin agierenden Akteuren. Dadurch wird es möglich, dass die Akteure aufgrund ihrer eigenen Möglichkeiten und Bedürfnissen frei bestimmen, wie sie innerhalb der Organisation handeln. Das OperA-Framework besteht

aus drei miteinander verknüpften Modellen: Das Organisationsmodell, das Interaktionsmodell und das Sozialmodell. Das Organisationsmodell beschreibt das gewünschte Verhalten der Organisation. Dieses wird von Stakeholdern der Organisation bestimmt. Das Sozialmodell definiert Rollen in der Organisation, welche von Agenten angenommen werden können. Bei einem Agent, welcher eine Rolle annimmt, spricht man von einem role-enacting agent. Das Interaktionsmodell definiert, wie die die verschiedenen role-enacting agents miteinander kommunizieren.

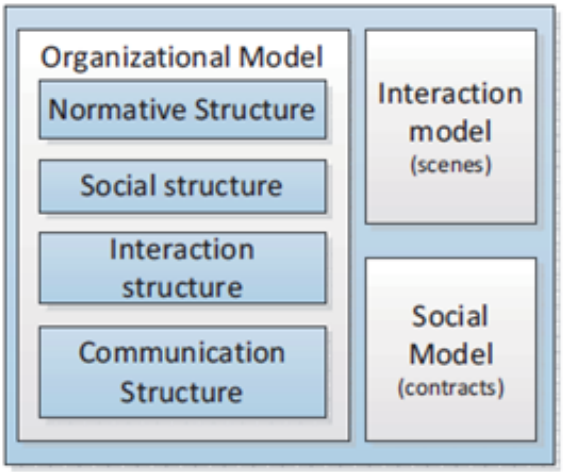


Abbildung 2.4.: OperA-Framework [3]

2.2.3. MAIA

In den beiden vorhergehenden Unterkapiteln wurden die beiden Frameworks IAD und OperA erläutert. Abbildung 2.5 zeigt nun den gesamten MAIA-Modellierungszyklus bzw. den Zusammenhang zwischen IAD, OperA und MAIA: Das MAIA Meta-Modell basiert auf IAD und OperA. Mithilfe dieses Meta-Modells wird das gewünschte zu modellierende Realsystem analysiert, und zu einem konzeptuellen Modell verarbeitet. Dieses konzeptuelle Modell liefert die Grundlage für eine Computer-Implementierung des Realsystems, wobei jedoch die Implementierung selbst schliesslich zum agentenbasierten Modellierungszyklus (siehe Kapitel 2.1) gehört, und nicht mehr Bestandteil des MAIA-Frameworks ist.

2.3. Ontologie

Der Begriff *Ontologie* stammt ursprünglich aus der Philosophie und bezeichnet dort *die Grunddisziplin der Seinswissenschaft oder Lehre vom Seienden, die die formalen (oberste Strukturen und Gesetzlichkeiten) und Materialien (inhaltl. Gliederung des Seienden) Prinzipien des Gegebenen untersucht*[8].

In der Informatik dient eine Ontologie dazu, Wissen in einer für Computer verständlichen Form zu speichern. Dazu enthalten Ontologien einerseits eine formale Beschreibung der Daten, die sie beinhalten, andererseits eine Beschreibung der Relationen zwischen diesen Daten. Mit

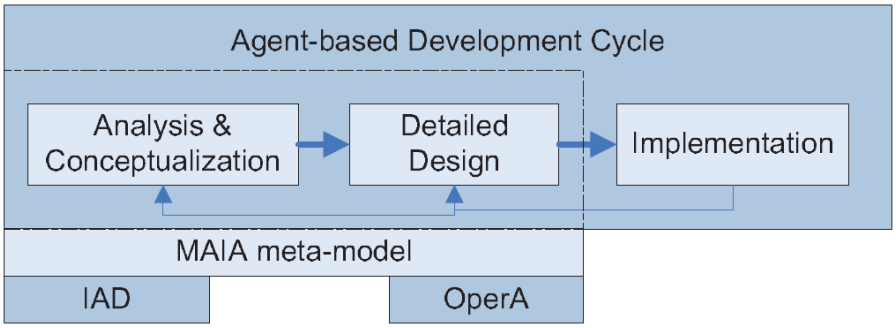


Abbildung 2.5.: Der MAIA-Modellierungszyklus [4]

Hilfe von Inferenzregeln kann damit aus vorhandenem Wissen neues Wissen abgeleitet werden. Ein einfaches Beispiel hierfür ist eine gespeicherte Mutter-Tochter-Beziehung: In der Ontologie wird festgehalten, dass die Beziehung *A ist Mutter von B* die Beziehung *B ist Kind von A* impliziert. Werden nun in der Ontologie die Subjekte Anna und Berta miteinander in Beziehung gesetzt, indem festgehalten wird, dass Anna die Mutter von Berta ist, kann nun mittels Inferenzregeln abgeleitet werden, dass Berta das Kind von Anna ist, ohne dass diese Beziehung explizit in der Ontologie festgehalten wird.

Es existieren verschiedene Sprachen für die Beschreibung von Ontologien. In der vorliegenden Arbeit wurde die Web Ontology Language (OWL) verwendet. OWL basiert auf der Syntax von RDF, dem *Resource Description Framework*[1]. OWL hat jedoch eine höhere Ausdrucksmächtigkeit als RDF. Zur Speicherung von einfachen OWL-Ontologie-Datenbanken werden in der Regel XML-Dateien verwendet. Da diese für grössere Datenbestände jedoch weniger geeignet sind, gibt es auch die Möglichkeit, Ontologien in relationalen Datenbanken zu speichern[2], oder sog. Triplestores zu verwenden. In einem Triplestore werden alle Daten der Ontologie als Triples abgelegt, d.h. Subjekt-Prädikat-Objekt (Beispiel: Anna - ist Mutter von - Berta). Wie bei relationalen Datenbanken kann hier die Performance durch Indizierung stark verbessert werden.

Im Simulationsprogramm wurde zur Speicherung der Eigenschaften der Agenten und deren Beziehungen eine auf einer Ontologie basierende Datenbank gewählt. Ziel war es, durch diesen Ansatz beispielsweise durch den Einbezug von Inferenzregeln bessere Auswertungsmöglichkeiten zu erhalten.

2.4. AHP

AHP (Analytic Hierarchy Process [12]) ist ein Verfahren zur Rationalisierung von Entscheidungsprozessen. Es dient der Entscheidungsfindung bei Vorhandensein von mehreren Alternativen und mehreren zu beachtenden Kriterien. Dazu wird in folgenden Schritten vorgegangen:

[1] Das Resource Description Framework (RDF) ist ein Standard, mit welchem Metadaten im World Wide Web beschrieben werden können. Es ist eine wichtige Komponente im Semantic Web.

[2] Dazu ist ein entsprechendes Framework nötig, welches die Ontologie-Datenbank auf eine relationale Datenbank abbildet. Ein Beispiel für ein solches Framework ist SDB, welches zusammen mit dem Jena-Framework verwendet werden kann.

- Auflistung aller vorhandenen Alternativen
- Festlegen der verschiedenen zu berücksichtigenden Kriterien
- Gewichtung der Kriterien durch paarweises Vergleichen der einzelnen Kriterien
- Abschätzen des Grades der Erfüllung für die verschiedenen Kriterien für jede einzelne Alternative, ebenfalls durch paarweise Vergleiche
- Berechnung der Gewichtungsvektoren, sowohl für die Kriterien, als auch für die Alternativen
- Mathematische Bestimmung der besten Alternative bzw. einer Rangfolge der n-besten Alternativen

Die Kriterien können in AHP hierarchisch geordnet werden. Für die vorliegende Arbeit wurden jedoch alle Kriterien in die gleiche Hierarchiestufe gesetzt, weshalb hier nicht auf das Vorgehen der Berechnung von hierarchischen geordneten Kriterien eingegangen wird. Tabelle 2.1 zeigt anhand eines Beispiels die Berechnung des Gewichtungsvektors für unterschiedliche Kriterien.

	Profit	Umwelt	Freundschaft	*Gewichtungsvektor*
Profit	1	5	7	0.73
Umwelt	1/5	1	3	0.19
Freundschaft	1/7	1/3	1	0.08

Tabelle 2.1.: Gewichtung von unterschiedlichen Kriterien in AHP und resultierender Gewichtungsvektor. In diesem Beispiel spielen für die Entscheidungsfindung die Kriterien Profit, Umwelt und Freundschaft eine Rolle. Mit paarweisen Vergleichen wird für jedes Kriterium die Wichtigkeit gegenüber den anderen Kriterien bestimmt. Dazu werden mit Hilfe von Tabelle 2.3 die Werte in die Matrix eingetragen. Der Wert 1 bedeutet, dass die beiden Kriterien die gleiche Wichtigkeit haben (deshalb sind die Werte auf der Hauptdiagonale immer 1). Sich in Bezug auf die Hauptachse gegenüberliegende Felder enthalten jeweils den reziproken Wert voneinander. Nach der Bestimmung aller Werte wird der Gewichtungsvektor berechnet. Der Gewichtungsvektor entspricht dem Eigenvektor der Matrix; jedoch schlägt Saaty in [12] als Alternativen auch vereinfachte Verfahren für die Berechnung des Gewichtungsvektors vor, da der Eigenvektor z.B. bei Matrizen höherer Dimension nicht mehr exakt berechnet werden kann.

Im Anschluss an die Berechnung des Gewichtungsvektors für die Kriterien muss derselbe Vorgang für alle Alternativen wiederholt werden. Anstatt der unterschiedlichen Kriterien stehen nun in den Spalten und Zeilen die verschiedenen Alternativen. Stehen drei Kriterien und vier Alternativen zur Auswahl, müssen drei 4x4 Matrizen mit Werten gefüllt werden. Pro Matrix müssen für jedes einzelne Kriterium die verschiedenen Alternativen gegeneinander abgewogen werden. Tabelle 2.2 zeigt ein Beispiel einer solcher Matrix mit dem zugehörigen Gewichtungsvektor.
In den bisherigen Schritten wurden somit nun vier Gewichtungsvektoren berechnet: einer für die Kriterien und drei, welche für jedes einzelne Kriterium die Werte der vier Alternativen

Profit	Offerte 1	Offerte 2	Offerte 3	Offerte 4	*Gewichtungsvektor*
Offerte 1	1	3	6	9	0.60
Offerte 2	1/3	1	3	4	0.24
Offerte 3	1/6	1/3	1	2	0.10
Offerte 4	1/9	1/4	1/2	1	0.06

Tabelle 2.2.: Gewichtung von unterschiedlichen Alternativen (vier unterschiedliche Offerten) anhand des Kriterium des Preises (Profit)

zeigen. Diese müssen nun miteinander verrechnet werden, um die Rangfolge der Alternativen in Bezug auf alle gewählten Kriterien zu erhalten. Es wird dazu für jede einzelne Alternative ein Prozentwert errechnet (wobei sich die Prozentwerte aller Alternativen schliesslich auf 100% ergänzen.) Um beispielsweise den Prozentwert von Alternative 1 zu berechnen, ist folgende Rechnung nötig:

$$p_{Alternative\ A} = \sum_{n=0}^{Anzahl\ Kriterien} w_{Kriterium\ n} * w_{Alternative\ a} \tag{2.1}$$

Um den Prozentwert für eine Alternative zu berechnen, der diese Alternative in Vergleich zu den übrigen Alternativen setzt, muss für sämtliche Kriterien das berechnete Gewicht des Kriteriums aus Matrix 2.1 mit dem berechneten Gewicht der Alternative für das entsprechende Kriterium aus Matrix 2.2 verrechnet werden.

Zur Berechnung eines Gewichtungsvektors aus einer AHP-Entscheidungsmatrix sind verschiedene Verfahren möglich. Saaty [12] schlägt als exakte Variante die Berechnung mittels Eigenvektor vor. Ein Eigenvektor kann jedoch nur bis zu Matrizen der Dimension 4, d.h 4x4 Matrizen, exakt berechnet werden (Satz von Abel-Ruffini). Saaty beschreibt auch alternative Verfahren, welche der vereinfachten Berechnung der Gewichtungsvektoren dienen. In der vorliegenden Arbeit wurde in der Implementierung ein vereinfachtes Verfahren gewählt, aus folgenden Gründen:

- Da in der gewählten Lösung die Entscheidungsmatrizen (siehe Kapitel 5.1.1) nicht empirisch bestimmt, sondern auf Basis des entsprechenden Agententyps geschätzte Präferenzwerte festgelegt wurden, spielen Ungenauigkeiten ab zweiter oder dritter Kommastelle eine vernachlässigbare Rolle.
- Die Implementierung eines vereinfachten Verfahrens bzw. einer Näherungsfunktion benötigt weniger Rechenleistung.

2.5. Der Schweizer Energieholzmarkt

Bevor im nachfolgenden Kapitel auf das konzeptuelle Modell des Schweizer Energieholzmarktes eingegangen wird, werden in diesem Kapitel kurz die wichtigsten Eigenschaften des Schweizer Energieholzmarktes festgehalten. Weitere Eigenschaften des Energieholzmarktes sind in Anhang C aufgelistet.

Intensity of Importance	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective
2	Weak or slight	
3	Moderate importance	Experience and judgement slightly favour one activity over another
4	Moderate plus	
5	Strong importance	Experience and judgement strongly favour one activity over another
6	Strong plus	
7	Very strong or demonstrated importance	An activity is favoured very strongly over another; its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favouring one activity over another is of the highest possible order of affirmation
Reciprocals of above	If activity i has one of the above non-zero numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i	A reasonable assumption
1.1 - 1.9	If the activities are very close	May be difficult to assign the best value but when compared with other contrasting activities the size of the small numbers would not be too noticeable, yet they can still indicate the relative importance of the activites.

Tabelle 2.3.: Die Bedeutung von unterschiedlichen Gewichten in AHP (Quelle: [14])

Abhängig von der Baumart besteht zwischen 10% und 50% der gesamten Holzmasse des Baumes aus Ästen und Blättern, welche nicht als hochqualitatives Rundholz genutzt werden können. Dieses weniger qualitative Holz kann jedoch für die Produktion von Energieholz (Holzschnitzel) oder z.B. Papierholz verwendet werden [7]. Da die Erträge beim Verkauf von Rundholz durch dessen höhere Qualität grösser sind, wird die verfügbare Menge des Nebenproduktes Energieholz hauptsächlich dadurch bestimmt, wie viel Rundholz auf den Markt kommt. Durch steigende Ölpreise wird die Alternative des Heizens mit Energieholz zunehmend attraktiv, was jedoch wiederum auch die Energieholzpreise in die Höhe treibt (wenn auch nicht im selben Ausmass). Dies wiederum macht die Energieholzproduktion für Förster immer lukrativer.

Anbieter von Rundholz sind sowohl Förster wie auch private Waldbesitzer. Diese schliessen sich auch häufig zu kleineren Organisationen zusammen, um damit auf dem Markt eine grössere Verhandlungsmacht zu erhalten. Energieholz wird ebenfalls direkt von den Förstern und privaten Waldbesitzern vertrieben. Ein Teil des auf dem Markt verfügbaren Energieholzes stammt jedoch auch aus Sägewerken, bei welchen z.B. bei der Verarbeitung von Rundholz zu Brettern

immer ein gewisser Teil „Abfallholz“ entsteht, der dann zu Energieholz weiterverarbeitet werden kann.

Abnehmer von Energieholz sind Personen, Gemeinden und Betriebe, welche über eine entsprechende Energieholzheizung verfügen. Das Spektrum reicht hier von kleinen Zimmeröfen bis zu grossen Fernwärmeanlagen, welche eine Vielzahl von Haushalten mit Wärme und Energie versorgen können. Insbesondere die Besitzer der grösseren Heizanlagen schliessen oft Langzeitverträge mit Energieholzlieferanten ab, um ihren Bedarf längerfristig zu sichern. Solche Langzeitverträge haben in der Regel eine Laufzeit von 10-20 Jahren ([7]). Da die Preisentwicklung über eine solch grosse Zeitspanne schwierig vorherzusehen ist, wird in solchen Verträgen normalerweise eine Klausel integriert, welche eine Anpassung des vereinbarten Preises an den aktuellen Ölpreis festhält.

3. Konzeptuelles Modell des Schweizer Energieholzmarktes

In diesem Kapitel wird das von Steubing et al. [16] entwickelte konzeptuelle Modell detaillierter erklärt. Nach einem kurzen Gesamtüberblick im ersten Unterkapitel wird im zweiten Unterkapitel auf die unterschiedlichen Rollen (Akteure) eingegangen. Anschliessend werden im dritten Unterkapitel die verschiedenen Action Situations näher erläutert.

3.1. Action Arena

Gemäss dem für die Entwicklung des konzeptuelle Modells verwendeten MAIA-Frameworks ist die Action Arena der zentrale Ort des Geschehens in der Simulation. In der Action Arena befinden sich alle Agenten der Simulation und interagieren dort miteinander in den verschiedenen Action Situations. Abbildung 3.1 zeigt einen graphischen Überblick über die Akteure und Action Situations.

3.2. Akteure

Nachfolgend werden die verschiedenen Rollen der Akteure und ihre wichtigsten Eigenschaften beschrieben, welche für die Implementierung des konzeptionellen Modells des Schweizer Energieholzmarktes relevant sind:

- **Forester**. Förster sind für Waldstücke zuständig, von denen eine Gemeinde, ein Kanton, eine Bürgergemeinde oder eine sonstige Kooperation der Eigentümer ist. Ihr Aufgabengebiet umfasst dabei unter anderem die Pflege des Waldes sowie die Holzernte.
- **Private Forest Owner**. Im Gegensatz zu den Förstern sind Private Forest Owner selbst die Eigentümer ihres Waldstückes.
- **Forest Company**. Eine Forest Company ist ein Unternehmen, welches einerseits als Zwischenhändler auf dem Holzmarkt agiert, andererseits über Maschinen für die Holzernte verfügt. Förster und Private Forest Owner, welche über keine eigenen Maschinen für die Holzernte verfügen, können eine Forest Company mit der Holzernte beauftragen. Führt eine Forest Company die Holzernte für einen Waldbesitzer durch, übernimmt sie häufig auch gleich den Verkauf des Holzes für den Waldbesitzer.
- **Bundling Organization**. Eine Bundling Organization ist ein Zusammenschluss von mehreren Förstern und/oder Private Forest Owner mit dem Ziel, Ressourcen (wie beispielsweise Maschinen) zu bündeln um dadurch einerseits Kosten zu sparen, andererseits durch das grössere Holzangebot eine stärkere Verhandlungsmacht auf dem Markt zu erhalten. Im Modell wird die Bundling Organization als eigenständiger Akteur dargestellt, d.h. ohne direkte Beziehungen zu einzelnen Waldbesitzern. Auf dem Holzmarkt treten Bundling Organizations als Intermediäre auf.

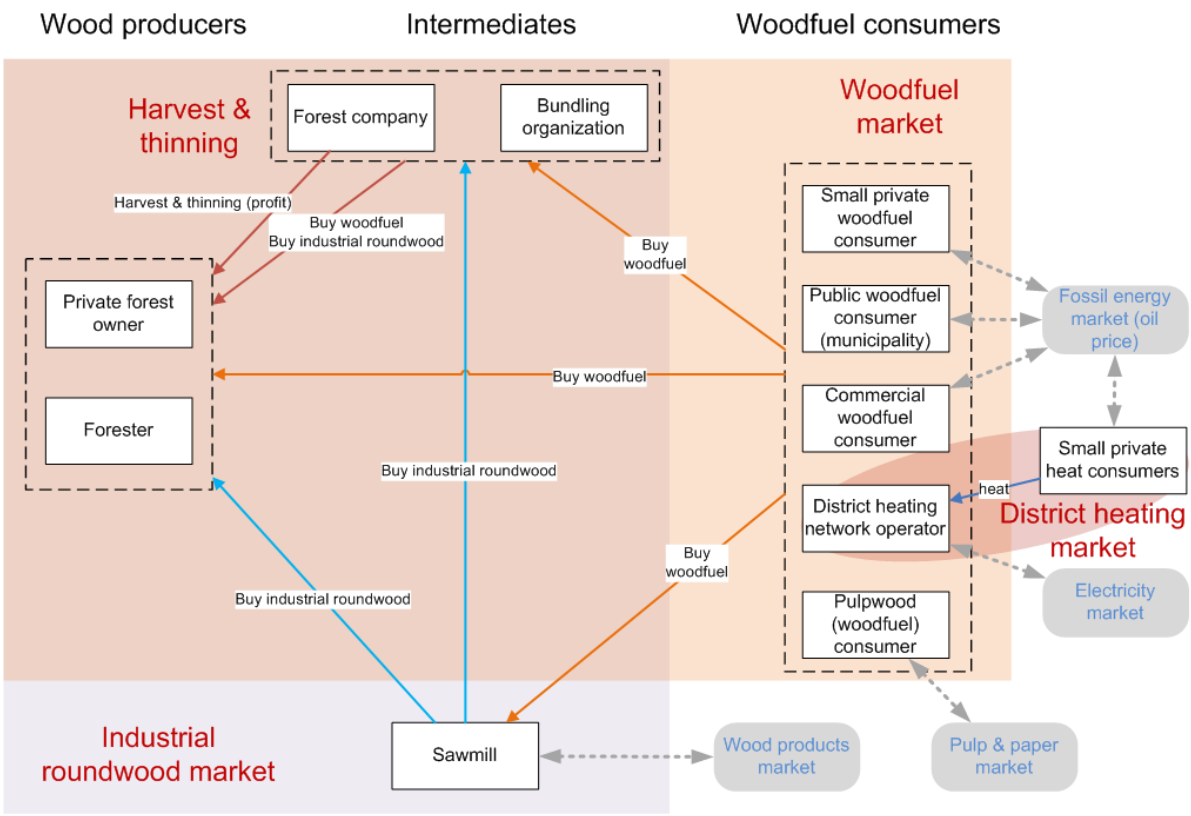


Abbildung 3.1.: Action Arena. Die weissen Kästchen zeigen die verschiedenen Rollen, die farbigen Hinterlegungen die verschiedenen Märkte (Action Situations). Die Pfeile zwischen den verschiedenen Rollen zeigen, welche Rollen mit welchen anderen wie in Verbindung stehen. Die grauen abgerundeten Rechtecke zeigen externe Märkte, welche die Action Arena beeinflussen. (Quelle: Steubing et al.[16])

- **Sawmill**. Eine Sawmill (Sägewerk) verarbeitet Rundholz zu Holzprodukten wie z.B. Brettern. Bei dieser Verarbeitung des Holzes fallen stets ca. 40% Abfallholz[1] an, welches zu Energieholz (Woodfuel) weiterverarbeitet werden kann.
- **Small Private Woodfuel Consumer**. Small Private Woodfuel Consumer sind private Einfamilienhausbesitzer, welche sich für die Installation einer Holzschitzelheizung entschieden haben, und somit einen kontinuierlichen Bedarf an Energieholz haben.
- **Public Woodfuel Consumer**. Public Woodfuel Consumer sind Gemeinden, welche ihre Immobilien mittels Energieholz beheizen. Sie unterscheiden sich von den Small Private Woodfuel Consumer hauptsächlich durch einen grösseren Bedarf an Energieholz.

[1] Quelle: BFS, Eidg. Holzverarbeitungserhubg 2007: *Restholzverwertung in den Sägereien* und *Rundholzeinschnitt in den Sägereien.*

- **Commercial Woodfuel Consumer**. Commercial Woodfuel Consumer sind Unternehmen, welche ihre Immobilien mittels Energieholz beheizen. Sie haben ähnliche Eigenschaften wie die Public Woodfuel Consumer.
- **Pulpwood Consumer**. Pulpwood Consumer verarbeiten Energieholz / Pulpwood (Papierholz) weiter zu Papier.
- **District Heating Network Operator**. Ein District Heating Network Operator ist ein Betreiber eines Fernwärme-Netzes. Über ein Fernwärmenetz kann er innerhalb eines bestimmten Radius Immobilien mit Wärme versorgen. Bei der Verarbeitung von Energieholz zu Wärme erzeugt der District Heating Network Operator auch Strom, welcher er in das Stromnetz einspeisen kann.
- **Small Private Heat Consumer**. Small Private Heat Consumer sind private Hausbesitzer, welche an ein von einem District Heating Network Operator betriebenes Fernwärmenetz angeschlossen sind.

In Tabelle 3.1 sind die verschiedenen Agententypen mit ihren Rollen innerhalb der verschiedenen Märkte aufgeführt.

Rolle	Industrial Roundwood Market	Woodfuel Market	Harvest-and-Thinning Market	District Heating Market
Forester	Verkäufer	Verkäufer	Käufer	-
Private Forest Owner	Verkäufer	Verkäufer	Käufer	-
Forest Company	Intermediär	Intermediär	Verkäufer	-
Bundling Organization	Intermediär	Intermediär	-	-
Sawmill	Käufer	Verkäufer	-	-
Small Private Woodfuel Consumer	-	Käufer	-	-
Public Woodfuel Consumer	-	Käufer	-	-
Commercial Woodfuel Consumer	-	Käufer	-	-
Pulpwood Consumer	-	Käufer	-	-
District Heating Network Operator	-	Käufer	-	Verkäufer
Small Private Heat Consumer	-	-	-	Käufer

Tabelle 3.1.: Die verschiedenen Rollen und ihr Bezug zu den verschiedenen Märkten

3.2.1. Entscheidungsverhalten der Akteure

Die verschiedenen Akteure unterscheiden sich auch durch ihr Entscheidungsverhalten. Folgende sind die wichtigsten Kriterien, welche die Akteure berücksichtigen, wenn sie vor einer Kauf- oder Verkaufsentscheidung stehen:

- **Profit**. Der Akteur versucht eine möglichst grossen Gewinn zu erzielen (Verkäufer), bzw. möglichst kostengünstig seine Einkäufe zu erledigen (Käufer).

- **Umweltbewusstsein**. Der Akteur hat ein grosses Umweltbewusstsein und versucht deshalb, Holz möglichst regional zu kaufen bzw. verkaufen.
- **Freundschaften**. Der Akteur deckt seinen Holzbedarf beim Verkäufer seines Vertrauens.

Verschiedene Akteure der gleichen Rolle gewichten diese Kriterien teilweise unterschiedlich. Bei einigen Rollen gibt es jedoch innerhalb der Rolle nur ein einziges Entscheidungsverhalten, d.h. die Kriteriengewichtung ist immer gleich. So wird beispielsweise bei kommerziellen Betrieben angenommen, dass sie stets in erster Linie profitorientiert arbeiten und handeln. Bei anderen Rollen gibt es innerhalb der Rolle unterschiedliche Entscheidungsverhalten. Dies ist beispielsweise bei Privatpersonen der Fall, die kostengünstiges Heizen und Umweltschutz unterschiedlich gewichten. Tabelle 3.2 gibt einen Überblick über die unterschiedlichen Entscheidungsverhalten pro Rollentyp.

3.3. Action Situations

In der Action Arena finden verschiedene Action Situations statt. Diese werden in den folgenden Unterkapiteln erklärt. Grundsätzlich lassen sich die Action Situations in zwei Gruppen einteilen: die einen finden monatlich statt, die anderen nur einmal pro Jahr. Eine Übersicht über die Ablaufreihenfolge der einzelnen Action Situations ist in Abbildung 3.2 ersichtlich.

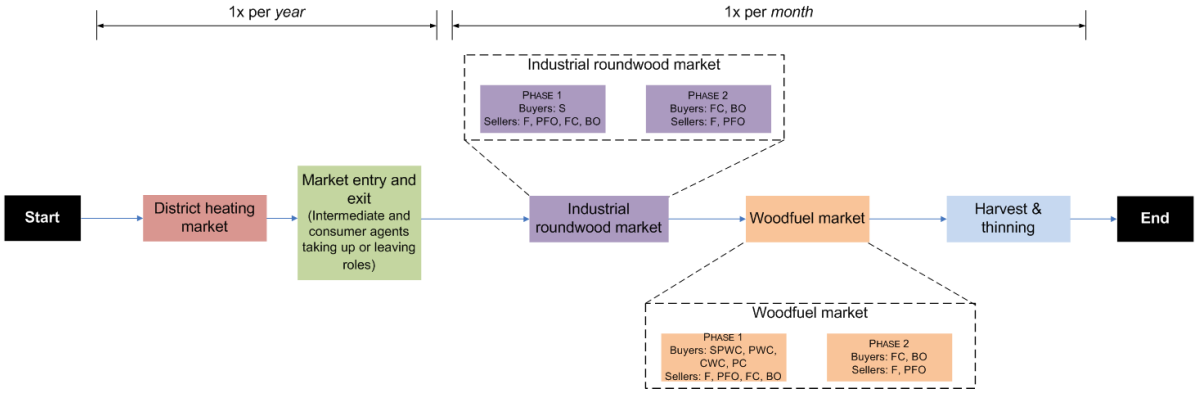


Abbildung 3.2.: Die Reihenfolge der verschiedenen Action Situations (Quelle: Steubing et al. [16]).

3.3.1. District Heating Market

Der District Heating Market ist die erste Action Situation und findet jeweils zu Beginn des Jahres statt. In dieser Action Situation beziehen alle Small Private Heat Consumer ihren Energiebedarf für das kommende Jahr, das heisst sie erneuern ihren Vertrag mit ihrem District Heating Network Operator, damit die Versorgung mit Fernwärme weiterhin sichergestellt ist.

Rolle (Subtyp)	Kriterium 1	Kriterium 2	Kriterium 3
Forester			
Conservative	Freundschaft	Umwelt	Profit
Environmentally Oriented Proactive	Profit	Umwelt	Freundschaft
Profit Oriented	Profit	Freundschaft	Umwelt
Educated Professional	Profit	Umwelt	Freundschaft
Private Forest Owner			
Farmer Type	Umwelt	Profit	Freundschaft
Recreational Type	Freundschaft	Umwelt	Profit
Profit Type	Profit	Freundschaft	Umwelt
Bankster Type	Profit	Freundschaft	Umwelt
Forest Company			
General Type	Profit	Umwelt	Freundschaft
Bundling Organization			
General Type	Profit	Umwelt	Freundschaft
Sawmill			
General Type	Profit	Umwelt	Freundschaft
Small Private Woodfuel Consumer			
Greedy Type	Profit	Umwelt	Freundschaft
Environmental Type	Umwelt	Profit	Freundschaft
Public Woodfuel Consumer			
General Type	Profit	Umwelt	Freundschaft
Commercial Woodfuel Consumer			
General Type	Profit	Umwelt	Freundschaft
Pulpwood Consumer			
General Type	Profit	Umwelt	Freundschaft
District Heating Network Operator			
General Type	Profit	Umwelt	Freundschaft
Small Private Heat Consumer			
Greedy Type	Profit	Umwelt	Freundschaft
Environmental Type	Umwelt	Profit	Freundschaft

Tabelle 3.2.: Die verschiedenen Rollen mit ihren Entscheidungsverhalten. Die Tabelle zeigt nur die Reihenfolge der Kriterien in der Stärke ihrer Gewichtung, deshalb haben beispielsweise bei den Förstern zwei verschiedene Subtypen die selbe Kriterienreihenfolge; sie entscheiden sich jedoch in der Gewichtung der Kriterien. Das exakten Gewichte, welche in der Simulation verwendet wurden, sind in Kapitel 5.1.1 beschrieben.

3.3.2. Market Entry and Exit

Die Action Situation *Market Entry and Exit* findet wie auch der District Heating Market nur einmal pro Jahr statt. Market Entry and Exit ist die einzige Action Situation, die keinen Markt darstellt. In dieser Action Situation entscheiden Agenten, ob sie in einem bestimmten Markt eine Rolle aufnehmen, oder ihre aktuelle Rolle ablegen. Beispielsweise entscheiden sich hier Small Private Woodfuel Consumer, ob sie ihre Energieholz-Heizung erneuern, bzw. falls sie bisher noch keine Energieholz-Heizung hatten, ob sie eine in ihr Haus installieren wollen. Die Entscheidung basiert aufgrund der wahrgenommenen Attraktivität des Energieholz-Marktes gegenüber anderen Energiemärkten wie z.B. dem Heizölmarkt, d.h. der Agent wägt aufgrund seiner Präferenzen bezüglich Kosteneinsparungen und Umweltbewusstsein ab, ob sich ein Umstieg auf eine Energieholzheizung lohnt.

Neben Markteintritten spielen in dieser Action Situation auch Marktaustritte eine Rolle. Marktaustritte geschehen einerseits dadurch, dass Agenten auf die Erneuerung ihrer Energieholzheizung verzichten, und z.B. eine Ölheizung installieren. Andererseits können einzelne Agenten, insbesondere die kommerziellen (z.B. Sawmill), Konkurs gehen. Förster und private Waldbesitzer können jedoch nicht Konkurs gehen, d.h. sie verlassen den Markt nie.

3.3.3. Industrial Roundwood Market und Woodfuel Market

Sowohl der Industrial Roundwood Market als auch der Roundwood Market finden monatlich statt. Diese beiden Action Situations sind in der zeitlichen Abfolge die ersten zwei der drei monatlich stattfindenden Action Situations. Sie finden in zwei Phasen statt (siehe Abbildung 3.3). Die zwei Phasen unterscheiden sich dadurch, dass pro Phase jeweils verschieden Agententypen (Rollen) als Käufer bzw. Verkäufer auf dem Markt auftreten. In der ersten Phase bestehen die Käufer aus Endverbrauchern, die Verkäufer sind sowohl die Holzproduzenten (Waldbesitzer) wie auch die Intermediäre (Bundling Organization und Forest Company). In der zweiten Phase treten als Käufer ausschliesslich Intermediäre auf, während die Verkäufer ausschliesslich die Waldbesitzer sind. Der Grund für die Aufteilung des Marktes in diese zwei Phasen ist die Sonderrolle der Intermediäre in diesen beiden Märkten. Es wird vereinfachend angenommen, dass Intermediäre nur Holz einkaufen, für welches sie auch einen Abnehmer haben. Somit verkaufen sie im Modell in der ersten Phase Holz, welches sie zu diesem Zeitpunkt noch nicht besitzen. In der zweiten Phase kaufen sie dann das Holz ein, welches sie in Phase 1 verkauft haben. Um sicherzustellen, dass in einer Phase möglichst alle Agenten ihren Bedarf decken können, wird eine Phase jeweils mehrmals wiederholt, bevor die nächste beginnt.

Der Ablauf einer einzelnen Phase (Verhandlungsrunde) ist in Abbildung 3.4 ersichtlich. Eine Verhandlungsrunde läuft in vier Teilschritten ab, wobei jeder Teilschritt erst für sämtliche Marktteilnehmer durchgeführt wird, bevor der nächste Teilschritt beginnt.

- **Schritt 1**. Der erste Schritt geht von den Käufern aus. Sie bestimmen ihren Bedarf und versenden für diesen Bedarf Anfragen an einige potentielle Verkäufer.
- **Schritt 2**. Die Verkäufer evaluieren die Anfragen, die sie erhalten haben. Falls sie die gefragte Menge liefern können und wollen, beantworten sie die Anfrage mit einer Offerte. Andernfalls lehnen sie die Anfrage ab.

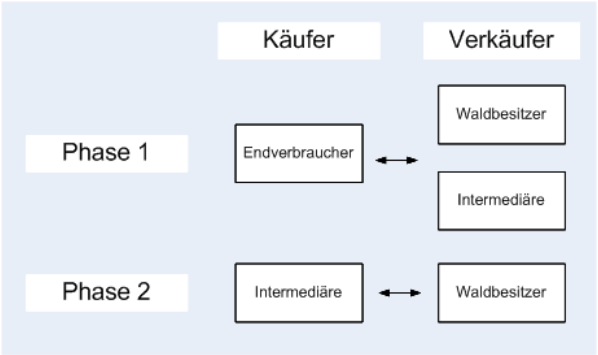


Abbildung 3.3.: Die zwei Phasen des Industrial Roundwood Markets und des Woodfuel Markets

- **Schritt 3**. Die Käufer vergleichen die Offerten, die sie erhalten haben, miteinander. Sie wählen die besten davon aus und senden sie unterschrieben zurück. Die übrigen lehnen sie ab.
- **Schritt 4**. Alle unterschriebenen Verträge werden erfüllt, d.h. es findet ein Tausch des gewünschten Gutes gegen Geld statt.

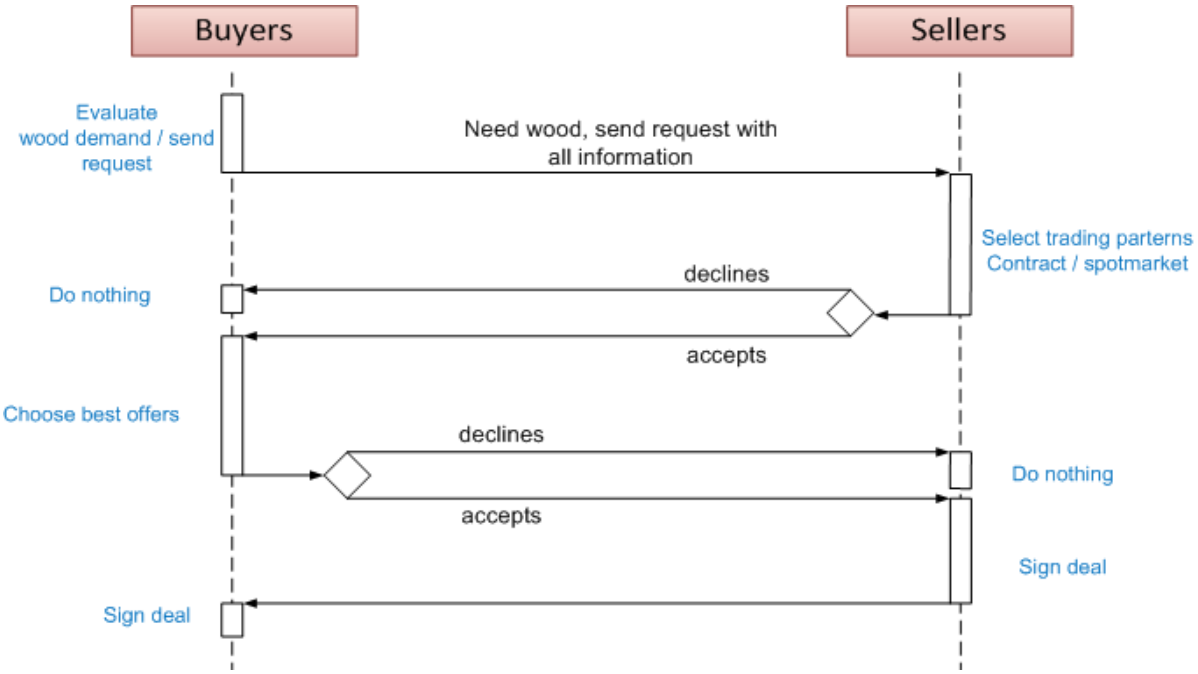


Abbildung 3.4.: Der Ablauf einer einzelnen Verhandlungsrunde

Die Abläufe im Woodfuel Market und im Industrial Roundwood Market sind praktisch identisch. Jedoch gibt es im Woodfuel Market im Gegensatz zum Industrial Roundwood Market auch Langzeitverträge. Langzeitverträge werden häufig z.B. zwischen einer Gemeinde (Public

Woodfuel Consumer) und einer Bundling Organization abgeschlossen. Sie haben den Zweck, sowohl dem Abnehmer eine längerfristige Versorgung, als auch dem Verkäufer eine längerfristig gesicherten Absatz zu garantieren. Langzeitverträge haben im Modell jeweils eine Laufdauer von 10 Jahren. In dieser Zeit wird z.B. monatlich eine bestimmte Menge Energieholz geliefert. Der Preis dafür wird zum Zeitpunkt des Vertragsabschlusses festgelegt. Dabei wird in der Regel eine Preisanpassungsklausel vereinbart, in der festgehalten wird, dass sich der Preis für das Energieholz zu einem gewissen Prozentsatz dem aktuellen Ölpreis anpasst. Damit kann für den Verkäufer das Risiko vermindert werden, dass er bei stark steigenden Energiepreisen sein Holz plötzlich massiv unter den aktuellen Marktpreisen verkaufen muss.

3.3.4. Harvest and Thinning Market

Der Harvest and Thinning Market ist die dritte Action Situation, welche monatlich ausgeführt wird. Er wird im Gegensatz zum Industrial- und dem Roundwood Market nur in einer Phase ausgeführt, da es in diesem Markt keine Intermediäre gibt. Im Harvest and Thinning Market wird im Gegensatz zu den anderen Märkten kein eigentliches Gut, sondern eine Dienstleistung gehandelt. Ein privater Waldbesitzer bespielsweise, der über keine eigenen Maschinen für die Holzernte verfügt, kann einen Harvest and Thinning Anbieter (z.B. eine Forest Company) beauftragen, die Holzernte in seinem Waldstück durchzuführen.

4. Architektur-Design und Implementierung

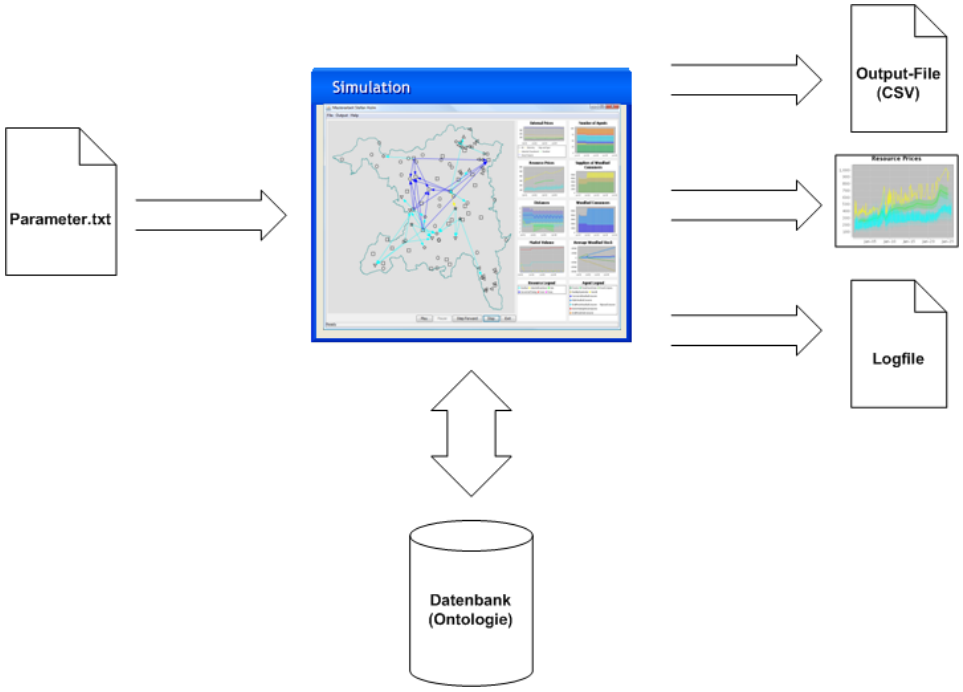


Abbildung 4.1.: Grobübersicht Architektur

Abbildung 4.1 zeigt eine Grobübersicht der Architektur. Als Eingabedaten für die Simulation dient eine Textdatei, welche Parameter für die Simulation enthält[1]. Während der Simulation interagiert das Simulationsprogramm mit einer Ontologie-Datenbank. Alle Informationen über die Agenten werden während der Simulation in diese Datenbank gespeichert. Auch werden bei Bedarf Informationen über Agenten aus der Datenbank gelesen; die Datenbank dient also während der Simulation der Verwaltung der Agenten. Auf der Ausgabeseite stehen drei unterschiedliche Arten von Ausgabedaten. Einerseits werden Diagramme ausgegeben, d.h. die wichtigsten Metriken der Simulation werden direkt in eine Bilddatei geschrieben. Zusätzlich werden die Rohdaten für diese Diagramme in einer CSV[2]-Datei gespeichert, welche im Anschluss an die Simulation beispielsweise mit Excel detailliert ausgewertet werden kann. Zusätzliche Informationen liefert das Logfile; dorthin werden Informationen über den Programmablauf geschrieben. Die Granularität der ins Logfile geschriebenen Informationen kann zum Start der Simulation

[1] In den Zielsetzungen der Arbeit in Kapitel 1.1 wurde erwähnt, dass die Eingabedaten des Simulationsprogramms Rollen, Akteure und Szenarien sein sollen. Ein Szenario wird in der Parameterdatei definiert. Die Rollen und Akteure als Eingabedaten sind in der Abbildung nicht dargestellt, da diese innerhalb des Quellcodes definiert werden. Das Hinzufügen von Rollen benötigt eine Neukompilierung des Simulationsprogramms.

[2] CSV: comma separated value

festgelegt werden. In den nachfolgenden Unterkapiteln wird die Architektur detaillierter beschrieben.

4.1. Layer

Die Architektur des Simulationsprogramms ist in vier Schichten aufgeteilt (siehe Abbildung 4.2). Die unterste Schicht (Schicht 1) besteht aus der Ontologie-Datenbank sowie den CSV-Dateien, welche unter anderem historische Preisdaten und Koordinaten von Forstgebieten enthalten. Schicht 2 dient der vereinfachten Interaktion des Simulationsprogramms mit der Datenbank. In der dritten Schicht finden alle Interaktionen der Agenten statt. Die oberste Schicht, Schicht 4, ist das GUI, welches per Observer-Pattern auf die Informationen in den weiter unten liegenden Schichten zugreift. Eine Schicht greift immer nur auf die nächstuntere Schicht zu, um z.B. Informationen über Agenten zu erhalten, oder mit der Datenbank zu interagieren. Ausnahmen bilden SPARQL-Queries, die immer direkt auf die unterste Schicht - die Datenbank-Schicht - zugreifen. So holt sich beispielsweise das GUI-Layer vereinzelt Informationen direkt per SPARQL-Query aus der Datenbank.

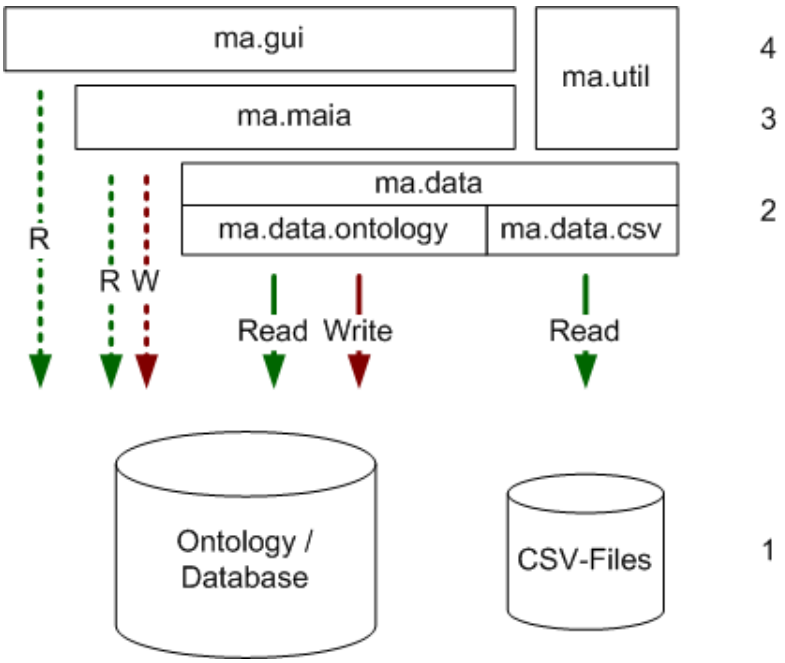


Abbildung 4.2.: Die vier Schichten des Simulationsprogramms

4.2. Architektur

Abbildung 4.3 zeigt ein stark vereinfachtes Klassendiagramm. Das Main-Package *ma* (Masterarbeit) besteht aus vier Unter-Packages:

- **ma.gui:** Dieses Package beinhaltet alle Klassen, die zum GUI gehören. Neben dem Hauptfenster sind dies hauptsächlich die verschiedenen Diagramme. Wird das Simulationsprogramm im Konsolen-Modus gestartet (siehe Kapitel 4.4.2), werden diese Klassen nicht benötigt.[1]

- **ma.maia:** Die in diesem Package enthaltenen Klassen beinhalten die Funktionen für die Interaktion zwischen den verschiedenen Agenten.

- **ma.data:** Dieses Package dient als Schnittstelle zwischen der Applikation und der Datenbank. Die Datenbank besteht einerseits aus der Ontologie-Datenbank, andererseits aus diversen CSV-Dateien, welche Daten enthalten, die für die Simulation relevant sind. Dies beinhaltet sowohl Preisdaten (historische Ölpreise, Holzpreise etc.), wie auch Daten über die Agenten (z.B. Forstreviere Kt. Aargau). Im Unterpackage ma.data.ontology sind die Klassen für die Interaktion mit der Ontologie-Datenbank definiert. Dies beinhaltet Klassen für die Erstellung der Datenbank (Erstellung der Agenten mit ihren Eigenschaften), sowie auch Klassen für die Interaktion mit der Datenbank.

- **ma.util:** Dieses Package beinhaltet diverse Hilfsklassen wie beispielsweise die Klassen Timer, welche zur Zeitmessung von einzelnen Methoden dient (was während der Entwicklung zur Evaluierung unterschiedlicher Algorithmen aufgrund deren Performance diente), oder VectorRandomizer, welche die von einer Vector-Instanz beinhalteten Objekte in eine neue, zufällige Reihenfolge bringt (wird z.B. benötigt, damit Verkäufer ihre Anfragen in einer zufällig gegebenen Reihenfolge beantworten können). Besonders erwähnenswert ist hier das Unterpackage ma.util.decision.ahp, welches die benötigten Klassen für die AHP-Methode (siehe Kapitel 2.4) zur Verfügung stellt.

4.3. Implementierung

Die aktuelle Version des Simulationsprogramms erstreckt sich über etwas mehr als 10'000 Zeilen Programmcode, aufgeteilt in 19 Packages und 87 Klassen. Eine detaillierte Erklärung aller Details der gewählten Implementierung würde den Rahmen dieser Masterarbeit sprengen. Deshalb wird in den nachfolgenden Unterkapiteln, nebst einer kurzen Erklärung der generellen Funktionsweise, ausschliesslich auf einige interessante Details der Implementierung eingegangen.

4.3.1. Generelle Funktionsweise

Entsprechend dem in Kapitel 3 gezeigten konzeptuellen Modell des Schweizer Energieholzmarktes, laufen Monat für Monat verschiedene Action Situations ab. Die Klasse ActionArena (siehe auch Abbildung 4.3) kennt die genaue Reihenfolge sowohl der monatlich wie auch der jährlich stattfindenden Action Situations. Diese Reihenfolge ist in der Ontologie-Datenbank gespeichert und wird von dort zu Beginn der Simulation ausgelesen. Für jede Action Situation werden schliesslich erst die potentiellen Käufer geladen, damit sie ihre Anfragen an potentielle Verkäufer senden können. Haben alle Käufer ihre Anfragen abgeschickt, werden die Verkäufer geladen,

[1] Falls mittels entsprechend gesetztem Programmargument Diagramme als Bilddateien ausgegeben werden sollen, wird das Package ma.gui.statistics teilweise verwendet.

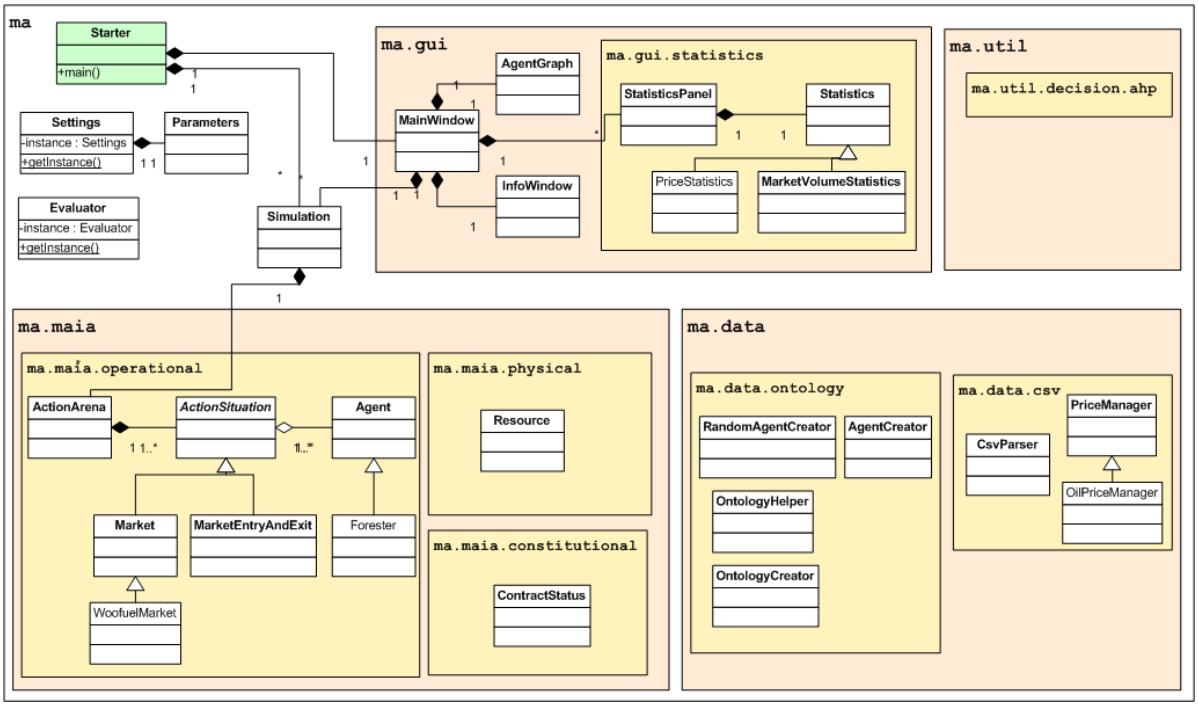


Abbildung 4.3.: Stark vereinfachtes Klassendiagramm der Architektur. Aus Gründen der Übersichtlichkeit wurden nur die wichtigsten Klassen und Assoziationen dargestellt.

damit sie die Anfragen beantworten können etc. (der Rest der Interaktion erfolgt analog dem in Kapitel 3.3.3 erklärten Ablauf der Verhandlungsrunden). Das Laden eines Agenten funktioniert durch die Verwendung einer sog. Wrapper-Klasse: Aus der Datenbank wird vorerst nur die eindeutige ID des Agenten geladen und in ein Java-Objekt (eine Instanz der Wrapper-Klasse) gepackt. Die übrigen Informationen über den Agenten bleiben in der Datenbank. Die Wrapper-Klasse stellt nun verschiedene Methoden zur Verfügung, wie z.B. eine Methode *makeRequests()*, die dazu dient, dass ein Agent anhand seines aktuellen Bedarfs Anfragen an potentielle Verkäufer verschickt. Wird diese Methode aufgerufen, werden die für die Methode benötigten Daten aus der Datenbank ausgelesen (in diesem Beispiel der aktuelle Bedarf sowie eine Liste von Verkäufern in der Nähe). Anschliessend werden den ausgelesenen Verkäufern die Anfragen zugeschickt, indem direkt Contract-Instanzen[1] in die Ontologie-Datenbank geschrieben werden. Mit diesem Vorgehen wird verhindert, dass Informationen gleichzeitig sowohl in der Ontologie-Datenbank als auch als Java-Objekte vorhanden sind (ausgenommen die ID des Agenten), da dies zu unerwünschten Redundanzen führen würde.

Erwähnenswert ist hierbei auch noch die Bedarfsbestimmung des Agenten. Ein Small Private Woodfuel Consumer beispielsweise verbraucht monatlich eine gewisse Menge an Energieholz. Neigt sich sein Vorrat an Energieholz dem Ende zu, muss er neues Energieholz kaufen: Es entsteht ein Bedarf. Um diesen Vorgang im Simulationsprogramm abbilden zu können, werden jeweils zu Beginn jedes simulierten Monats, noch vor Beginn der ersten Action Situation,

[1] Das komplette Datenbankschema der Ontologie-Datenbank ist in Anhang B abgebildet.

sämtliche Agenten aus der Datenbank geladen. In allen Wrapper-Klassen der verschiedenen Agententypen gibt es eine Methode *produce()*, welche nun für jeden geladenen Agenten aufgerufen wird. In dieser Methode wird nun - betrachtet man wiederum das Beispiel des Small Private Woodfuel Consumers - eine gewisse Menge Energieholz aus seinem Ressourcen-Portfolio entfernt: diejenige Menge, welche er zur Beheizung seines Hauses benötigt. Zusätzlich erhält er einen „Lohn“, d.h. es wird ihm ein gewisser Geldbetrag gutgeschrieben, mit dem er später Ressourcen kaufen kann, oder falls seine Energieholzheizung das Ende ihrer Lebensdauer erreicht, sie erneuern lassen. Schliesslich wird auch überprüft, wie gross sein Vorrat an Energieholz noch ist. Abhängig davon bleibt der Bedarf bei 0 (d.h. er will im aktuellen Monat noch nicht einkaufen) oder er wird auf die gewünschte Einkaufsmenge gesetzt. Die produce-Methode ist für jede Rolle (Förster, Sägewerk etc.) unterschiedlich. So werden bei einem Sägewerk beispielsweise monatlich Fixkosten abgezogen, während ein District Heating Network Operator Energieholz in Wärme und Elektrizität umwandelt, damit er diese weiterverkaufen kann. Die produce-Methode ist also für die monatlich wiederkehrenden Aufgaben der Agenten zuständig. Der Inhalt dieser Methoden macht einen Grossteil der getroffenen Annahmen (siehe Anhang A) aus.

4.3.2. Multithreading

Teile des Simulationsprogramms sind Multithreading-fähig. Es wurde darauf verzichtet, das komplette Simulationsprogramm Multithreading-fähig zu implementieren, da beschlossen wurde, die Simulation auf dem High-Performance-Cluster nur in einem Thread laufen zu lassen. Möchte man beim verwendeten Cluster eine Applikation auf mehreren Prozessoren parallel laufen lassen, muss die gewünschte Anzahl Prozessoren vorgängig reserviert werden. Durch die dadurch entstehende Wartezeit ist die Gesamtzeit für eine Simulation dann mit hoher Wahrscheinlichkeit grösser, als wenn jede Simulation nur auf einem Prozessor läuft. Trotzdem wurde ein einzelner, besonders rechenintensiver Teil des Simulationsprogramms Multithreading-fähig programmiert. Es konnte gezeigt werden, dass wenn man Multithreading gezielt in rechenintensiven Teilen einsetzt, ein Zeitvorteil entsteht. Der Programmieraufwand hält sich dagegen eher klein; Jena, das verwendete Framework für die Interaktion mit der Ontologie-Datenbank (siehe Kapitel 4.5.1), unterstützt Multithreading. Um Probleme mit der Nebenläufigkeit von Datenbank-Interaktionen zu verhindern, steht in Jena das Prinzip *Multiple-Reader / Single-Writer* (MRSW) zur Verfügung. Das heisst, es können zur gleichen Zeit problemlos mehrere Threads lesen, sobald jedoch ein Thread schreibend auf die Datenbank zugreift, werden für den Zeit des Zugriffs alle anderen Anfragen gesperrt (d.h. müssen warten).

Es hat sich während der Implementierung herausgestellt, dass häufig die Lesezugriffe diejenigen sind, welche viel Zeit in Anspruch nehmen. So werden beispielsweise Durchschnittspreise über eine grosse Anzahl Verträge berechnet, was wesentlich aufwändiger ist als das Schreiben eines einzelnen Vertrages in die Datenbank. Insofern ist es ideal, wenn somit die zeitaufwändigen Lesezugriffe parallel abgearbeitet werden können, während die kürzeren Schreibvorgänge sequentiell ausgeführt werden. Ein weiterer Vorteil, der für die Verwendung von Multithreading im Simulationsprogramm spricht, sind die Abläufe der einzelnen Verhandlungsrunde, die streng sequentiell ablaufen (vgl. Kapitel 3.3): Erst geben alle Agenten ihre Anfragen an Lieferanten ab, und erst dann geben alle Lieferanten ihre Offerten ab. Somit können erst die Agenten in verschiedene Threads aufgeteilt werden, wo sie ihre Anfrage absenden. Sobald alle Threads

beendet sind, können die verkaufenden Agenten in verschiedene Threads aufgeteilt werden etc. Somit kommt es zu keinen Nebenläufigkeitsproblemen.

Effektiv eingesetzt wurde Multithreading bei der Erstellung des „Telefonbuchs“ der Agenten. Da die Distanzen zwischen Käufer und Verkäufer im Holzmarkt eine sehr grosse Rolle spielt, versuchen Käufer ihre benötigten Produkte möglichst in der Nähe zu kaufen. Aus algorithmischer Sicht bedeutet dies, dass ein Käufer erst alle potentiellen Verkäufer aus der Datenbank auslesen muss, dann zu jedem die Distanz berechnen (Pythagoras), und schliesslich alle Verkäufer nach ihrer Distanz sortieren muss. Diese Operationen kosten sehr viel Rechenleistung, insbesondere bei einer hoher Anzahl Agenten. Dies sieht man auch, wenn man die Komplexitätsklassen der einzelnen Schritte anschaut:

- Suche nach Verkäufer: O(n)
- Pythagoras: O(n)
- Sortieren: O(n * log(n))

Folglich ist es das Sortieren, welche den Grossteil der Zeit für die Suche nach geeigneten Verkäufern ausmacht. Dies wurde auch während der Implementierung so festgestellt: Sucht man nach allen Verkäufern, berechnet die Distanz zu ihnen, und *filtert* dann zuerst die Distanzen über einem gewissen Grenzwert, geht die Suche nach Verkäufern wesentlich schneller. Um eine bestimmte Anzahl an Verkäufern zu erhalten, kann dies mehrmals hintereinander mit wachsendem Grenzwert durchgeführt werden, bis die gewünschte Anzahl Agenten erreicht ist.

Jedoch ist auch mit dieser Vereinfachung der Rechenaufwand noch sehr hoch, da bei jeder Verhandlungsrunde erneut die selben Berechnungen durchgeführt werden müssen. Folglich wurde in die Ontologie-Datenbank für jeden Käufer ein Telefonbuch erstellt, welches eine bestimmte Anzahl Verkäufer in seiner Nähe beinhaltet. Da jeweils nur zu Jahresbeginn Agenten in den Markt ein- oder austreten können, müssen die entsprechenden Berechnungen nur noch einmal pro simuliertem Jahr durchgeführt werden.

Schliesslich wurde dieser Ansatz noch mit Multithreading kombiniert (aufgrund der Cluster-Anforderungen ist die Multithreading-Option per Programm-Argument ausschaltbar bzw. auf eine gewisse Anzahl Prozessoren begrenzbar): Da das Schreiben eines errechneten Telefonbuchs mit den entsprechenden Anbietern im Vergleich zur Suchzeit nach passenden Anbietern tief ist, lohnt sich hier der MRSW-Ansatz von Jena. Im Code sieht das Ganze wie folgt aus (vereinfachtes Beispiel):

```
private void createPhoneBookForAgent(String agentUri) {
   OntModel model = OntologyHelper.getInstance().getModel();

   //Verkäufer in der Nähe auslesen -> Read-Lock beantragen!
   model.enterCriticalSection(Lock.READ);
   Vector<Agent> localVendors = getLocalVendorsUri(agentUri);
   model.leaveCriticalSection();

   //Erhaltene Verkäufer ins Telefonbuch schreiben -> Write-Lock beantragen!
```

```
    model.enterCriticalSection(Lock.WRITE);
    writePhoneBook(agentUri, market.getLocalName(), localVendors);
    model.leaveCriticalSection();
}
```

Wird diese Methode mit Hilfe von vier parallelen Threads ausgeführt, wobei jeder jeweils den nächsten Agenten der Methode übergibt, ergab diese Gegenüber der Berechnung in einem einzelnen Thread eine Zeitersparnis von 50%.

4.3.3. CSV-Dateien mit externen Preisen

Für die Darstellung des *External Price*-Diagramms (siehe Kapitel 4.4.1) werden die historischen Preise von verschiedenen Gütern benötigt. Diese werden aus CSV-Dateien geladen. Die folgenden Daten wurden verwendet:

- Ölpreise (1986 - 2011, monatlich, in USD/Barell)[1]
- Energieholz-Preise (2000 - 2010, Werte im 4-Monats-Intervall)[2]
- Preise für Holzprodukte (2000 - 2010, Werte im 4-Monats-Intervall)[3]
- Pulp-and-Paper-Preise (2000 - 2010, Werte im 4-Monats-Intervall)[4]
- Rundholz-Preise (2000 - 2010, Werte im 4-Monats-Intervall)[5]
- Elektrizitätspreise (1993 - 2009, jährlich)[6]

Wenn nun eine Simulation im Jahr 2001 beginnt und bis 2025 laufen soll, werden in denjenigen Jahren, für welche reelle Preise vorhanden sind, diese für die Simulation verwendet. Ab dem Monat, in welchem die Daten nicht mehr vorhanden sind, werden die Daten als konstant angenommen, ausgehend vom letzten vorhandenen Wert. Da insbesondere für den Fall des Energieholzmarktes der Ölpreis von grosser Relevanz ist, kann der Preisverlauf auch variiert werden, indem im Parameter-File (siehe Kapitel 4.3.4) Werte für die monatliche Preissteigerung festgelegt werden. Die Preissteigerung kann durch einen relativen Wert (z.B. monatlich +0.5%) und durch einen absoluten Wert (z.B. monatlich + CHF 2) festgelegt werden.

4.3.4. Parameter-File

Das Simulationsprogramm bietet die Möglichkeit, gewisse Input-Parameter, wie z.B. der Verlauf des Ölpreises, zu variieren. Dies dient der vereinfachten Simulation von verschiedenen Szenarien: Die wichtigsten Parameter müssen nicht im Code geändert werden, sondern können in einem Parameter-File variiert werden, welches beim Start des Simulationsprogramms eingelesen wird. Abbildung 4.4 zeigt einen Ausschnitt aus einem solchen Parameter-File.

[1] Quelle: US Energy Information Administration
[2] Quelle: BFS, Produzentenpreise Holzschnitzel Schweiz (72% Nadelholz, 28% Laubholz)
[3] Quelle: BFS, Produzentenpreise Schnittholz Schweiz (Konstruktionsholz)
[4] Quelle: BFS, Produzentenpreise Rohholz Schweiz (Papierholz, Nadel)
[5] Quelle: BFS, Produzentenpreise Säge-Rundholz Schweiz (72% Fichte, 28% Buche)
[6] Quelle: BFS, Schweizerische Elektrizitätsstatistik 1998 - 2009

```
%
% PARAMETER FILE
% ==============
%
% TITLE: probabilityDecBehSpwcGreedy high
%
%
% PROBABILITY OF AGENT TYPES
%
probabilityDecBehSpwcGreedy 0.90f
probabilityDecBehSpwcEnvironm 0.10f
```

Abbildung 4.4.: Auszug aus einer Parameter-Datei. Kommentarzeilen beginnen mit dem %-Zeichen. In dieser Datei, welche für die Sensitivitätsanalyse verwendet wurde, wird die Wahrscheinlichkeit, dass ein Agent mit der Rolle *Small Private Woodfuel Consumer* (Spwc) den Greedy-Subtyp annimmt, von 50% auf 90% erhöht (Gleichzeitig muss die Wahrscheinlichkeit für den anderen Subtyp so verringert werden, damit die Summe der Subtypen wieder 100% ergibt.).

4.3.5. Agent-Upscaling

Steubing et al. ([16]) legten für eine Simulation des Energieholzmarktes des Kanton Aargau pro Rolle jeweils die Anzahl Agenten zu Beginn der Simulation, sowie die maximale Anzahl Agenten, die während einer Simulation erreicht werden kann, fest. Diese Zahlen sind in den ersten beiden Spalten von Tabelle 4.1 ersichtlich. Zu Simulationsbeginn wären somit rund 25'000 Agenten simuliert worden. Im Laufe der Entwicklung des Simulationsprogramms hat sich herausgestellt, dass das Simulationsprogramm bis maximal 2000 Agenten effizient und ohne Speicherprobleme läuft[1]. Auch hat sich gezeigt, dass sich in den 25 Jahren Simulationszeitraum in keinem Szenario die Agentenzahl mehr als verdoppelt hat. Somit musste ein Weg gefunden werden, dass die Simulation mit maximal 1000 Agenten gestartet werden kann, um Speicherprobleme zu verhindern.

Dazu wurden Agenten von Rollen, die eigentlich eine sehr hohe Anzahl Agenten aufweisen würden, zusammengefasst. Der *Upscale-Faktor* (siehe Tabelle 4.1) definiert pro Rolle, wie viele Agenten zu einem einzelnen (jedoch mit entsprechend höherem Bedarf an Rohstoffen) zusammengefasst werden. Der Upscale-Faktor musste individuell pro Rolle gewählt werden, da das Spektrum der Agentenzahlen zwischen den einzelnen Rollen stark variiert: Während es zu Simulationsbeginn über 14'000 Private Forest Owner gibt, gibt es nur gerade einen einzigen Pulpwood Consumer. Dies verunmöglicht die Wahl eines einheitlichen Upscaling-Faktors, der für alle Rollen gültig ist.

Das Upscaling von Agenten löst zwar das Problem von mangelndem Arbeitsspeicher und steigert die Performance. Es ist jedoch nicht auszuschliessen, dass durch das Upscaling eine Verzerrung der Agenteninteraktionen entsteht, da einzelne Agenten z.B. plötzlich eine viel hö-

[1] Für die Simulation wurde der maximale Java-Heap-Size jeweils auf 2GB erhöht. Würde dieser noch mehr erhöht werden, könnte auch die Anzahl Agenten weiter erhöht werden.

here Verhandlungsmacht haben als in der Realität. Ausserdem erschwert das Upscaling die Validierung des Simulationsprogramms, da einzelne Agenten plötzlich sehr hohe Mengen an Energieholz einkaufen wollen. Dazu müssen sie wesentlich mehr Lieferanten anfragen, als dies in Realität der Fall wäre.

Rolle	Initial No. of Agents ([16])	Max. No. of Agents ([16])	Upscale-Factor	Initial No. of Agents (upscaled)	Max. No. of Agents (upscaled)
Forester	80	80	1	80	80
Private Forest Owner	14'275	14'275	50	285	285
Forest Company	35	50	1	35	50
Bundling Organization	5	20	1	5	20
Sawmill	20	50	1	20	50
Small Private Woodfuel Cons.	10'000	120'000	50	200	2'400
Public Woodfuel Consumer	293	2'000	10	29	200
Commercial Woodfuel Cons.	397	2'000	10	40	200
Pulpwood Consumer	1	5	1	1	5
District Heating Network Op.	20	60	1	20	60
Small Private Heat Consumer	10'000	600'000	50	200	12'000
Total	25'106	738'480	-	915	15'350

Tabelle 4.1.: Upscaling der Agenten. Die ersten beiden Spalten zeigen die von Steubing et al. definierten Werte, die letzten beiden die resultierenden Werte nach Anwendung des Upscale-Faktors. Da bei der Simulation von verschiedenen Szenarien festgestellt wurde, dass sich die Anzahl Agenten im Simulationszeitraum von 25 Jahren im Extremfall verdoppeln kann, wird das theoretische Total von 15'350 Agenten kaum erreicht werden.

4.3.6. Random-Seeds

Sollen im Rahmen einer Sensitivitätsanalyse die Ergebnisse verschiedener Szenarien miteinander verglichen werden können, muss sichergestellt sein, dass die miteinander verglichenen Simulationsdaten auch auf den selben Ausgangsdaten basieren. Bei der Erstellung der Agenten und ihren Eigenschaften werden jedoch einige Eigenschaften zufällig angenommen (z.B. geographische Position des Agenten[1]). Deshalb wird an sämtlichen Stellen im Quellcode, wo ein Zufallsgenerator genutzt wird, ein Randomseed verwendet. Dieser kann als Programmargument beim Start der Simulation übergeben werden. Wird ein Zufallsgenerator mit einem Randomseed initialisiert, liefert er die Zufallszahlen bei verschiedenen Ausführungen des Programms immer in der gleichen Reihenfolge. Damit wird die Vergleichbarkeit von verschiedenen Simulationsergebnissen garantiert.

Trotzdem kann mittels Randomseed alleine noch keine absolute Vergleichbarkeit der Simulationsergebnisse garantiert werden. Dies liegt an den SPARQL-Abfragen im Quellcode. Werden

[1] Dies gilt nicht für alle Agententypen: Bei den Förstern beispielsweise sind die Koordinaten der Forstreviere bekannt und werden auch entsprechend verwendet.

bei einer SPARQL-Abfrage mehrere Triples zurückgeliefert, ist die Reihenfolge der Triples zufällig. Da es für den Verlauf der Simulation entscheidend sein kann, in welcher Reihenfolge z.B. Verkäufer ihre Anfragen beantworten, sind so theoretisch bei gleicher Ausgangslage (gleiche Parameter und gleicher Randomseed) verschiedene Simulationsausgänge möglich.

Abhilfe könnte hier beispielsweise damit geschaffen werden, indem bei allen SPARQL-Abfragen die Ergebnisse, bevor sie weiterverarbeitet werden, noch sortiert werden (z.B. nach der URI[1] des Subjekts). Sortieralgorithmen sind im Allgemeinen aber sehr aufwändig, deshalb würde sich die Performance dadurch möglicherweise stark verschlechtern.

4.3.7. Evaluator

Der Evaluator ist eine Komponente im Simulationsprogramm, welche jeweils am Ende eines simulierten Monats einige Auswertungen vornimmt. Diese Auswertungen beinhalten beispielsweise das Auslesen der gehandelten Energieholzmengen oder der aktuellen Preise eines Gutes. Die Ergebnisse dieser Auswertungen werden in eine CSV-Datei geschrieben. Die Diagramme im GUI, welche in Kapitel 4.4.1 beschrieben werden, beziehen ihre Daten vom Evaluator, d.h. am Ende jeder Simulation sind die Rohdaten für die Diagramme auch in einer CSV-Datei vorhanden, die dann auch separat beispielsweise mit Excel ausgewertet werden können. In der aktuellen Version des Simulationsprogramms wertet der Evaluator pro simuliertem Monat 108 verschiedene Parameter aus.

4.4. GUI

Das GUI (*Graphical User Interface*) besteht in erster Linie aus verschiedenen Diagrammen, welche dem Benutzer während des Simulationsvorgangs möglichst übersichtlich die wichtigsten Informationen liefern sollen. Insbesondere für die Fehlersuche und die Validierung sind gute Diagramme unerlässlich. Alle verwendeten Diagramme werden im nachfolgenden Unterkapitel kurz vorgestellt.

Aus technischer Sicht bleibt zu erwähnen, dass für die Implementierung des GUI die Swing-Bibliothek verwendet wurde. Für die Implementierung von GUIs in Java stehen drei Bibliotheken zur Verfügung: Swing, AWT und SWT. Swing und AWT sind plattformunabhängig, was aufgrund der späteren Simulation auf einem Linux-Cluster ein wichtiges Kriterium war. Da Swing auf AWT aufbaut und AWT auch um einige Konzepte erweitert, fiel schliesslich die Entscheidung auf die Verwendung von Swing.

Das GUI basiert komplett auf dem Observer-Pattern[2], d.h. das GUI registriert sich bei Beginn der Simulation bei denjenigen Programmkomponenten, von denen es Informationen erhalten will. Sobald sich diese Programmkomponenten wesentlich geändert haben (in der Simulation ist dies am Ende eines simulierten Monats), informieren diese Programmkomponenten das GUI. Dieses holt sich anschliessend die benötigten Informationen, beispielsweise die Interaktionsdaten der Agenten im vergangenen Monat. Das Observer-Pattern schafft eine lose Kopplung

[1] Ein *Uniform Resource Identifier* (URI) dient dazu, eine Ressource eindeutig zu identifizieren. Eine Ressource kann dabei z.B. eine Website sein, oder in Bezug auf die hier verwendete Ontologie-Datenbank ein einzelner Agent.

[2] Das Observer-Pattern ist ein Entwurfsmuster in der Softwareentwicklung, welches von der sog. *Gang of Four* entwickelt wurde. In ihrem Buch [2] beschreiben sie eine Vielzahl von Entwurfsmustern, die der Entwicklung von besser wiederverwendbarer Software dienen.

zwischen Beobachter (hier das GUI) und dem beobachteten Objekt (hier die Simulation). Dies hat den Vorteil, dass jederzeit ein neues GUI programmiert werden könnte, ohne das Eingriffe in die übrigen Teile des Simulationsprogramms erfolgen müssten.

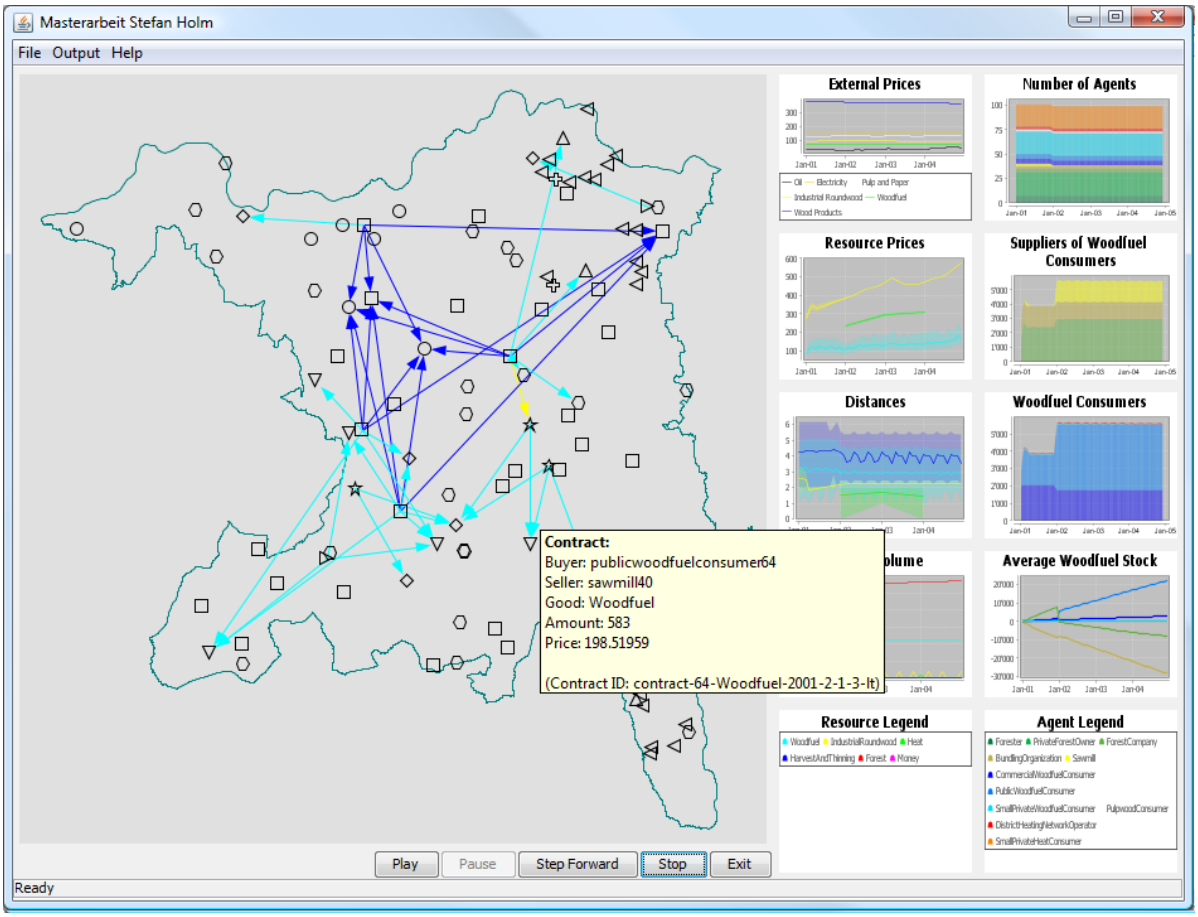


Abbildung 4.5.: Das Haupt-GUI des Simulationsprogramms. Der auf der Abbildung ersichtliche Tooltip wird angezeigt, wenn mit dem Mauszeiger auf einen Pfeil gefahren wird. Im Tooltip sind dann zusätzliche Informationen über die entsprechende Interaktion zwischen den beiden Agenten ersichtlich.

4.4.1. Visualisierung

Insgesamt gibt es zehn verschiedene Diagramme im Simulationsprogramm. Das Hauptdiagramm im linken Teil des GUI (siehe Abbildung 4.5) zeigt die geographische Verteilung der Agenten innerhalb der Kantonsgrenzen (Kanton Aargau[1]). Jedem Agententyp (Rolle) ist dabei ein unterschiedliches Symbol zugeordnet[2]. Ein Pfeil von einem Agenten zu einem anderen zeigt einen

[1] Quelle der GIS-Daten für die Kantonsgrenzen: Geoportal Kt. Aargau, www.ag.ch/agis

[2] Effektiv unterstützt das für diese Darstellung verwendete Framework Prefuse nur 10 unterschiedliche Symbole. Da die Simulation jedoch 11 verschiedene Typen von Agenten kennt, werden jeweils zwei Agententypen mit den selben Symbolen dargestellt.

Verkauf eines Gutes in die vom Pfeil angezeigte Richtung. Die unterschiedlichen Farben stehen für unterschiedliche Güter. Welche Farbe für welches Gut steht, ist in der *Resource Legend* in der unteren rechten Ecke des GUI zu sehen. Detaillierte Informationen über die Agenten sind ersichtlich, wenn mit dem Mauszeiger über das Symbol eines Agenten gefahren wird. Es wird dann ein Tooltip angezeigt. Dies ist ebenfalls bei den Pfeilen möglich: Fährt man mit dem Mauszeiger über einen Pfeil, erscheinen detaillierte Informationen über den Vertrag zwischen diesen beiden Agenten, der dem Verkauf zu Grunde liegt (siehe Abbildung 4.5).

Neben dem Diagramm mit der geographischen Verteilung der Agenten wurden neun weitere Diagramme implementiert, von welchen aus Platzgründen jedoch nur acht angezeigt werden (das neunte kann im Quellcode zugeschaltet werden, jedoch muss dafür auch ein anderes Diagramm entfernt werden):

- **External Prices**. Dieses Diagramm zeigt die Entwicklung von externen Preisen. Mit externen Preisen sind Preise gemeint, die nicht innerhalb des Simulationsprogramms berechnet werden, sondern als externe Modellvariablen vorgegeben sind. Sie dienen einerseits als Vergleich zu den berechneten (internen) Preisen, andererseits haben sie teilweise auch einen direkten Einfluss auf die Simulation. So erfolgt beim Woodfuel-Preis beispielsweise in Langzeitverträgen eine prozentuale Anpassung an den aktuellen Ölpreis. Das *External Prices*-Diagramm ist das einzige, welches eine eigene Legende besitzt.

- **Resource Prices**. Dieses Diagramm zeigt die Entwicklung der internen, vom Simulationsprogramm berechneten Preise. Die Zuordnung der Ressourcen zu den verwendeten Farben ist in der *Resource Legend* ersichtlich.

- **Distances**. Dieses Diagramm zeigt den durchschnittlichen, den minimalen und den maximalen Preis eines gehandelten Gutes in Abhängigkeit der Zeit. Der Durchschnitt ist jeweils die dunkle Linie, die Minima / Maxima die Unter bzw. Obergrenze des schwächer gefärbten Bereichs. Zur besseren Übersichtlichkeit können Minima und Maxima ausgeblendet werden (mittels Option im Quellcode). Die Legende zum Diagramm ist ebenfalls die *Resource Legend*.

- **Market Volume**. Dieses Diagramm zeigt die Transaktionsvolumina pro Ressource in Abhängigkeit der Zeit. Zusätzlich wird der Gesamtwaldbestand (rote Linie) angezeigt. Dies dient als Kontrollmechanismus, da der Waldbestand z.B. nicht plötzlich abnehmen sollte (ausser z.B. bei einem Unwetter). Auch dieses Diagramm verwendet als Legende die *Resource Legend*. Erwähnenswert bei diesem Diagramm ist, dass wenn eine Ressource in einer einzelnen Runde (i.d.R. ein Monat) von einem Produzenten via Zwischenhändler zum Endkonsumenten gelangt, die Menge doppelt dargestellt wird, da sie ja auch zweimal gehandelt wird. Da dies je nach Verwendungszweck des Diagramms zu einer Verzerrung führen kann, wurde durch die Diagramme *Suppliers of Woodfuel Consumers* und *Woofuel Consumers* Abhilfe geschaffen.

- **Number of Agents**. Dieses Diagramm zeigt die Entwicklung der Agentenzahlen als Stapeldiagramm. Dies hat den Vorteil, dass einerseits die Gesamtzahl Agenten anhand der oberen Begrenzungslinie sofort ersichtlich ist. Andererseits ist auch die Gesamtzahl an Agenten je Agententyp, durch die Höhe der einzelnen Farbbalken, ersichtlich. Die genaue

Anzahl pro Agententyp kann auch aus einem Tooltip ausgelesen werden; dazu muss mit dem Mauszeiger über das Diagramm gefahren werden. Die Legende zum Diagramm ist die *Agent Legend*.

- **Woodfuel Consumers**. Dieses Diagramm zeigt die Gesamtmenge des von Endverbrauchern gekauften Energieholzes, geordnet nach Agententyp. Das Stapeldiagramm ermöglicht das Ablesen der Gesamtmenge über alle Typen sowie die Gesamtmengen pro Typ. Im Gegensatz zum *Market Volume*-Diagramm werden hier die Käufe von Zwischenhändlern nicht angezeigt.

- **Suppliers of Woodfuel Consumers**. Dieses Diagramm stellt das Gegenstück zum *Woodfuel Consumers*-Diagramm dar. Es zeigt, von welchen Agenten die Endkonsumenten ihr Energieholz beziehen. Die jeweils dargestellte Gesamtsumme der Verkäufe entspricht zu jedem Zeitpunkt der Gesamtsumme der Einkäufe im *Woodfuel Consumers*-Diagramm.

- **Average Woodfuel Stock**. Dieses Diagramm zeigt den durchschnittlichen Bestand an Energieholz pro Agententyp. Die dargestellte Information dient in erster Linie als Kontrolle; die Linien sollten hier jeweils möglichst parallel verlaufen, da ansonsten von Agenten entweder mehr eingekauft als verbraucht, oder mehr verbraucht als eingekauft wird (was in der Realität natürlich nicht möglich ist, in der Simulation aufgrund des Einkaufverhaltens der Intermediäre jedoch schon).

- **Financial Performance**. Dieses Diagramm zeigt den durchschnittlichen Geldbestand pro Agententyp und dient wie das *Average Woofuel Stock*-Diagramm in erster Linie als Kontrollmechanismus. Idealerweise sollten sich die Linien nicht zu stark nach unten oder oben bewegen (die Geldzuflüsse sollten in etwa den Geldabflüssen entsprechen). Kurzfristige Schwankungen sind jedoch möglich, beispielsweise wenn die Heizkosten plötzlich stark ansteigen.

- **Resource Legend**. Diese Legende stellt eine gemeinsame Legende für die Diagramme dar, welche Informationen über Ressourcen anzeigen. Die gemeinsame Legende für verschiedene Diagramme bietet den Vorteil, dass die Diagramme selbst mehr Fläche zur Verfügung haben für die Informationen, welche sie darstellen sollen.

- **Agent Legend**. Die *Agent Legend* zeigt die Zuordnung von Farben zu Agententypen für die Diagramme oberhalb von ihr, und stellt somit wie die *Resource Legend* eine von verschiedenen Diagrammen gemeinsam verwendete Legende dar.

In allen Diagrammen ist es möglich, über deren Kontextmenü (Aufrufbar über einen Rechtsklick) hinein- oder heraus zu zoomen. Auch können die Diagramme via Kontextmenü in eine Bilddatei gespeichert werden. Mit Aktivierung einer entsprechenden Optionen beim Programmstart können auch jeweils am Ende eines simulierten Jahres automatisch alle Diagramme als PNG-Datei gespeichert werden (auch diejenigen, welche nicht im GUI angezeigt werden). Dies dient einer vereinfachten Analyse im Anschluss an den Simulationsvorgang, wie zum Beispiel bei einer Sensitivitätsanalyse (siehe Kapitel 5.4).

4.4.2. Konsolen-Modus

Um die Verwendung des Simulationsprogramms auf dem Cluster zu vereinfachen, wurde neben dem GUI-Modus auch die Option erstellt, das Simulationsprogramm per Konsole zu starten (siehe Abbildung 4.6). Dies wird mit dem Programmargument *-console* ermöglicht, während das GUI entweder mit der Option *-gui* oder ohne Programmargumente gestartet wird (z.B. mit einem Doppelklick auf das Programm-Icon).

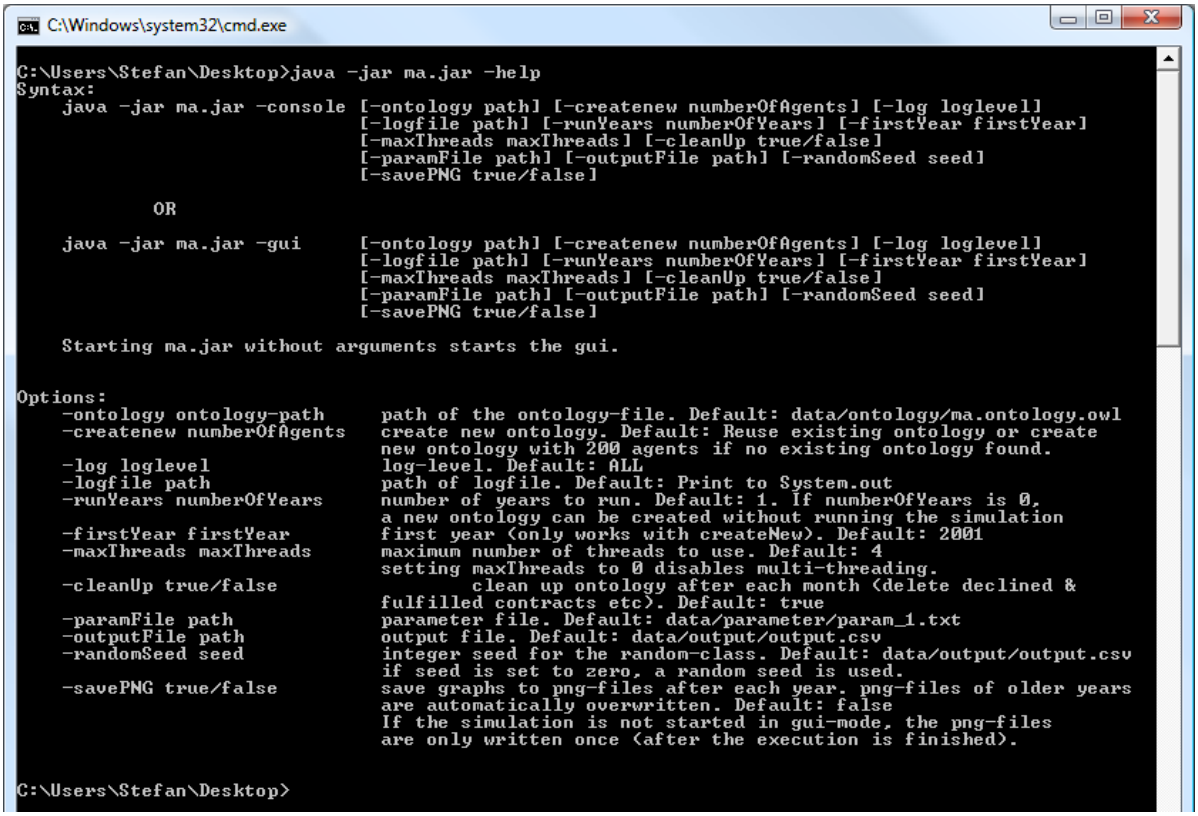


Abbildung 4.6.: Start des Simulationsprogramms in der Konsole. Eine vollständige Übersicht über die möglichen Programmargumente wird in Anhang D gezeigt.

4.5. Verwendete Frameworks

Für die Implementierung wurden verschiedene Frameworks verwendet. Diese werden in den nachfolgenden Unterkapiteln kurz erklärt.

4.5.1. Jena

Jena ist das zentrale Framework des Simulationsprogramms. Jena ist ein Java-Framework, welches ursprünglich für die Programmierung von *Semantic Web*[1] Applikationen erstellt wurde. Im Simulationsprogramm wurde dieses Framework eingesetzt, da es eine Schnittstelle zur Speicherung von OWL-Daten bietet, und ausserdem die Möglichkeit von SPARQL[2]-Abfragen bietet.

4.5.2. Prefuse

Prefuse ist ein Framework für die Darstellung von interaktiven Graphen in Java-Applikationen. Im Simulationsprogramm wird Prefuse für die Darstellung der geographischen Position der Agenten sowie die Interaktionen zwischen ihnen verwendet.

4.5.3. JFreeChart

JFreeChart ist eine Bibliothek für Java-Applikationen, welche es ermöglicht, mit wenig Programmieraufwand verschiedenste Arten von Diagrammen darzustellen. Im Simulationsprogramm wird JFreeChart für die Darstellung der auszuwertenden Variablen verwendet. Auch die ausgegebenen Bild-Dateien werden mit Hilfe von JFreeChart erstellt.

4.5.4. JUnit

JUnit ist ein Framework für automatisiertes Testing von Java-Applikationen. Um das Testing zu automatisieren, werden möglichst viele Testklassen geschrieben, die Methoden enthalten, welche die Funktionalität des Hauptprogramms testen. Die Gesamtheit dieser Testklassen wird als Testsuite bezeichnet. Wird am Quellcode des Hauptprogramms eine Änderung vorgenommen, kann danach die Testsuite ausgeführt werden und es ist sofort ersichtlich, ob immer noch alle Tests erfolgreich ablaufen. Ist dies nicht der Fall, muss mit der vorangegangenen Änderung ein Fehler im Code entstanden sein.

In der Simulation wird JUnit verwendet, um die Korrektheit einzelner Methoden sicherzustellen. So wurden beispielsweise Tests für die Umrechnungsfunktionen von verschiedenen Einheiten (z.B. ha Wald -> m^3 Holz) erstellt.

4.5.5. Javadoc

Javadoc ist ein Werkzeug für die Erstellung der Softwaredokumentation. Es wurde wie Java ursprünglich von Sun Microsystems entwickelt, und ist in der Zwischenzeit vollständig in den Java Development Kit (JDK) integriert. Mithilfe von Javadoc lässt sich z.B. aus Klassen- und Methodenkommentaren direkt eine HTML-Dokumentation erzeugen. Bei der Implementierung des Simulationsprogramms wurde konsequent darauf geachtet, neben den normalen Code-Kommentaren (innerhalb der Methoden) auch möglichst viele Schnittstellen (mindestens alle Klassen, möglichst viele öffentliche Methoden) mit Javadoc-Kommentaren zu versehen, um die Wiederverwendbarkeit des Codes zu vereinfachen.

[1] Das *Semantic Web* hat den Zweck, Informationen im Internet für Computer verständlich zu machen, indem Texte mit Metadaten versehen werden.

[2] *SPARQL* ist eine RDF-Abfragesprache. Die Syntax ist ähnlich wie diejenige von SQL.

5. Simulation

5.1. Annahmen

Das implementierte Simulationsprogramm basiert auf einem konzeptuellen Modell des Schweizer Energieholzmarktes. Ein Modell ist ein Abbild der Realität, welches jedoch gewisse Eigenschaften der Realität weglässt, so dass nur die für den Verwendungszweck wesentlichen Merkmale abgebildet werden. Es ist jedoch oft schwierig zu beurteilen, welche Merkmale der Realität für das Modell wesentlich sind und welche nicht. Während der Programmierung des Simulationsprogramms wurden deshalb viele Annahmen getroffen (siehe vollständiger Annahmenkatalog in Anhang A). Sämtliche Annahmen im Annahmenkatalog wurden mit einem Kürzel versehen. Das selbe Kürzel wurde im Java-Code als Kommentar an derjenigen Stelle eingefügt, an der die Annahme in Programmcode umgesetzt wurde. Wird nun eine bestimmte Annahme angezweifelt oder soll sie angepasst werden, kann mittels des Kürzels die entsprechende Stelle im Code gesucht und angepasst werden.

5.1.1. AHP

In den nachfolgenden Tabellen sind die für die Simulation verwendeten Gewichtungsvektoren für die Subtypen der einzelnen Agententypen dargestellt. Subtypen gibt es nur für die Rollen Forester, Private Forest Owner, Small Private Woodfuel Consumer und Small Private Heat Consumer. Für alle anderen Rolle gibt es lediglich einen Standardtyp. Dieser ist in den nachfolgenden Tabellen jeweils in der ersten Wertespalte zum Vergleich aufgelistet. In der Standardeinstellung des Simulationsprogramms sind die einzelnen Subtypen jeweils zu gleichen Teilen vorhanden. Die Wahrscheinlichkeit der Zugehörigkeit zu einem bestimmten Subtyp kann jedoch mittels Parameter-Einstellung (siehe Kapitel 4.3.4) variiert werden. Die gezeigten Gewichtungsvektoren werden normalerweise mittels AHP (siehe Kapitel 2.4) berechnet. Auf dies wurde jedoch verzichtet, da die Werte für die einzelnen Subtypen nur grobe Schätzungen sind, und sie zudem noch mittels Sensitivitätsanalyse variiert werden. Das Simulationsprogramm bietet innerhalb des Codes jedoch sowohl die Möglichkeit der Verwendung vorberechneter Gewichtungsvektoren, wie auch der Verwendung von paarweisen Vergleichsmatrizen (wie in AHP üblich). Möchte man lieber von paarweisen Vergleichsmatrizen ausgehen - der exakteren Variante - müsste man idealerweise erst die Werte z.B. durch Umfragen bei Förstern (des entsprechenden Stereotyps) bestimmen.

	Standard	Conservative	Environmentally Oriented Proactive	Profit Oriented	Educated Professional
Profit	*0.60*	0.20	0.45	0.45	0.60
Umwelt	*0.20*	0.30	0.30	0.20	0.25
Freundschaft	*0.20*	0.50	0.25	0.35	0.15

Tabelle 5.1.: Gewichtungsvektoren für die Subtypen der Rolle *Forester*

	Standard	Farmer Type	Recreational Type	Profit Type	Bankster Type
Profit	*0.60*	0.25	0.15	0.70	0.60
Umwelt	*0.20*	0.60	0.30	0.10	0.15
Freundschaft	*0.20*	0.15	0.55	0.20	0.25

Tabelle 5.2.: Gewichtungsvektoren für die Subtypen der Rolle *Private Forest Owner*

	Standard	Greedy Type	Environmental Type
Profit	*0.60*	0.70	0.20
Umwelt	*0.20*	0.20	0.70
Freundschaft	*0.20*	0.10	0.10

Tabelle 5.3.: Gewichtungsvektoren für die Subtypen der Rolle *Small Private Woodfuel Consumer* und *Small Private Heat Consumer*

Normalisierung

Die Gewichtungsvektoren sind nur der eine Bestandteil im AHP. Ebenso wichtig sind die Matrizen für den Vergleich von verschiedenen Alternativen: die Normalisierung der Ausgangsdaten für den Vergleich von verschiedenen Alternativen hat einen sehr grossen Einfluss auf die schliesslich getroffene Entscheidung. Das Problem lässt sich am besten anhand eines Beispiels erläutern: Ein kaufwilliger Agent hat drei Offerten zur Auswahl. Für diese drei Alternativen will er bezüglich dem Kriterium *Preis* eine Entscheidungsmatrix erstellen. In allen drei Offerten wird dieselbe Menge des gewünschten Gutes angeboten. Die erste Offerte hat einen Preis von CHF 100, die zweite CHF 105, die dritte CHF 110. Die Preise müssen nun paarweise verglichen werden, damit die Entscheidungsmatrix erstellt werden kann. Welches Gewicht soll man nun dem günstigsten Preis im direkten Vergleich mit dem teuersten Angebot geben? Eine Variante wäre es, den maximalen Wert neun zu geben (vgl. Abb. 2.3), da es von allen drei die günstigste Offerte ist. Man könnte aber auch z.B. den Wert 3 geben, der bedeutet, dass CHF 100 nur ein bisschen besser ist als CHF 110, schliesslich beträgt die Differenz zwischen den beiden Offerten ja nur CHF 10. Wenn ein hoher Wert gewählt wird, hat dies zur Folge, dass weitere Kriterien in der Berechnung schliesslich einen wesentlich geringeren Einfluss auf die Schlusswertung haben. Wird hingegen ein niedriger Wert gewählt, kann es beispielsweise passieren, dass sich die Alternativen bei der in der Schlusswertung dann kaum mehr unterscheiden, obwohl das bewertete Kriterium ein besonders wichtiges war.

Saaty beschreibt dies in [12] mit den Begriffen *scarcity* (Knappheit) und *abundance* (Überfluss) und am Beispiel von Früchten. Gibt es viele rote Früchte (Überfluss), spielt die Farbe bei der Auswahl einer Frucht kaum mehr eine Rolle. Gibt es hingegen nur sehr wenige rote Früchte (Knappheit), während andere Früchte sonstige Farben habe, macht dies die Farbe Rot wertvoll.

Im Simulationsprogramm mussten dazu für jedes Kriterium ein geeigneter Algorithmus gefunden werden, damit verschiedene Alternativen richtig miteinander verglichen werden konnte. Beim Kriterium *Preis* beispielsweise funktioniert der gewählte Algorithmus wie folgt: Will ein

kaufwilliger Agent verschiedene Offerten in Bezug auf den Preis miteinander vergleichen, berechnet er zuerst den Durchschnittspreis seiner Offerten. Anschliessend trifft er die Annahme, dass der bestmögliche Preis 15% unter dem Durchschnittspreis liegt, der schlechtmöglichste 15% über dem Durchschnittspreis. Somit erhält er eine Preisspanne. Nun ordnet er den Preis jeder einzelnen Offerte in diese Preisspanne ein, indem er das Prozentil bestimmt, in welchem der Preis in der Preisspanne liegt. Somit erhält er ein Wert zwischen 0 und 1, wobei 0 für ein besonders gutes Angebot steht, 1 für ein besonders schlechtes[1]. Dieser Wert wird für sämtliche Offerten berechnet[2]. Für den anschliessenden paarweisen Vergleich der Offerten wurde eine Helper-Methode geschrieben, welche aus den beiden Werten zwischen 0 und 1 einen AHP-tauglichen Wert zwischen 1/9 und 9 berechnet. Dazu wird angenommen, dass wenn der eine Wert 0 und der andere 1 ist, der AHP-Wert 9 (bzw. 1/9) der passendste Wert ist. Dazwischen erfolgt eine lineare Abstufung.

5.1.2. Weitere Kriterien des Entscheidungsverhaltens

Im vorangegangenen Unterkapitel wurden gezeigt, wie die einzelnen Agenten die Kriterien Profit, Umwelt und Freundschaft gewichten. Die verschiedenen Subtypen unterscheiden sich jedoch noch in einer Reihe weiterer Eigenschaften. Diese sind in Tabelle 5.4 festgehalten. Die Eigenschaften basieren unter anderem auf den Annahmen in Anhang A.

5.2. Verifikation und Validierung

Verifikation und Validierung bedeuten sowohl im Zusammenhang mit Software Engineering wie auch in der agentenbasierten Modellierung: *Das Richtige* (Validierung) *richtig* (Verifikation) tun. Die beiden Begriffe wurden in Kapitel 2.1 detaillierter erklärt.

In der vorliegenden Arbeit wurde die Validierung hauptsächlich in Form von Gesprächen mit den Betreuern durchgeführt. Für die Verifikation wurden einerseits diverse Unit-Tests (JUnit-Tests, siehe Kapitel 4.5.4) durchgeführt, andererseits wurde eine spezielle Klasse entwickelt, welche die Ontologie-Datenbank auf ihre Konsistenz überprüft. Dieses sog. *ConsistencyChecker*-Klasse wurde ähnlich wie eine JUnit-Klasse aufgebaut und beinhaltet verschiedene Methoden, die jeweils eine spezifische Eigenschaft der Daten in der Datenbank überprüft: So darf beispielsweise kein Small Private Heat Consumer existieren, der nicht mit einem District Heating Network Operator verbunden ist (da dies im Realsystem einem Haus entsprechen würden, welches eine Fernwärmeheizung besitzt, welche jedoch nicht an ein Fernwärmenetz angeschlossen ist). Für die Validierung dienen insbesondere auch die vom Simulationsprogramm dargestellten Diagramme (siehe Kapitel 4.4.1). So sollten beispielsweise die Kurven im *Financial-Performance*-Diagramm, welches die finanzielle Entwicklung eines Agenten zeigt, unter normalen Bedingungen nicht über längere Zeit ansteigen oder ins Negative fallen (auch wenn dies in

[1] Theoretisch kann es passieren, dass der Preis einer einzelnen Offerte ausserhalb der berechneten Preisspanne liegt, da ja angenommen wurde, dass sich die gesamte Preisspanne im Bereich von +/- 15% um den Durchschnittspreis liegt. In diesem Fall wird der Preis einfach auf den nächstliegenden Wert innerhalb der Preisspanne gerundet.

[2] Da angenommen wird, dass jeder Einkauf gewisse Transaktionskosten verursacht, fliesst zusätzlich auch die offerierte Menge in die Berechnung des Profit-Kriteriums ein, damit es für einen Käufer attraktiver ist, wenn er ein mal 60 Einheiten kaufen kann, anstatt drei mal 20 Einheiten kaufen zu müssen. Auf den dafür verwendeten Algorithmus wird hier jedoch nicht eingegangen.

Rolle (Subtyp)	Bevorzugung Langzeitverträge	Anteil Produktion Energieholz
Forester		
Conservative	75%	10%
Environmentally Oriented Proactive	40%	30%
Profit Oriented	60%	10%
Educated Professional	40%	20%
Private Forest Owner		
Farmer Type	40%	35%
Recreational Type	75%	30%
Profit Type	60%	20%
Bankster Type	60%	10%
Forest Company	50%	20%
Bundling Organization	50%	-
Sawmill	50%	-
Small Private Woodfuel Consumer		
Greedy Type	50%	-
Environmental Type	50%	-
Public Woodfuel Consumer	80%	-
Commercial Woodfuel Consumer	70%	-
Pulpwood Consumer	40%	-
District Heating Network Operator	90%	-

Tabelle 5.4.: Diese Tabelle zeigt einerseits, wie stark die einzelnen Agententypen das Abschliessen von Langzeitverträgen gegenüber von einzelnen Käufen bevorzugen. Steht ein Agent vor einer Kaufentscheidung, entscheidet er sich anhand dieser Wahrscheinlichkeit entweder für den Abschluss eines Langzeitvertrages oder für einen Einzelkauf. Diese Wahrscheinlichkeiten wurden für alle im Woodfuel Market agierenden Teilnehmer festgelegt. In der aktuellen Version des Simulationsprogramms findet die Entscheidung für oder gegen den Abschluss eines Langzeitvertrages jedoch ausschliesslich auf der Käuferseite statt; die Verkäufer behandeln Anfragen in einer zufälligen Reihenfolge und senden für jede Anfrage eine Offerte zurück, solange sie die Offerten mit ihrem aktuellen Lagerbestand decken können. Die rechte Spalte zeigt für die Waldbesitzer, welchen Anteil der Holzmasse eines kompletten Baumes sie für die Produktion von Energieholz verwenden. Für die Forest Company ist ebenfalls ein Eintrag vorhanden; dies kommt daher, dass eine Forest Company, wenn sie für einen Förster die Holzernte übernimmt, das Holz unter Umständen danach selbst verkauft und somit auch entscheiden kann, welchen Anteil des Baumes sie zu Energieholz verwertet. In der Realität wird der Anteil Energieholz an der gesamten Holzmasse eher durch die Baumart bestimmt [7]. Da jedoch in der Simulation kein Unterschied zwischen verschiedenen Holzarten gemacht wird, wurde versucht, auf diese Weise diese Variable in das Simulationsprogramm einzubringen.

Einzelfällen möglich sein und auch der Realität entsprechen kann). Ebenso dürfen die Intermediäre nicht über längere Zeit einen negativen Warenbestand haben, da dies in der Realität nicht möglich ist. Abbildung 5.1 zeigt anhand eines Beispiels, wie solche Diagramme zur Validierung des Simulationsprogramms genutzt werden können.

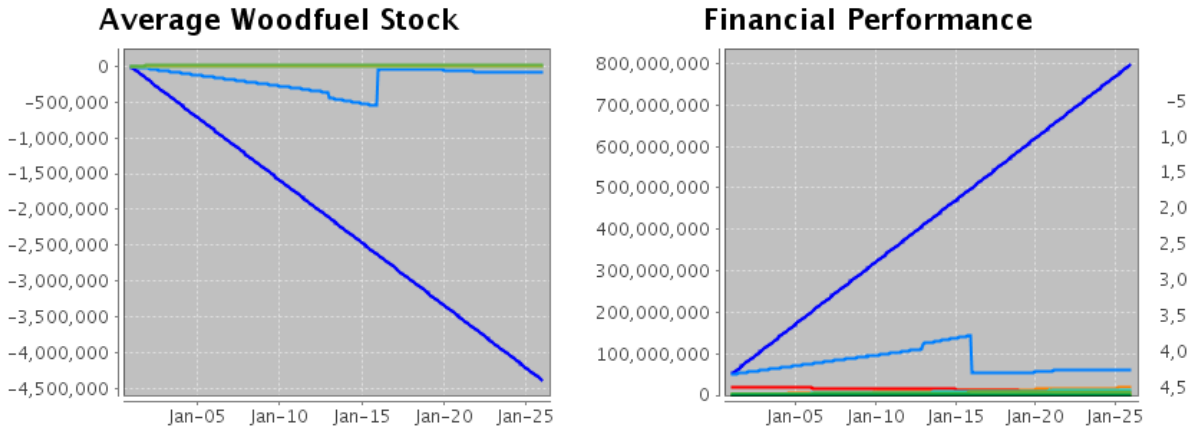


Abbildung 5.1.: Beispiel von zwei Diagrammen, welche im Zuge der Validierung analysiert und interpretiert wurden. Die jeweils dunkelblauen Linien stellen Commercial Woodfuel Consumer dar, während die hellblauen Linien Public Woodfuel Consumer darstellen. Auffallend ist, dass die Linien des linken Diagramms praktisch dem Spiegelbild des rechten Diagramms entsprechen. Ausserdem zeigen beide Diagramme ein Szenario, welches in der Realität kaum in dieser Form auftreten kann: über Jahre hinweg stark sinkende negative(!) Lagerbestände (links) und über Jahre linear steigendes Vermögen (rechts). Offensichtlich legen hier Agenten Geld zur Seite, welches sie für Energieholz benötigen würden. Aus irgendwelchen Gründen kaufen sie das benötigte Energieholz mit dem vorhandenen Geld jedoch nicht. Zeigen Diagramme solche Kurven, sind dies deutliche Anzeichen für Fehler in der Computersimulation.

5.3. Simulation auf dem Cluster

Damit die für die Sensitivitätsanalyse (siehe Kapitel 5.4) festgelegten 28 verschiedenen Szenarien in möglichst kurzer Zeit simuliert werden können, war eine parallele Simulation auf einem High-Performance-Cluster notwendig. Die Simulation wurde deshalb auf dem WSL-Cluster durchgeführt. Die Szenarien wurden über einen Simulationszeitraum von 25 Jahren simuliert, beginnend beim Jahr 2001. Das Jahr 2001 wurde deshalb gewählt, da für die Jahre 2001 - 2011 umfangreiche Datensätze vom Bundesamt für Statistik (BFS) über Holzpreise etc. vorhanden waren. Der Cluster wurde auch für die in Kapitel 5.5 erwähnten Szenarien sowie für Simulationen während dem Prozess der Verifikation und Validierung verwendet.

Die Simulationszeit (d.h. die für die Ausführung eines Simulationsdurchgangs benötigte Zeit) ist - neben Simulationszeitraum und Anzahl Agenten - insbesondere auch von den Anzahl Iterationen pro Marktphase (siehe Kapitel 3.3.3) abhängig. Standardmässig wurde mit vier

Iterationen pro Phase simuliert (falls schon vorher Ende der vier Iterationen alle Agenten ihren Bedarf decken können, werden die restlichen Iterationen übersprungen). Damit dauerte ein einzelner Simulationsdurchgang bei einer Anzahl von 1000 Agenten zwischen 2.5 und 3 Stunden. Erhöhte man den Standardwert für die Anzahl Iterationen auf acht, dauerte die Simulation über 10 Stunden.

5.4. Sensitivitätsanalyse

Abbildung 5.2 zeigt eine graphische Übersicht, welche für die qualitative Sensitivitätsanalyse erstellt wurde. In den Spalten sind jeweils unterschiedliche Szenarien, horizontal jeweils ein spezifisches Diagramm abgebildet. Das Ziel einer solchen Abbildung ist das Erkennen von Mustern, bzw. von Unterschieden zwischen einzelnen Diagrammen. Da die Erkennung von Mustern für Menschen wesentlich einfacher ist als für Computer, ist eine solche Abbildung eine gute Grundlage für eine qualitative Sensitivitätsanalyse. Wird ein Simulationsdurchgang durchgeführt, können mit dem Setzen des entsprechenden Programmarguments automatisch die verschiedenen Diagramme erzeugt und als Bilddatei abgespeichert werden. Pro Simulationsdurchgang werden elf Diagramme erzeugt. Diese befinden sich am Ende der Simulation alle in separaten Bilddateien. Insgesamt wurden für die Sensitivitätsanalyse 28 verschiedene Szenarien definiert. Somit entstehen bei der Simulation der 28 Szenarien schliesslich 28 x 11 Diagramme, die zu einem einzelnen Bild zusammengefügt werden müssen. Um diese Arbeit zu automatisieren, wurde ein kleines Batch-Skript erstellt, welches diese Aufgabe übernimmt. Somit muss nach der Simulation der 28 Szenarien nur noch einmalig das Batch-Skript ausgeführt werden, um die gezeigte Abbildung zu erhalten.

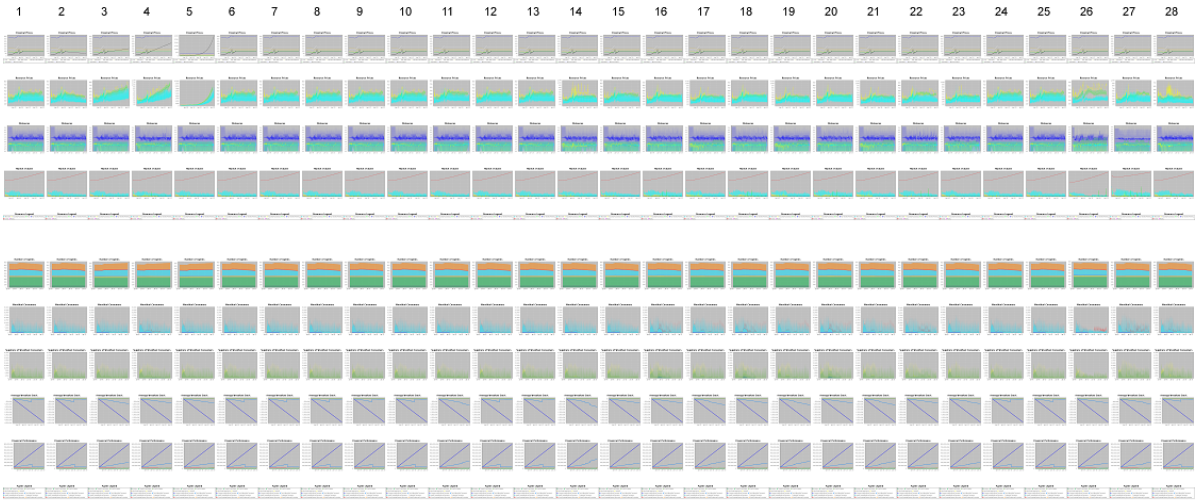


Abbildung 5.2.: Aus der Sensitivitätsanalyse resultierende Übersicht über alle Diagramme. Die verschiedenen Spalten zeigen jeweils die Ergebnisse aus Simulationen mit unterschiedlichen Input-Parametern. Da das Simulationsprogramm noch nicht vollständig validiert ist, ist dieses Beispiel der Sensitivitätsanalyse eher als Proof of Concept zu verstehen.

5.5. Szenarien

In diesem Abschnitt soll anhand eines einfachen Beispiels gezeigt werden, wie mit Hilfe des Simulationsprogramms eine bestimmte Forschungsfrage beantwortet werden kann. Die Forschungsfrage, anhand der das Beispiel demonstriert wird, lautet: *Wirkt sich ein starkes Umweltbewusstsein der Mehrheit der Marktteilnehmer auf die Entwicklung des Energieholzpreises aus?* Dazu werden zwei verschiedene Szenarien geschaffen. Im ersten Szenario achten sich die Marktteilnehmer hauptsächlich auf die Umwelt, während im zweiten Szenario für die Marktteilnehmer hauptsächlich der Profit (bzw. für Konsumenten geringe Kosten) im Vordergrund stehen. Dazu wurden zwei Parameter-Files erstellt, welche je eines der beiden Szenarien abdecken. Tabelle 5.5 zeigt eine Gegenüberstellung der für die beiden Szenarien gewählten Parameter. Diese Parameter-Dateien werden anschliessend dem Simulationsprogramm als Input übergeben. Somit müssen zwei Simulationsdurchgänge durchgeführt werden. Anschliessend können die vom Simulationsprogramm ausgegebenen Diagramme analysiert werden. Abbildung 5.3 zeigt eine kurze Analyse des Ausgangs der beiden simulierten Szenarien.

Rolle (Subtyp)	Wert im Szenario „Umweltorientierung“	Wert im Szenario „Profitorientierung“
Forester		
Typ „Conservative“	65%	5%
Typ „Environmentally Oriented“	25%	20%
Typ „Profit Oriented“	5%	20%
Typ „Educated Professional“	5%	55%
Private Forest Owner		
Typ „Farmer“	65%	5%
Typ „Recreational“	25%	5%
Typ „Profit“	5%	55%
Typ „Bankster“	5%	35%
Small Private Woodfuel Consumer		
Typ „Greedy“	10%	90%
Typ „Environmental’	90%	10%
Small Private Heat Consumer		
Typ „Greedy“	10%	90%
Typ „Environmental’	90%	10%
Anteil Forester, die Komplettoutsourcer sind:	20%	80%

Tabelle 5.5.: Gewählte Parameter für die Simulation der beiden Szenarien „profitorientierte Agenten“ und „umweltbewusste Agenten“. Details zu den einzelnen Agententypen sind in Kapitel 5.1.1 ersichtlich.

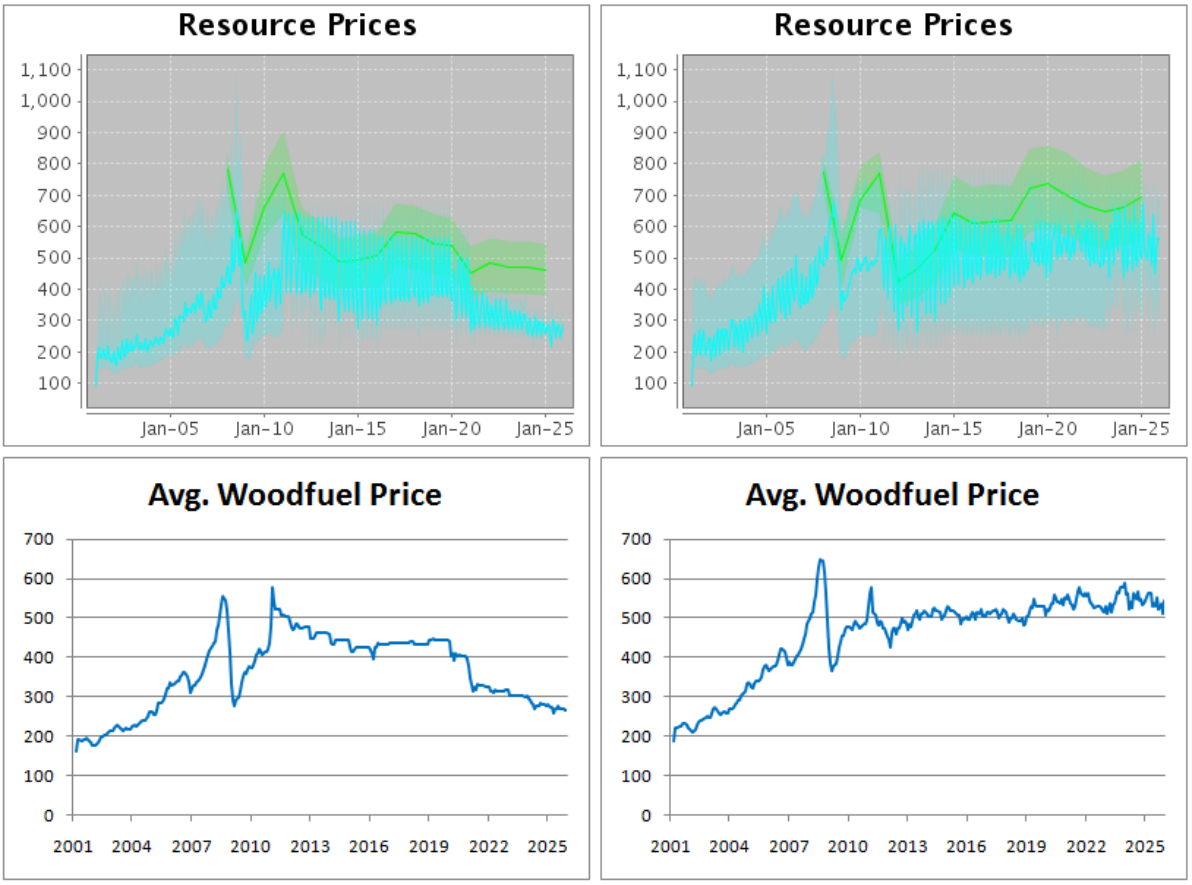


Abbildung 5.3.: Ergebnis der Simulation der beiden Szenarien. Links das Szenario „umweltbewusste Agenten“, rechts das Szenario „profitorientierte Agenten“. Die oberen beiden Diagramme wurden direkt vom Simulationsprogramm ausgegeben und zeigen sowohl den Woodfuel-Preis (blau) wie auch den Heat-Preis (Preis für Fernwärme / grün). Die unteren Diagramme, welche nur den Preisverlauf von Energieholz zeigen, wurden mit Hilfe der vom Simulationsprogramm ausgegebenen Rohdaten mit Excel erstellt. Zur besseren Erkennung des Preistrends wurde in den Excel-Diagrammen der Preisverlauf als gleitender Durchschnitt über drei Monate dargestellt.
Interessant ist vor allem der Preisverlauf ab ca. Mitte 2011. Ab diesem Zeitpunkt sind keine realen Ölpreise mehr bekannt; der Ölpreis wird ab diesem Zeitpunkt als konstant angenommen. Beim Vergleich der Diagramme fällt auf, dass bei vielen umweltbewussten Agenten der Preis längerfristig sinkt, während er bei vielen profitorientierten Agenten längerfristig steigt.
Wichtig: Dieses Beispiel dient zur Veranschaulichung der Auswertung von Simulationsergebnissen und ist somit keinesfalls als valide Aussage über den Schweizer Energieholzmarkt zu werten!

6. Diskussion

6.1. Eignung der Verwendung einer Ontologie-Datenbank

Das Besondere an dem für die Implementierung des Simulationsprogramms gewählten Ansatz ist die Verwaltung der Agentendaten in einer Ontologie-Datenbank. Die Agenten werden nicht als Java-Objekte im Arbeitsspeicher gehalten; alle Informationen über die Eigenschaften der Agenten, wie beispielsweise ihr Entscheidungsverhalten oder die Menge an Güter, welche sie besitzen, werden in die Datenbank geschrieben. Sobald die Informationen wieder benötigt werden, werden sie wieder aus der Datenbank ausgelesen. Dieses Vorgehen hat mehrere Vorteile:

- Es muss sich nicht um die Java-Speicherstrukturen gekümmert werden, d.h. es müssen keine Vektoren, Hashmaps etc. verwaltet werden, welche die Eigenschaften der Agenten beinhalten. Der Quellcode wird damit übersichtlicher.
- Mittels SPARQL-Queries können beliebig komplexe Anfragen an die Datenbank geschickt werden. So kann beispielsweise mit minimalem Aufwand (d.h. mit einer einzigen Datenbankabfrage) der Durchschnittspreis eines Gutes in einer Verhandlungsrunde ausgelesen werden.
- Mittels eines Ontologie-Editors kann der Inhalt der Ontologie-Datenbank analysiert werden. Damit können einerseits in der Entwicklungsphase Inkonsistenzen in der Datenbank festgestellt werden, andererseits sind auch Auswertungen im Anschluss an eine Simulation möglich. In der vorliegenden Arbeit wurde Protégé[1] als Ontologie-Editor verwendet (siehe Abbildung 6.1).

Ein weiterer Vorteil bei der Verwendung einer Ontologie-Datenbank ist die Möglichkeit der Verwendung eines Reasoners. Damit können mittels zuvor definierten Inferenz-Regeln aus vorhandenen Daten logische Schlüsse gezogen werden, um damit neues Wissen zu generieren (siehe Kapitel 2.3). Es hat sich jedoch im Verlauf der Arbeit herausgestellt, dass die Verwendung eines Reasoners die Performance massiv beeinträchtigt. Folglich wurde auf die Verwendung eines Reasoners verzichtet. Triples, die zuvor durch den Reasoner kreiert wurden, wurden alle explizit erstellt. So gibt es beispielsweise zwischen *Agent*-Instanzen und *Contract*-Instanzen die Beziehung *isBuyingPartyIn*, welche für einen konkreten Agenten die Verbindung zu einem Vertrag herstellt, in welchem er als Käufer auftritt, sowie die Umkehrbeziehung *hasBuyingParty*, welche die Verbindung von einer Vertragsinstanz zum Käufer in diesem Vertrag herstellt (der Aufbau der Ontologie bzw. das komplette Datenbankschema ist in Anhang B ersichtlich). Wird ein Reasoner verwendet, muss nur eine der beiden Beziehung zwischen Agent und Contract gesetzt werden. Der Reasoner setzt dann die entsprechende Umkehrbeziehung automatisch, sofern in der Ontologie auch definiert wurde, welche Beziehung welche Umkehrbeziehung hat.

[1] Protégé ist ein auf Java basierender Open-Source Ontologie-Editor, welcher an der Stanford University entwickelt wurde (http://protege.stanford.edu)

In Bezug auf die Performance war jedoch nicht nur die Verwendung eines Reasoners problematisch. Auch wenn auf einen Reasoner verzichtet wird, benötigen die Lese- und Schreibzugriffe auf die Datenbank noch relativ viel Zeit. Eine frühe Version des Simulationsprogramms arbeitete noch ohne Ontologie-Datenbank, d.h. alle Informationen über die Agenten wurden direkt als Java-Objekte im Hauptspeicher gehalten. Als schliesslich in einer weiteren Version alle Agenten-Informationen in die Ontologie-Datenbank ausgelagert wurden, zog dies einen Geschwindigkeitsverlust von ca. einem Faktor 10 mit sich.

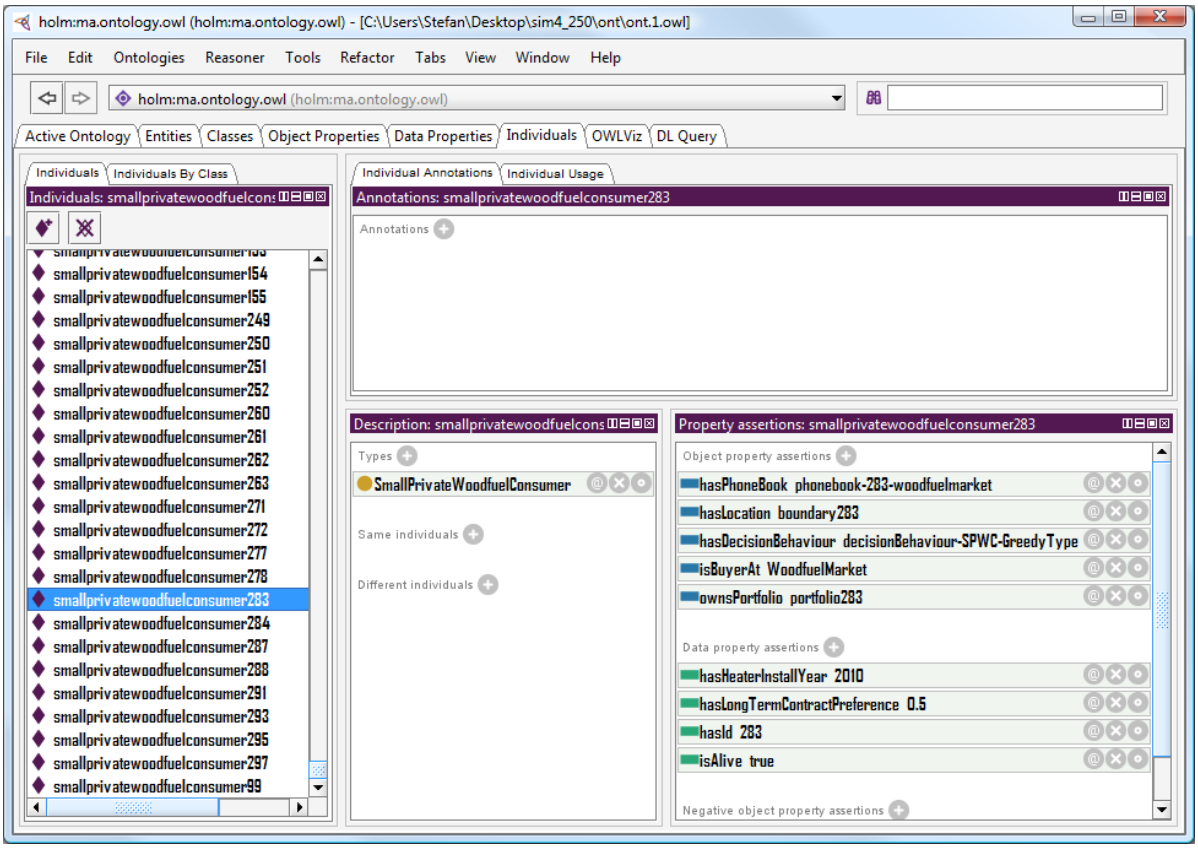


Abbildung 6.1.: Screenshot von Protégé mit geöffneter Ontologie-Datenbank. Protégé ist ein Ontologie-Editor, der sich im Zusammenhang mit der vorliegenden Arbeit hervorragend zum Debugging eignet, da allfällige Inkonsistenzen in der Datenbank schnell erkennt werden können. In der Abbildung ersichtlich ist die Instanz eines Small Private Woodfuel Consumers. Im rechten unteren Teil des Fensters werden die Eigenschaften dieses Agentens aufgelistet.

6.2. Eignung des generischen Frameworks für zukünftige Arbeiten

Eines der Ziele der vorliegenden Arbeit war es, nicht nur das bestehende konzeptuelle Modell des Schweizer Energieholzmarktes zu implementieren, sondern auch eine Basis für zukünftige Arbeiten zu schaffen (siehe dazu auch Kapitel 1.2). Ein klar vom konkreten Simulationsprogramm am

Beispiel des Schweizer Energieholzmarktes getrenntes generisches Framework (z.B. für andere Märkte mit ähnlichen Eigenschaften) konnte jedoch nicht geschaffen werden. Dies hauptsächlich aus dem Grund, dass die Aufteilung der gesamten Klassenstruktur in eine abstrakte und eine konkrete Ebene den Code für den vorgegebenen Zweck unnötig verkompliziert hätte. Da zudem kein konkreter weiterer Verwendungszweck für ein allfälliges Framework bekannt war (z.B. Implementierung des italienischen Energieholzmarktes), wäre es schwierig gewesen, die richtige Abstraktionsebene für ein generisches Framework zu finden.

Trotzdem wurde bei der Implementierung stark darauf geachtet, dass Komponenten - wenn auch unter leichten Anpassungen - für zukünftige ähnliche Simulationsprogramme wiederverwendet werden können. Die GUI-Komponenten beispielsweise wurden so implementiert, dass sie ohne Änderungen am Code zusätzliche Agententypen und/oder Ressourcen verarbeiten und darstellen können.

Für das Hinzufügen von weiteren Action Situations und Agententypen ist grundsätzlich nur eine Änderung in der Ontologie-Datenbank nötig. Erst falls die hinzugefügten Action Situations oder Agenten spezifische Eigenschaften haben müssen (beispielsweise wenn bei einem Agententyp monatliche fixe Betriebskosten anfallen, die ihm von seinem Kontostand abgezogen werden müssen), ist eine Änderung am Code nötig. Es müssen dann zusätzliche Wrapper-Klassen (vgl. Kapitel 4.3.1) für die entsprechenden Action Situations / Agenten erstellt werden. Mit Hilfe einer Wrapper-Klasse können die über einen konkreten Agenten in der Ontologie-Datenbank gespeicherten Daten in ein Java-Objekt gefasst werden. Aus Performance-Gründen werden dabei nur die wichtigsten Informationen aus der Datenbank ausgelesen, in den meisten Fällen sogar nur die eindeutige ID des Agenten. Die Wrapper-Klasse stellt nun verschiedene Methoden zur Verfügung, um Eigenschaften des Agenten auszulesen oder zu verändern. So kann beispielsweise der aktuelle Bedarf an einem bestimmten Gut (z.B. Energieholz) erhöht werden. Wird die entsprechende Methode aufgerufen, werden im Hintergrund mittels einer SPARQL-Query die entsprechenden Daten in der Datenbank verändert. Somit wird eine klare Trennung zwischen den Daten und den Funktionen geschaffen: Die Daten sind immer nur in der Datenbank, während die Funktionen in den Wrapper-Klassen gehalten werden.

Um manuelle Prozesse möglichst zu vermeiden, werden sowohl die konkreten Agenten wie aber auch die Basis-Ontologie direkt mit Java-Code erstellt. Mit Basis-Ontologie ist hier die Definition von allen abstrakten Klassen gemeint, wie zum Beispiel *Agent*, *Contract*, *ActionSituation* etc. Die komplette Basis-Ontologie ist in einem Diagramm in Anhang B abgebildet.

7. Schlussfolgerungen

Die vorliegende Masterarbeit zeigte anhand eines konzeptuellen Modells des Schweizer Energieholzmarktes, wie der Ansatz der agentenbasierten Modellierung mit einer Ontologie-Datenbank kombiniert werden kann. Ziel des Ansatzes der Verwendung einer Ontologie war es, einerseits ein nachträgliches Nachvollziehen von allen Interaktionen zwischen den Agenten zu ermöglichen, andererseits verbesserte Auswertungsmöglichkeiten zu erhalten. Die Nachvollziehbarkeit der Interaktionen sollte sichergestellt werden, indem jede Interaktion in der Datenbank festgehalten wird. Die besseren Auswertungsmöglichkeiten sollten sich einerseits dadurch ermöglichen, dass mittels SPARQL-Queries gezielt nach gewünschten Informationen gesucht werden kann, andererseits durch die Anwendung von Inferenz-Regeln. Für das Erstellen einer Ontologie-Datenbank sowie der Interaktion mit dieser wurde das Jena-Framework verwendet. Das Jena-Framework hält in der Standardkonfiguration den kompletten Datenbestand der Datenbank im Arbeitsspeicher. Dadurch ist es zwar relativ schnell, jedoch ist die Grösse der Datenbank auch durch die Grösse des Arbeitsspeichers begrenzt. Durch die Verwendung einer zusätzlichen Framework-Komponente (TDB) wäre es möglich gewesen, die Daten fortwährend auf den Harddisk zu schreiben, und somit den Arbeitsspeicher zu entlasten. Dies hätte jedoch wiederum den Nachteil mit sich gebracht, dass die Simulation nur ca. halb so schnell gewesen wäre. Es wurde deshalb entschieden, auf die Verwendung von TDB zu verzichten, und nur mit der Standard-Datenbank zu arbeiten, welche alle Daten im Arbeitsspeicher hält.

Im Laufe der Arbeit wurde festgestellt, dass die durch den Arbeitsspeicher nach oben begrenzte maximale Datenbankgrösse immer mehr zum Flaschenhals der gesamten Simulation wurde. So musste schliesslich eine Funktion in das Simulationsprogramm integriert werden, welche jeweils nach jeder Ausführung einer Marktrunde (i.d.R. nach einem simulierten Monat) die Datenbank aufräumt, d.h. nicht mehr benötigte Daten aus der Datenbank entfernt. Dieses Vorgehen widerspricht jedoch dem Zweck, für welchen der Ontologie-Ansatz ursprünglich gewählt wurde: die Nachvollziehbarkeit aller Interaktionen und die detaillierten Auswertungsmöglichkeiten durch den grossen Datenbestand. Folglich wurde eine weitere Komponente in das Simulationsprogramm integriert: Der Evaluator. Diese Komponente evaluiert die Daten in der Datenbank und schreibt sie in eine CSV-Datei, bevor sie aus der Datenbank gelöscht werden. Dies hat jedoch wiederum den Nachteil, dass schon vor Beginn der Simulation genau bekannt sein muss, welche Daten ausgewertet werden sollen. Eine nachträgliche Mustersuche in den Daten ist nicht mehr möglich, da durch den Evaluator bereits eine Auswertung und dadurch Komprimierung der Daten stattgefunden hat.

Wäre anstatt der Standard-In-Memory Datenbank von Jena die High-Performance-Datenbank TDB verwendet worden, hätten diese Probleme möglicherweise vermieden werden können, jedoch zu Lasten einer schlechteren Performance. Auf den Wechsel auf TDB wurde jedoch verzichtet, da dadurch nicht nur sämtliche Datenbankanfragen umgeschrieben hätten werden müssen, sondern an diversen Stellen teilweise massive Eingriffe in die Architektur nötig gewesen wären. Dies aus dem folgenden Grund: Während Jena über Java-Klassen direkt das Erstellen von OWL-Ontologien ermöglicht, wurde TDB für die Verwaltung von RDF-Daten entwickelt.

OWL-Ontologien entsprechen aus technischer Sicht zwar ebenfalls der RDF-Syntax; jedoch bietet das Jena-Framework Methoden, welche die direkte Bearbeitung von OWL-Ontologien stark vereinfachen. Würde nun ein Wechsel auf TDB vollzogen werden wollen, müsste auf viele hilfreiche Funktionen von Jena verzichtet werden.

Ein grosser Vorteil der Verwendung einer Ontologie-Datenbank für die Verwaltung der Daten gegenüber dem direkten Halten der Daten in Java-Objekten ist es, dass man sich weniger mit den Java-Speicherstrukturen (d.h. sinnvolle Anwendungen von Vektoren und HashMaps) beschäftigen muss: da alle Daten in der Datenbank gespeichert werden, besitzen die verschiedenen Klassen viel weniger Instanzvariablen, was die Programmierung erleichtert. Demgegenüber steht die Herausforderung, dass ein effizientes Datenbankschema wichtig ist für eine gute Performance, ebenso wie effiziente SPARQL-Queries. Insbesondere bei längeren oder verschachtelten Queries ist die Abfolge der einzelnen Statements innerhalb der Query entscheidend: Es wurde festgestellt, dass Statements in ungünstiger Reihenfolge dazu führen können, dass eine Abfrage, die im Normalfall wenige 100ms dauert, plötzlich mehrere Sekunden, im Extremfall gar Minuten dauern kann.

Grössere Schwierigkeiten bereiteten auch die Verifikation und Validierung des Simulationsprogramms. Durch die hohe Simulationsdauer, welche abhängig von den gewählten Einstellungen zwischen zwei und zehn Stunden dauerte, konnten Verbesserungen und Anpassungen im Code nur mit hohem Zeitaufwand vorgenommen werden, da nach jeder Änderung im Code wieder die entsprechende Zeitspanne abgewartet werden musste. Dieses Problem stellte sich vor allem in den späteren Entwicklungsschritten. In früheren Programmversionen z.B. konnte mittels JUnit-Tests die formale Korrektheit von einzelnen Methoden getestet werden. Erst in den späteren Versionen, als es vermehrt um die Validierung und weniger um die Verifikation ging, stellte sich dieses Problem.

8. Ausblick

Die Hauptschwierigkeiten bei dem implementierten Simulationsprogramm sind die mangelhafte Performance sowie die Probleme im Zusammenhang mit der sehr grossen Datenmenge, die während des Simulationsvorgangs entsteht. Interessant wäre deshalb ein Versuch einer Implementierung mit TDB. Dies würde zwar die Performance nicht verbessern, jedoch liesse sich somit wenigstens das Problem mit der Datenmenge lösen, so dass die Datenbank nicht regelmässig aufgeräumt werden muss, und dabei möglicherweise wichtige Informationen verloren gehen.

Ausserdem könnte geprüft werden, für welche anderen Realsysteme als für den Schweizer Energieholzmarkt das Simulationsprogramm ebenfalls verwendet werden könnte, bzw. welche Anpassungen am Simulationsprogramm dazu nötig wären. Somit könnte für Märkte, welche ähnliche Eigenschaften aufweisen wie der Schweizer Energieholzmarkt, ein generisches Framework geschaffen werden.

In Bezug auf den Schweizer Energieholzmarkt konnten mit der vorliegenden Arbeit noch keine Aussagen getroffen werden. Dies hängt damit zusammen, dass sich sowohl das Debugging wie auch die Validierung des Simulationsprogramms durch die schlechte Performance sehr schwierig gestaltete. Durch weitergehende Verifikation und Validierung wäre jedoch zu erwarten, dass mit Hilfe von Simulationen Aussagen über den Schweizer Energieholzmarkt getroffen werden könnten, insbesondere über den Einfluss des Akteurverhaltens (z.B. unterschiedliche Gewichtung von verschiedenen Entscheidungskriterien) auf die Nutzung von Energieholz. Da in der durchgeführten Sensitivitätsanalyse pro Simulationsdurchgang jeweils nur ein einzelner Parameter isoliert variiert wurde, ist noch kein Wissen darüber vorhanden, wie sich die unterschiedlichen Parameter gegenseitig beeinflussen. Interessant wäre hier die Erstellung von einer wesentlich grössere Anzahl unterschiedlicher Kombinationen von Input-Parametern, damit die Wechselwirkungen von unterschiedlichen Parametern gesehen werden könnten. Dies hat jedoch auch eine weitaus grössere Anzahl von Simulationsdurchgängen zur Folge und setzt entsprechend ein grösseres Zeit- und Rechenleistungsbudget voraus. Da durch eine steigende Anzahl von Inputparametern die Anzahl möglicher Kombinationen der Werte der Input-Parameter exponentiell zunimmt, wäre hier auch eine Monte-Carlo-Simulation denkbar, indem die Input-Parameter pro Simulationsdurchlauf zufällig gewählt werden.

Literaturverzeichnis

[1] Dignum, V. (2004). A model for organizational interaction: based on agents, founded in logic. PhD thesis, University of Utrecht, The Netherlands.

[2] Gamma E., Helm R., Johnson R. and Vlissides J. (2004). Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software. Addison Wesley, München.

[3] Ghorbani, A. (2010). MAIA: An Analysis and Design Framework for Building Artificial Societies. Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011), Yolum, Tumer, Stone, and Sonenberg (eds.), International Foundation for Autonomous Agents and Multiagent Systems, Taipei, Taiwan.

[4] Ghorbani A., Dignum V. and Dijkema G. (2011): An Analysis and Design Framework for Agent-based Social Simulation. In: Proceedings of the First International Workshop an Agent-based Modeling for Policy Engineering (AMPLE 2011). TU Delft, The Netherlands.

[5] Hofer, P. and J. Altwegg. (2008). Holz-Nutzungspotenziale im Schweizer Wald auf Basis LFI3. Geo Partner AG. Schweiz.

[6] Janssen, M. A. and Ostrom E. (2006). Empirically based, agent-based models. Ecology and Society 11(2): 37.

[7] Kostadinov F. and Steubing B. (2011). An agent-based model of an energy wood market in a Swiss region. ESSA 2011. Montpellier, France.

[8] Ontologie (1979). In: Der neue Brockhaus (6. Auflage). Wiesbaden: F.A. Brockhaus.

[9] Ostrom E., Gardner R., and Walker J. (1994). Rules, games, and common-pool resources. University of Michigan Press, USA.

[10] Page, B. (1991). Diskrete Simulation: Eine Einführung mit Modula-2. Berlin, Heidelberg, New York: Springer-Verlag.

[11] Primas A., Müller-Platz C., and Kessler F. M., Basler und Hofmann AG. (2010). Schweizerische Holzenenergiestatistik: Erhebung für das Jahr 2009. Bern: Bundesamt für Energie.

[12] Saaty T. (1980). The Analytic Hierarchy Process. McGraw-Hill, USA.

[13] Saaty T. (1990). How to make a decision: The Analytic Hierarchy Process. University of Pittsburgh, Pittsburgh, USA.

[14] Saaty T. (2008). Decision Making with the analytic hierarchy process. In: Int. J. Services Scienes, Vol. 1, No. 1, 2008. Pittsburg, USA.

[15] Sargent R. (2008). Verification and Validation of Simulation Models. In: Proceedings of the 2008 Winter Simulation Conference. Miami, USA.

[16] Steubing B., Kostadinov F., Ghorbani A., Wäger P., Thees O., and Ludwig C. (2011). Agent-based modelling of a woodfuel market - factors that affect the availability of woodfuel. ISIE 2011. Berkeley, California, USA.

[17] Steubing B., Zäh R., Waeger P., and Ludwig C. (2010). Bioenergy in Switzerland: Assessing the domestic sustainable biomass potential. Renewable and Sustainable Energy Reviews 14:2256–2265.

[18] Thees O., Frutig F., Breitenstein M., Lemm R., Kaufmann E., and Keilen K. (2003). Schätzung des Potenzials an Energieholz im Schweizer Wald und Kalkulation der Bereitstellungspreise. Eidg. Forschungsanstalt für Wald, Schnee und Landschaft (WSL), Birmensdorf, Switzerland.

Anhang

A. Annahmen

Allgemein

Nr.	Annahme / Entscheid
MA-ALLG-01	Es gibt keine Teuerung
MA-ALLG-02	Langzeitverträge existieren nur im Woodfuel Market
MA-ALLG-03	Langzeitverträge haben alle eine fixe Laufzeit von 10 Jahren, ausser bei Small Private Woodfuel Consumer, wo die Vertragslaufzeit zufällig 1, 2 oder 3 Jahre beträgt
MA-ALLG-04	In einem Langzeitvertrag wird während seiner Laufzeit alle 3 Monate die selbe Menge geliefert (Ausnahmen siehe Subtypen)
MA-ALLG-05	In Langzeitverträgen passt sich der vereinbarte Preis zu 40% dem Ölpreis an. (Variation möglich durch Parameter-Einstellung)
MA-ALLG-06	Der Waldbestand des Kanton Aargau beträgt bei Simulationsbeginn 48'707 ha
MA-ALLG-07	An externe Märkte angeschlossene Marktteilnehmer können immer verkaufen. Der Verkaufspreis ist durch externe Variablen vorgegeben (internationaler Marktpreis)
MA-ALLG-08	Ein Einfamilienhaus benötigt eine jährliche Heizleistung von 22'500 kWh
MA-ALLG-09	Bei der Initialisierung wird für jeden Anbieter ein Preis für sein Gut festgelegt. Dieser Preis passt sich während der Simulation zu 40% dem Ölpreis an.
MA-ALLG-10	Ein einzelner Anbieter bietet seine Güter für sämtliche Abnehmer zum selben Preis an (unabhängig von der gekauften Menge etc.)
MA-ALLG-11	Die bei der Initialisierung festgelegten Preise (MA-ALLG-09) sind innerhalb der Rolle des Agenten normalverteilt, wobei der Durchschnittswert innerhalb der Rolle fix ist, und die Standardabweichung bei 10% des festgelegten Durchschnittspreises liegt.
MA-ALLG-12	Der Freundschafts-Wert ist ein Zufallswert zwischen 0 und 1, wobei 0 keine Freundschaft, 1 maximale Freundschaft bedeutet. Es gibt keine ‚negative Freundschaften' / Feindschaften
MA-ALLG-13	Intermediäre kaufen nie bei anderen Intermediären ein
MA-ALLG-14	Ein Agent geht Konkurs, wenn Ende Jahr sein Kontostand < 0 ist.
MA-ALLG-15	Ausser Forester, Private Forest Owner und District Heating Network Operator können alle Agententypen Konkurs gehen.
MA-ALLG-16	Die Liefermenge in einem Langzeitvertrag wird bestimmt durch den Bedarf des Käufers bei Vertragsabschluss, d.h. wenn er zum Zeitpunkt des Vertragsabschlusses einen Bedarf von 10 Einheiten hat, werden danach während der Laufzeit des Vertrages in jeder Lieferung 10 Einheiten geliefert.

MA-ALLG-17	Die von Forester / Private Forest Owner auf den Markt gebrachte Menge Woodfuel wird durch die Menge des zuvor verkauften Industrial Roundwood bestimmt.
MA-ALLG-18	Forester / Private Forest Owner verarbeiten ihr Holz in Industrial Roundwood und Woodfuel, nachdem sie wissen, wieviel Industrial Roundwood sie verkauft haben. (D.h. die Verarbeitung von der Ressource *Forest* in Industrial Roundwood und Woodfuel findet zwischen den Ausführungen des Industrial Roundwood Market und des Roundwood Market statt.)
MA-ALLG-19	Die von Forester / Private Forest Owner maximal zu verkaufende Menge Holz pro Runde wird durch den (monatlichen) Hiebsatz bestimmt (+ allfällige Vorräte).
MA-ALLG-20	Komplettoutsourcer treten nicht im Woodfuel Market bzw Industrial Roundwood Market auf, da sie ihr Holz direkt an einen Harvest-and-Thinning-Anbieter verkaufen. (siehe auch MA-RES-08)
MA-ALLG-21	Sowohl Käufer wie auch Verkäufer bezahlen für jedes Geschäft CHF 50 Transaktionskosten (d.h. der Preis pro Einheit beim Kauf einer grösseren Menge ist günstiger als beim mehrmaligen Kauf von kleinen Mengen)
MA-ALLG-22	Die Distanzen zwischen den geographischen Positionen der Agenten sind homogen (d.h. topologische Eigenschaften wie Berge oder Flüsse sowie Verkehrswege werden beim Vergleich von Distanzen nicht berücksichtigt).

Ressourcen

Nr.	Annahme / Entscheid
MA-RES-01	Es gibt keine Unterscheidung zwischen unterschiedlichen Holzarten. Die (angenommene) verwendete Holzart besteht aus einem Mix aus 28% Buche und 72% Fichte
MA-RES-02	Wald regeneriert sich jährlich um 2.6% (= 0.214% monatlich)
MA-RES-03	Der jährliche Hiebsatz beträgt 2.6% (= 0.214% monatlich)
MA-RES-04	Ein ha Wald entspricht 336.6 m^3 Holz (Festmeter)
MA-RES-05	1 Festmeter (fm) entspricht 2.5 Schüttmetern (srm)
MA-RES-06	Die Dichte von Holz entspricht 530 kg/m^3 (entsprechend MA-RES-01)
MA-RES-07	1kg Woodfuel hat einen Energiegehalt von 4.5 kWh
MA-RES-08	Kauft jemand eine HarvestAndThinning-Dienstleistung, dann übergibt er dem Harvest and Thinning-Anbieter sämtliches von diesem geerntetes Holz. Der Harvest and Thinning-Anbieter bezahlt im Gegenzug dem Käufer der Dienstleistung nur 80% des Marktüblichen Holzpreises (die restlichen 20% entsprechen seinen Kosten für die Ernte).
MA-RES-09	Im Harvest and Thinning Market sind Anbieter und Abnehmer nie mehr als 25km voneinander entfernet (Einschränkung: MA-RES-20).
MA-RES-10	Im Woodfuel Market sind Anbieter und Abnehmer nie mehr als 10km voneinander entfernt (Einschränkung: MA-RES-20).

MA-RES-11	Im Industrial Roundwood Market sind Anbieter und Abnehmer nie mehr als 10km voneinander entfernt (Einschränkung: MA-RES-20).
MA-RES-12	Im Harvest and Thinning Market kann ein Käufer nur von den 10 am nächsten bei ihm liegenden Anbietern einkaufen
MA-RES-13	Im Woodfuel Market kann ein Käufer nur von den 10 am nächsten bei ihm liegenden Anbietern einkaufen
MA-RES-14	Im Industrial Roundwood Market kann ein Käufer nur von den 10 am nächsten bei ihm liegenden Anbietern einkaufen
MA-RES-15	Ob Akteure in den Markt eintreten oder daraus austreten, hängt von der Attraktivität des Woodfuel-Marktes ab. Ist diese positiv, treten Akteure in den Markt ein; ist sie negativ, treten Akteure aus dem Markt aus. Die Attraktivität wird berechnet, indem aus dem Ölpreis der vergangenen 60 Monaten eine Tendenz berechnet wird. Ist die Tendenz steigend, steigt auch die Attraktivität des Woodfuel-Marktes. Ebenfalls berücksichtigt wird die Woodfuel-Knappheit. Je knapper die Ressource Woodfuel (siehe MA-RES-18), desto unattraktiver der Woodfuel Market.
MA-RES-16	Bei minimalster Attraktivität des Woodfuel-Marktes verlassen 50% der Agenten, die ihre Heizung erneuern müssten, den Markt.
MA-RES-17	Bei maximaler Attraktivität des Woodfuel-Marktes vergrössert sich die Anzahl Agenten um 10%
MA-RES-18	Die Knappheit von Woodfuel wird bestimmt, in dem das gehandelte Marktvolumen durch die maximal zu produzierende Menge Energieholz in einem Monat geteilt wird. Die maximal zu produzierende Menge Energieholz in einem Monat wird angenommen als: Waldbestand * (Hiebsatz pro Monat) * 0.2
MA-RES-19	Bei maximaler Woodfuel-Knappheit ist der Preis doppelt so hoch wie er bei minimaler Knappheit wäre.
MA-RES-20	Findet ein Abnehmer innerhalb seines Suchradius keine Anbieter, vergrössert er seinen Suchradius.

Forester

Nr.	Annahme / Entscheid
MA-FRTR-01	80% des Gesamtwaldbestandes wird von Förstern verwaltet
MA-FRTR-02	Ein Förster entscheidet zu Beginn des Jahres, wie viel Prozent von jedem gefällten Baum er für Energieholz, und wie viel für Rundholz verwendet
MA-FRTR-03	Die Menge Wald pro Forester ist normalverteilt, mit einer Standardabweichung von 25% des Durchschnittsbestandes aller Förster
MA-FRTR-04	Der Durchschnittspreis für Woodfuel liegt bei der Initialisierung bei CHF 80 / srm
MA-FRTR-05	Der Durchschnittspreis für Industrial Roundwood liegt bei der Initialisierung bei CHF 250 / m3

MA-FRTR-06	50% aller Förster sind Komplettoutsourcer (Variation möglich in Parameter-Einstellung)

Private Forest Owner

Nr.	Annahme / Entscheid
MA-PFO-01	20% des Gesamtwaldbestandes wird von Private Forest Owner verwaltet
MA-PFO-02	Bei einem Private Forest Owner kann auch ein Forester das Harvest and Thinning übernehmen
MA-PFO-03	Ausser dem Subtyp *Farmer Type* sind alle Private Forest Owner Komplett-toutsourcer, d.h. ein Anbieter auf dem Harvest and Thinning Market übernimmt die Ernte und den Verkauf des Holzes.
MA-PFO-04	Die Menge Wald pro Private Forest Owner ist normalverteilt, mit einer Standardabweichung von 25% des Durchschnittsbestandes aller PrivateForestOwner
MA-PFO-05	Der Durchschnittspreis für Woodfuel liegt bei der Initialisierung bei CHF 80 / srm
MA-PFO-06	Der Durchschnittspreis für Industrial Roundwood liegt bei der Initialisierung bei CHF 250 / m3

Forest Company

Nr.	Annahme / Entscheid
MA-FCO-01	Der Durchschnittspreis für Woodfuel liegt bei der Initialisierung bei CHF 90 / srm
MA-FCO-02	Der Durchschnittspreis für Industrial Roundwood liegt bei der Initialisierung bei CHF 275 / m3

Bundling Organization

Nr.	Annahme / Entscheid
MA-BORG-01	Der Durchschnittspreis für Woodfuel liegt bei der Initialisierung bei CHF 90 / srm
MA-BORG-02	Der Durchschnittspreis für Industrial Roundwood liegt bei der Initialisierung bei CHF 275 / m3

Sawmill

Nr.	Annahme / Entscheid
MA-SAW-01	Eine Sawmill hat als Output fix 40% Energieholz und 60% Rundholz

MA-SAW-02	Der Durchschnittspreis für Woodfuel liegt bei der Initialisierung bei CHF 100 / srm
MA-SAW-03	Eine Sawmill verarbeitet pro Monat 1000m3 (fm) Holz.

Small Private Woodfuel Consumer

Nr.	Annahme / Entscheid
MA-SPWC-01	Ein Small Private Woodfuel Consumer benötigt die Heizleistung eines Einfamilienhauses (siehe MA-ALLG-08)
MA-SPWC-02	Die Heizung eines Small Private Woodfuel Consumer hat eine Lebensdauer von 15 Jahren
MA-SPWC-03	Der Einbau / das Ersetzen einer Heizung kostet CHF 15'000
MA-SPWC-04	Ein Small Private Woodfuel Consumer kauft Woodfuel, sobald seine Restmenge weniger als einen Monatsbedarf beträgt. Er kauft dann einen dreifachen Monatsbedarf ein.
MA-SPWC-05	Schliessen Small Private Woodfuel Consumer Langzeitverträge ab, erhalten sie alle 3 Monate dieselbe Menge.

Public Woodfuel Consumer

Nr.	Annahme / Entscheid
MA-PWFC-01	Ein Public Woodfuel Consumer benötigt jährlich fix 350 Schüttmeter Energieholz
MA-PWFC-02	Die Heizung eines Public Woodfuel Consumer hat eine Lebensdauer von 15 Jahren
MA-PWFC-03	Der Einbau / das Ersetzen einer Heizung kostet CHF 3'000'000
MA-PWFC-04	Public Woodfuel Consumer kaufen monatlich ein (jeweils 1/12 des Jahresbedarfs)
MA-PWFC-05	Schliessen Public Woodfuel Consumer Langzeitverträge ab, erhalten sie jeden Monat dieselbe Menge.

Commercial Woodfuel Consumer

Nr.	Annahme / Entscheid
MA-CWFC-01	Ein Commercial Woodfuel Consumer benötigt jährlich fix 350 Schüttmeter Energieholz
MA-CWFC-02	Die Heizung eines Commercial Woodfuel Consumer hat eine Lebensdauer von 15 Jahren
MA-CWFC-03	Der Einbau / das Ersetzen einer Heizung kostet CHF 3'000'000
MA-CWFC-04	Commercial Woodfuel Consumer kaufen monatlich ein (jeweils 1/12 des Jahresbedarfs)

MA-CWFC-05	Schliessen Commercial Woodfuel Consumer Langzeitverträge ab, erhalten sie jeden Monat dieselbe Menge.

Pulpwood Consumer

Nr.	Annahme / Entscheid
MA-PULP-01	Ein Pulpwood Consumer benötigt monatlich eine fixe Holzmenge von 100'000 Schüttmetern

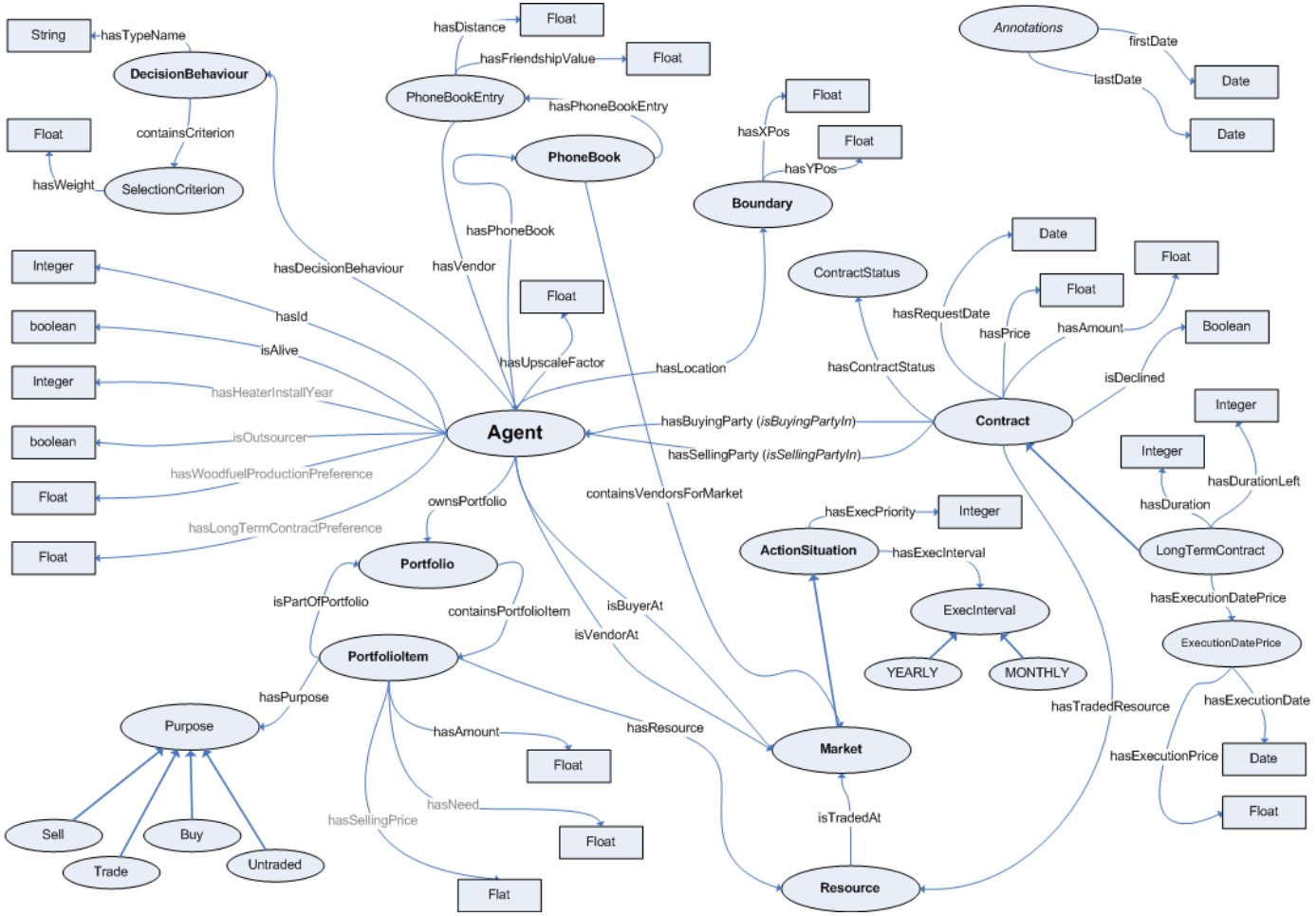
District Heating Network Operator

Nr.	Annahme / Entscheid
MA-DHNO-01	Ein District Heating Network Operator erzeugt aus dem Energieholz fix 25% Strom und 55% Wärme (bei 20% Verlust). Umwandlung Energieholz -> Strom / Energieholz -> Wärme siehe MA-RES-07
MA-DHNO-02	Der Durchschnittspreis für Heat liegt bei der Initialisierung bei CHF 200 / MWh
MA-DHNO-03	Ein District Heating Network Operator hat einen Maximalradius von 5km. Agenten können nur innerhalb dieses Radius mit Wärme versorgt werden.
MA-DHNO-04	Die Heizung eines District Heating Network Operator hat eine Lebensdauer von 15 Jahren
MA-DHNO-05	Der Einbau / das Ersetzen einer Heizung kostet CHF 10'000'000
MA-DHNO-06	Ein District Heating Network Operator bestimmt seinen Bedarf aufgrund der Anzahl seiner Kunden

Small Private Heat Consumer

Nr.	Annahme / Entscheid
MA-SPHC-01	Ein Small Private Heat Consumer ist immer mit genau einem District Heating Network Operator verbunden
MA-SPHC-02	Small Private Heat Consumer können nur dort entstehen, wo ein District Heating Network Operator in der Nähe ist (d.h. der District Heating Network Operator muss zuerst entstehen).

B. Ontologie


String
hasTypeName
DecisionBehaviour
Float
containsCriterion
hasWeight
SelectionCriterion
hasDistance
Float
hasFriendshipValue
Float
PhoneBookEntry
hasPhoneBookEntry
PhoneBook
hasXPos
Float
Float
hasYPos
Boundary
Annotations
firstDate
Date
lastDate
Date
hasPhoneBook
hasVendor
Integer
hasDecisionBehaviour
hasId
boolean
isAlive
Integer
hasHeaterInstallYear
boolean
isOutsourcer
Float
hasWoodfuelProductionPreference
Float
hasLongTermContractPreference
Float
hasUpscaleFactor
hasLocation
Agent
ownsPortfolio
containsVendorsForMarket
hasBuyingParty (isBuyingPartyIn)
hasSellingParty (isSellingPartyIn)
ContractStatus
hasContractStatus
Date
hasRequestDate
Float
hasPrice
Float
hasAmount
Boolean
isDeclined
Contract
Integer
Integer
hasDuration
hasDurationLeft
LongTermContract
hasExecPriority
Integer
ActionSituation
hasExecInterval
ExecInterval
YEARLY
MONTHLY
hasExecutionDatePrice
ExecutionDatePrice
hasExecutionDate
Date
hasExecutionPrice
Float
hasTradedResource
Portfolio
isPartOfPortfolio
containsPortfolioItem
isBuyerAt
isVendorAt
PortfolioItem
hasPurpose
Purpose
Sell
Buy
Trade
Untraded
hasAmount
Float
hasNeed
Float
hasSellingPrice
Flat
hasResource
Market
isTradedAt
Resource

C. Wichtigste Markteigenschaften

Nachfolgend werden einige Markteigenschaften aufgelistet, welche typisch sind für den Schweizer Energieholzmarkt.

Eigenschaften der gehandelten Guter:

- Knappe Güter
- Produktionsort des Gutes ist geographisch fixiert (Wald)
- Konsumationsort des Gutes ist geographisch fixiert (Heizung)
- Güters sind kurzfristig nicht substituierbar, langfristig schon (z.B. durch Einbau von anderen Heizungstypen)
- Kein aus verschiedenen Bestandteilen zusammengesetztes Gut (wie z.B. ein Hammer aus Holz und Metall), Verarbeitung des Gutes (Rundholz -> Schnitzel) jedoch möglich
- Gut (Holzschnitzel) ist heute eher eine Neben- bzw. Abfallprodukt von der Rundholzverarbeitung. Bei stark steigenden Ölpreisen konnte sich dies ändern.
- Holz wird vereinfacht als homogenes Produkt angenommen, d.h. keine Qualitätsunterschiede zwischen verschiedenen Holzsorten
- Preis des Gutes ist stark abhängig von einem anderen Gut (Bindung zum Ölpreis)

Eigenschaften des Marktes, der Agenten und der Interaktion:

- Kurze Transportdistanzen sind wichtig (z.B. nicht wie bei einer Bestellung eines Buches bei Amazon)
- Soziale Aspekte wie z.B. Freundschaft sind wichtig für Interaktion / den Handel
- Langzeit- und Kurzzeitverträge
- Agenten können in den Markt ein- oder austreten. Jedoch existieren hohe Ein- und Austrittsbarrieren (z.B. Einbau einer Öl- oder einer Holzheizung)
- Es gibt Intermediäre auf dem Markt. Diese bestimmen ihren Bedarf anhand vergangener Verkaufe
- Ablauf des Güteraustausches in vier Phasen, Nachfrage-getrieben (Details siehe Modell von Steubing et al. [16], bzw. Kapitel 3.3.3)
- Förster stehen im Gegensatz zu kommerziellen Betrieben nicht unter hohem Gewinndruck

D. Programmargumente

Das Simulationsprogramm kann verschiedene Programmargumente entgegennehmen. Diese sind in nachfolgender Tabelle erklärt.

Argument	Bedeutung
Start ohne Argumente	Das Simulationsprogramm wird im GUI-Modus mit Standardeinstellungen gestartet
-gui	Das Simulationsprogramm wird im GUI-Modus gestartet. Dieses Argument ist zwingend notwendig, falls danach weitere Einstellungen mittels Programmargumenten vorgenommen werden sollen
-console	Das Simulationsprogramm wird im Konsolen-Modus gestartet. Dieses Argument ist zwingend notwendig, falls danach weitere Einstellungen mittels Programmargumenten vorgenommen werden sollen
-ontology Pfad	Pfad der Datei, in welche die Ontologie geschrieben wird
-createNew Anzahl	Anzahl zu erstellender Agenten bei Beginn der Simulation
-log loglevel	Log-Level. Anhand des Log-Levels wird festgelegt, wie feinkörnig die Informationen geloggt werden. Mögliche Log-Level sind Off, Severe, Warning, Info, Config, Fine, Finer, Finest, All.
-logfile Pfad	Pfad der Logdatei
-runYears Anzahl	Anzahl zu simulierende Jahre
-firstYear Jahr	Jahreszahl des ersten zu simulierenden Jahres
-maxThreads Anzahl	Maximale Anzahl paralleler Threads. Wird 0 als Anzahl übergeben, werden keine Threads verwendet.
-cleanUp true/false	Legt fest, ob die Ontologie-Datenbank nach jedem simulierten Monat aufgeräumt wird, d.h. nicht mehr benötigte Daten gelöscht werden.
-paramFile Pfad	Pfad der Datei, welche die Parameter enthält (siehe Kapitel 4.3.4)
-outputFile Pfad	Pfad der Output-Datei (CSV-Datei mit Auswertungen)
-randomSeed Seed	Random-Seed (siehe Kapitel 4.3.6)
-savePNG true/false	Legt fest, ob die Diagrammen in dem Simulationsprogramm automatisch in PNG-Dateien (Bilddateien) gespeichert werden sollen.

Tabelle D.1.: Mögliche Programmargumente für das Simulationsprogramm. Gross-/Kleinschreibung spielt bei der Eingabe der Argumente keine Rolle.