

# WISSENSREPRÄSENTATION UND VERARBEITUNG

## PROJEKTARBEIT



EINGEREICHT VON  
Fabia Holzer  
Elena Pineider  
Kerem Akkaya  
Bastian Büeler

FRÜHLINGSSEMESTER 2025

# Inhaltsverzeichnis

Abstract.....	5
Einleitung .....	6
1    Aufgabe 1.....	7
1.1   Aufgabenstellung .....	7
1.2   Entscheidungsmodell .....	7
1.2.1   Unser Entscheidungsmodell .....	7
1.2.2   Entscheidungstabelle – Subdecision: Schwierigkeitsgrad.....	8
1.2.3   Entscheidungstabelle – Subdecision Themenwelt .....	10
1.2.4   Entscheidungstabelle – Subdecision Erfahrung.....	12
1.2.5   Entscheidungstabelle – Subdecision Zielgruppe.....	13
1.2.6   Entscheidungstabelle – Lego Set .....	14
1.2.7   Test Cases .....	15
2    Aufgabe 2.....	16
2.1   Aufgabenstellung .....	16
2.2   Mögliche Kundenanfragen .....	16
2.2.1   Geschenk-Szenarien.....	16
2.2.2   Erfahrungs- & Kompetenz-basiert .....	17
2.2.3   Themenwelt-Interessierte .....	17
2.2.4   Nutzerzentriert (individuelle Präferenzen).....	17
2.2.5   Budget-orientiert.....	17
2.2.6   Sammlung Produkt und Kundenbedürfnisse.....	17
2.2.7   Produktbezogene Informationen (Eigenschaften der LEGO-Sets).....	18
2.2.8   Kundenbezogene Informationen (Bedarf und Präferenzen).....	18

2.3	Implementierung der Wissensbasis und Regeln in Prolog .....	19
2.3.1	Faktenbasis .....	19
2.3.2	Regel zur Bestimmung der Zielgruppe .....	21
2.3.3	Regel zur Bestimmung der Erfahrung .....	22
2.3.4	Regel zur Bestimmung des Schwierigkeitsgrads .....	23
2.3.5	Regel zur Bestimmung der Themenwelt .....	24
2.3.6	Regel zur Bestimmung des Preisbereichs .....	25
2.3.7	Regel zur Auswahl eines Kunden und deren Fakten .....	26
2.3.8	Fakten Check mit eventueller Liste .....	27
2.3.9	Regel Gesamtempfehlung eines LEGO-Sets von Kundenempfehlung .....	27
3	Aufgabe 3 .....	29
3.1	Aufgabenstellung .....	29
3.2	Ontologieschema .....	29
3.2.1	Zentrale Klassen: .....	29
3.2.2	Objekteigenschaften (Domains → Ranges): .....	30
3.2.3	Dateneigenschaften (Domains → Ranges): .....	31
3.2.4	Klassenhierarchie .....	32
3.3	SWRL-Regeln .....	33
3.3.1	Erfahrung .....	33
3.3.2	Themenwelt .....	37
3.3.3	Zielgruppe .....	40
3.3.4	Schwierigkeitsgrad .....	44
3.3.5	LegoSet Empfehlungen Regeln .....	46
3.4	SPARQL-Abfragen .....	51
3.5	Visualisierung des Wissensgraphen .....	54
3.6	Umsetzung in Protégé .....	55

4	Abschluss.....	56
	Abbildungen .....	57
	Anhang/Anhänge.....	58
	Anhang 1: Eigenständigkeitserklärung .....	58
	Anhang 2: Hilfsmittelverzeichnis .....	60
	Anhang 3: DMN Versionenübersicht mit iterativen Erläuterungen .....	64
	Anhang 4: Coaching Protokoll vom 04. April 2025.....	66
	Anhang 5: Coaching Protokoll vom 30. April 2025.....	66
	Anhang 6: Coaching Protokoll vom 21. Mai 2025 .....	67

## Abstract

Im Rahmen dieses Projekts haben wir ein intelligentes Empfehlungssystem für LEGO-Sets entwickelt, das individuelle Kundenpräferenzen automatisch berücksichtigt und passende Produkte vorschlägt. Ausgangspunkt war die Idee, den Auswahlprozess im LEGO-Online-Shop zu vereinfachen, etwa für Eltern, die ein Geschenk für ihr Kind suchen, oder für erwachsene Fans, die sich für Technik-Modelle interessieren.

Unser System analysiert Informationen wie Alter, Erfahrung, Interessen, Preisvorstellungen oder auch Vorlieben bezüglich Bauweise und Bauteilen, um daraus passende Zielgruppen und einen Schwierigkeitsgrad abzuleiten. Basierend darauf werden thematisch passende LEGO-Sets vorgeschlagen, wie beispielsweise ein einfaches City-Set für ein Kind ohne Bauerfahrung oder ein komplexes LEGO-Technik-Modell für einen erfahrenen Erwachsenen.

Zur Umsetzung werden mehrere Methoden der Wissensrepräsentation kombiniert:

Ein **DMN-Modell** (Decision Requirements Diagram) ermöglicht die strukturierte Entscheidungslogik mit Subdecisions.

In **Prolog** werden die Regeln formal abgebildet, um komplexe Schlussfolgerungen programmatisch ableiten zu können.

In **Protégé** wird schliesslich eine Ontologie modelliert, in der mithilfe von **OWL**, **SWRL** und **SPARQL** Kundenprofile und Produktempfehlungen semantisch verknüpft werden. So kann man den Wissensgraphen nachvollziehbar anzeigen lassen.

Das Projekt zeigt anschaulich, wie theoretische Konzepte aus der Wissensverarbeitung auf ein reales, nachvollziehbares Beispiel wie die LEGO-Produktempfehlung übertragen werden können und wie dadurch auch einfache Kundenfragen ("Welches LEGO-Set passt zu mir?") durchdacht und nachvollziehbar beantwortet werden können.

## Einleitung

Wissensbasierte Systeme stellen eine zentrale Form der künstlichen Intelligenz dar, die auf Regeln, Logik und ontologischen Modellen basieren, um intelligente Schlussfolgerungen zu ziehen. Diese Systeme spielen eine immer wichtigere Rolle in unserer digitalisierten Gesellschaft, da sie komplexe Entscheidungsprozesse strukturieren und automatisieren können. Typische Anwendungsgebiete solcher Expertensysteme finden sich in der Medizin, der Technik und insbesondere im E-Commerce-Bereich, wo sie personalisierte Empfehlungen ermöglichen.

Für Lego als führenden Anbieter von kreativen Bauspielzeugen besteht die zentrale Business-Herausforderung darin, Kunden in der riesigen Produktvielfalt zielgerichtet zu beraten. Das System soll den Kundenberatern dabei helfen Kunden relevante Produkte zu empfehlen.

Lego-Kundinnen und -Kunden stehen heute vor einer riesigen Vielfalt von Sets, die sich in Preis, Bautechnik und Themenwelt unterscheiden. Diese Vielfalt ist einerseits ein Wettbewerbsvorteil, führt aber auch zu Beratungs- und Entscheidungsaufwand. Unser Projekt zeigt, wie ein wissensbasiertes Empfehlungssystem diesen Prozess systematisch von der Bedarfserhebung bis zur transparent begründeten Produktempfehlung unterstützt.

Den roten Faden bildet ein Decision Model and Notation (DMN)-Modell, das alle relevanten Produkt- und Kundenmerkmale als Entscheidungslogik abbildet. Auf dieser Grundlage werden die Regeln in Prolog implementiert und anschliessend in eine OWL-Ontologie überführt. So entsteht ein durchgängiger Wissenspfad, der jederzeit erklärt, warum ein bestimmtes Set vorgeschlagen wurde.

Die folgende Dokumentation führt Schritt für Schritt durch diesen Wissenspfad und validiert ihn anhand konkreter Testfälle. Damit wird deutlich, wie sich theoretische Methoden der Wissensrepräsentation in eine praxisnahe Lösung für den digitalen Einzelhandel überführen lassen.

# 1 Aufgabe 1

## 1.1 Aufgabenstellung

«Erstellen Sie ein Entscheidungsmodell, das Produkte für eine Situation zusammenstellt. Zum Beispiel Computer, die für Gamen oder Programmieren geeignet sind. Kleider für Arbeit oder Freizeit. Überlegen sie sich die für die Entscheidung relevanten Eingaben und die möglichen Ausgaben.

Erstellen Sie eine Entscheidungsmodell, das eine Unterentscheidung enthält, also eine Entscheidung, deren Ergebnis als Eingabe für die Hauptentscheidung verwendet wird.»

## 1.2 Entscheidungsmodell

### 1.2.1 Unser Entscheidungsmodell

Unser wissensbasiertes Lego-Empfehlungssystem basiert auf einer detaillierten Analyse der offiziellen Lego-Webseite, aus der wir entscheidende Produktkriterien und Kaufparameter extrahiert haben. Die finale Modellversion zeigt die optimierte Struktur unseres Systems, dass sich auf die wesentlichen Entscheidungsfaktoren konzentriert. Durch die systematische Auswertung der Lego-Produktdaten und die gezielte Reduktion überflüssiger Kriterien ist ein schlankes, aber präzises Modell entstanden, das reale Kundenbedürfnisse effizient abbildet.

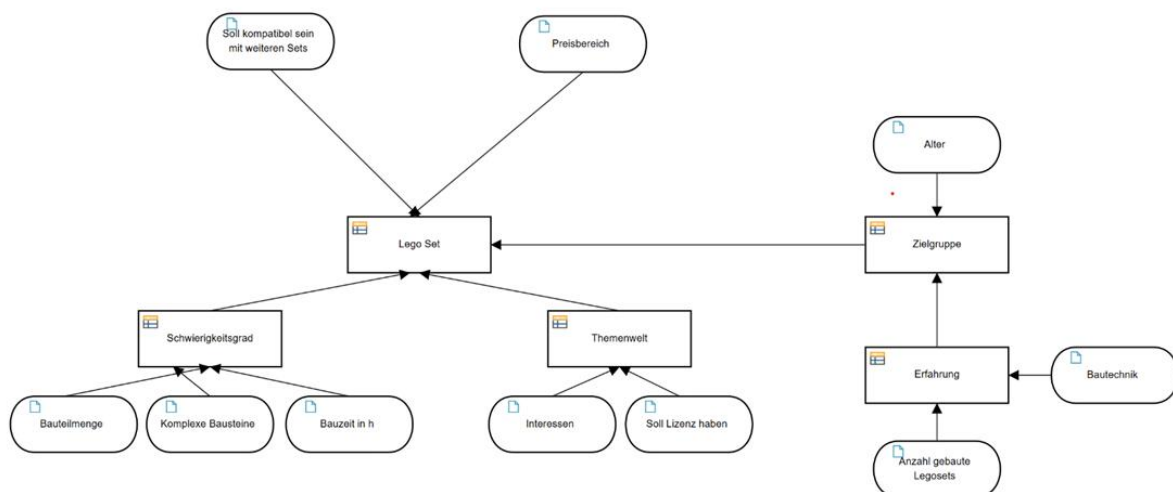


Abbildung 1: DMN Entscheidungsmodell für LEGO Sets

Das Modell enthält vier Subdecisions, welche direkt oder indirekt in die finale Entscheidung einfließen. Die Ausgabe ist ein spezifisches Set, welches dann dem Kunden empfohlen wird.

### **1.2.2 Entscheidungstabelle – Subdecision: Schwierigkeitsgrad**

Lego spricht eine äusserst breite Demografie an. Von unerfahrenen Kindern bis hin zu erfahrenen erwachsenen Lego-Begeisterten. Gerade deshalb war es uns besonders wichtig, jedem Kunden ein Lego-Set zu empfehlen, dass optimal zu seinem individuellen Erfahrungslevel passt. Auf diese Weise möchten wir sicherstellen, dass sowohl Anfänger als auch Fortgeschrittene ein passendes Bauerlebnis finden, das weder unterfordert noch überfordert.

#### **Inputs:**

##### **Bauteilmenge:**

Wenig Bauteile bieten mehr Übersicht und deuten auf ein einfacheres Set hin während Sets mit vielen Bausteinen auf eine höhere Komplexität des Sets hindeuten.

Datentyp: Integer -> Nummer: 0 – 12'000 Bausteine

##### **Bauzeit in h:**

Geschätzte Bauzeit für einen durchschnittlichen Nutzer. Eine kurze Bauzeit suggeriert ein einfacheres Set, da es schnell gebaut werden kann und daher simpler ist. Eine lange Bauzeit weist auf ein komplexeres Modell hin, welches dabei mehr Ausdauer Geduld und auch mehr technisches Verständnis fordert.

Datentyp: Integer -> Nummer: 1 – 20 Stunden

##### **Komplexe Bausteine:**

Viele LEGO-Sets enthalten eine Vielzahl an speziellen oder ungewöhnlichen Bauteilen, die den Schwierigkeitsgrad des Modells erheblich erhöhen können. Diese speziellen Elemente erfordern vom Nutzer ein höheres Mass an Erfahrung.

Datentyp: Boolean -> Binäre Aussage true/false



	inputs			outputs	annotations
U	Bauteilmenge	Bauzeit in h	Komplexe Bausteine	Schwierigkeitsgrad	Description
	<i>Number</i> (0..12000)	<i>Number</i> (1..20)	<i>Boolean</i>	<i>t_Schwierigkeitsgrad</i> "Einfach", "Mittel", "Schwer"	
1	(0..1500]	<=2	false	"Einfach"	
2	<=500	>2	false	"Mittel"	
3	(0..1500]	[1..5]	true	"Mittel"	
4	>=500	[2..5]	false	"Mittel"	
5	>1500	[1..5]	true	"Mittel"	
6	>=500	>5	false	"Schwer"	
7	-	>5	true	"Schwer"	

**Hit-Policy:** *Unique*

**Mögliche Outputs:** «Einfach», «Mittel», «Schwer»

### **1.2.3 Entscheidungstabelle – Subdecision Themenwelt**

Die «Themenwelt» ist ein zentraler Bestandteil bei der Empfehlung eines geeigneten LEGO-Sets, da sie die individuellen Interessen der Nutzer berücksichtigt. LEGO bietet eine Vielzahl an Themenwelten an – von klassischen Fahrzeugmodellen über Fantasy-Abenteuer, bis hin zu lizenzierten Serien und Filmen, wie Star Wars oder Disney. Um sicherzustellen, dass das empfohlene Set wirklich den persönlichen Vorlieben entspricht und damit die Begeisterung für das Bauen gesteigert wird, wird anhand der gewählten Interessen und der Frage nach einer gewünschten Lizenz gezielt eine passende Themenwelt ausgewählt. Dadurch wird die Produktempfehlung deutlich präziser und auf die Wünsche der jeweiligen Zielgruppe zugeschnitten.

#### **Inputs:**

##### **Interessen:**

Beschreibt die thematischen Vorlieben des Nutzers. Dabei kann zwischen verschiedenen Interessensgebieten gewählt werden, wie beispielsweise Fahrzeuge, Disney, Superhelden, Architektur, Sci-Fi, Tiere und Natur.

Datentyp: Text

##### **Soll Lizenz haben:**

Gibt an, ob der Nutzer ein Set bevorzugt, das auf einer bekannten Lizenz basiert. Lizenzen wie Star Wars, Disney oder Marvel stehen hier im Vordergrund. Lego hat neben lizenzierten Sets eigene Themenwelten, wie Technik, Architektur und weitere.

Datentyp: Boolean -> Binäre Aussage: true/false

	inputs		outputs	annotations
	Interessen	Soll Lizenz haben	Themenwelt	Description
U	$\tau_{Interessen}$ "Fahrzeuge", "Disney", "Superheroes", "Architektur", "Sci-Fi", "Kinderserie / Film", "Ninjas", "Gaming", "Sport", "Tier/Natur", "Abenteuer", "Historische Epochen", "Fantasy", "Simple Spiele"	$Boolean$ $not(null)$	$\tau_{Themenwelt}$ "Disney", "Marvel", "Harry Potter", "Ninjago", "Star Wars", "Minecraft", "City", "Technik", "Architektur", "Friends", "Creator", "Classic"	
1	"Fahrzeuge"	false	"Technik"	
2	"Fahrzeuge"	true	"Star Wars"	
3	"Disney"	false	"Friends"	
4	"Disney"	true	"Disney"	
5	"Superheroes"	false	"Ninjago"	
6	"Superheroes"	true	"Marvel"	
7	"Architektur"	false	"Architektur"	

Abbildung 2: Entscheidungstabelle – Subdecision Themenwelt

**Hit-Policy:** *Unique*

**Mögliche Outputs:** «Disney», «Marvel», «Harry Potter», «Ninjago», «Star Wars», «Minecraft», «City», «Technik», «Architektur», «Friends», «Creator», «Classic»

### 1.2.4 Entscheidungstabelle – Subdecision Erfahrung

Damit eine Empfehlung für ein Lego-Set wirklich passend ist, reicht es nicht aus, nur den Schwierigkeitsgrad des Sets zu bestimmen. Auch die Erfahrung des Nutzers muss berücksichtigt werden. Die Subdecision „Erfahrung“ hilft dabei, die Fähigkeiten des Bauenden besser einzuschätzen und so eine fundierte Empfehlung abzugeben. In Kombination mit der Subdecision „Schwierigkeitsgrad“ wird sichergestellt, dass das empfohlene Set weder unterfordert noch überfordert, sondern gut zum Erfahrungsniveau des Nutzers passt.

#### Inputs:

##### Bautechnik:

Die Auswahl eines Lego-Sets wird massgeblich durch die Bauweise beeinflusst, da diese direkt die Erfahrung des Bauenden widerspiegelt. Ein Kunde, der regelmässig Eigenkreationen erschafft, ist in der Regel sicherer und versierter im Umgang mit LEGO-Steinen und -Techniken, da er oder sie bereits ein gutes Verständnis für Struktur, Kreativität und Problemlösung entwickelt hat. Auswahl: «Anleitung», «Eigenkreation», «Modular».

Datentyp: Text

##### Anzahl gebaute Lego Sets:

Mit mehr gebauten Sets kommt mehr Erfahrung. Die Anzahl spiegelt wieder, wie vertraut jemand bereits mit verschiedenen Lego-Modellen und -Techniken ist.

Datentyp: Nummer: 0 – 1'000 gebaute Sets

	inputs		outputs	annotations
U	Bautechnik	Anzahl gebaute Legosets	Erfahrung	Description
	$t_{Bautechnik}$ "Anleitung", "Eigenkreation", "Modular", null	$t_{AnzahlgebautLegosets}$ [0..1000]	$t_{Erfahrung}$ "Anfänger", "Fortgeschritten", "Expert"	
1	"Anleitung", "Modular"	<=5	"Anfänger"	
2	"Anleitung"	(15..25]	"Fortgeschritten"	
3	"Eigenkreation"	<=5	"Fortgeschritten"	
4	"Anleitung", "Modular", "Eigenkreation"	(5..15]	"Fortgeschritten"	
5	"Modular", "Eigenkreation"	(15..25]	"Expert"	
6	"Anleitung", "Modular", "Eigenkreation"	>25	"Expert"	

Abbildung 3: Entscheidungstabelle – Subdecision Erfahrung

**Hit-Policy:** Unique

**Möglicher Output:** «Anfänger», «Fortgeschritten», «Expert»

### 1.2.5 Entscheidungstabelle – Subdecision Zielgruppe

Die Definition der Zielgruppe ist entscheidend, um Lego-Sets an das Alter und die Erfahrung des Kunden anzupassen. Die Auswahl eines Sets sollte die Fähigkeiten und Interessen der Zielgruppe berücksichtigen, um sicherzustellen, dass es sowohl herausfordernd als auch altersgerecht ist. So wird eine positive und motivierende Bau-Erfahrung gewährleistet. Damit aber auch zum Beispiel Kinder, welche sich schon viel mit Lego auseinandergesetzt haben, haben wir mit der Subdecision «Zielgruppe» noch einen zusätzlichen Abgrenzungsfaktor eingeführt, der das Alter und aber auch die Erfahrung berücksichtigt.

#### Inputs:

##### Alter des Kunden

Datentyp: Integer -> Nummer: 1.5 – 100 Jahre

**Erfahrung (Subdecision):** (Wurde bereits oben beschrieben)

	inputs		outputs	annotations
	Alter	Erfahrung	Zielgruppe	Description
U	$\tau_{\text{Alter}}$ [1.5..100]	$\tau_{\text{Erfahrung}}$ "Anfänger", "Fortgeschritten", "Expert"	$\tau_{\text{Zielgruppe}}$ "Kleinkinder", "Kinder - Einsteiger", "Kinder - Fortgeschritten", "Jugendliche - Einsteiger", "Jugendliche - Fortgeschritten", "Erwachsene - Einsteiger", "Erwachsene - Fortgeschritten", "Erwachsene - Experte"	
1	<4	"Anfänger", "Fortgeschritten", "Expert"	"Kleinkinder"	
2	[4..12]	"Anfänger"	"Kinder - Einsteiger"	
3	[4..12]	"Fortgeschritten", "Expert"	"Kinder - Fortgeschritten"	
4	(12..18)	"Anfänger"	"Jugendliche - Einsteiger"	
5	(12..18)	"Fortgeschritten", "Expert"	"Jugendliche - Fortgeschritten"	
6	>=18	"Anfänger"	"Erwachsene - Einsteiger"	
7	>=18	"Fortgeschritten"	"Erwachsene - Fortgeschritten"	
8	>=18	"Expert"	"Erwachsene - Experte"	

Abbildung 4: Entscheidungstabelle – Subdecision Zielgruppe

#### Hit-Policy: Unique

**Möglicher Output:** «Kleinkinder», «Kinder-Einsteiger», «Kinder-Fortgeschritten», «Jugendliche-Anfänger», «Jugendliche-Fortgeschritten», «Erwachsene-Einsteiger», «Erwachsene-Einsteiger», «Erwachsene-Fortgeschritten», «Erwachsene-Experte»

## 1.2.6 Entscheidungstabelle – Lego Set

Die Entscheidungstabelle «Lego-Set» stellt unsere definitive Entscheidung (also Empfehlung) dar. Die Entscheidung wird anhand von fünf Kriterien getroffen, wovon drei Inputs direkt aus einer Subdecision weitergeleitet werden:

- Preisbereich
- Kompatibilität mit anderen Sets
- Schwierigkeitsgrad (Input aus Subdecision)
- Themenwelt (Input aus Subdecision)
- Zielgruppe (Input aus Subdecision)

### Inputs:

#### Preisbereich:

Bestimmt die Preiskategorie in der der Kunde einkaufen möchte.

Datentyp: Nummer zwischen 20.00 - 1'000.00 CHF

#### Soll kompatibel sein mit einem weiteren Set:

Bestimmt ob ein Set mit anderen Sets kompatibel ist, beispielsweise für einen Kunden der/die eine Modulare Bauweise präferiert.

Datentyp: Binäre Aussage true/false

Inputs						Outputs
	Zielgruppe	Themenwelt	Preisbereich	Soll kompatibel sein mit weiteren Sets	Schwierigkeitsgrad	Lego Set
C	<i>t_Zielgruppe</i> "Kleinkinder", "Kinder - Einsteiger", "Kinder - Fortgeschritten", "Jugendliche - Einsteiger", "Jugendliche - Fortgeschritten", "Erwachsene - Einsteiger", "Erwachsene - Fortgeschritten", "Erwachsene - Experte"	<i>t_Themenwelt</i> "Disney", "Marvel", "Harry Potter", "Ninjago", "Star Wars", "Minecraft", "City", "Technik", "Architektur", "Friends", "Creator", "Classic"	<i>t_Preisbereich</i> [20..1000]	<i>Boolean</i>	<i>t_Schwierigkeitsgrad</i> "Einfach", "Mittel", "Schwer"	<i>t_LegoSet</i> "Kreativer Reisekoffer", "Hogwarts Express" & der Bahnhof von Hogsmeade, "Imperial Sternenjäger", "Donut Truck", "Abenteuer - Wohnmobil", "Spinjitzu-Tempel der Ninja", "Hinterhalt auf Mandalore™ Battle Pack", "Ducati Panigale V4 S Motorrad", "Ferrari Daytona SP3", "NASA Artemis Startrampe", "Grosse interaktive Eisenbahn", "3-in-1-Zauberschloss", "Paisleys Haus", "Bunte Bouzouki-Bau", "Der Sprechende Hut", "Minecraft Mini-Höhle", "Mein erster Bauernhof", "Die Milchstrassen-Galaxie", "Titanic", "Schloss Hogwarts mit Schlossgelände", "McLaren P1", "Schloss Hogwarts", "Invisible Hand", "Die Werkbank", "Disney Hocus Pocus: Das Hexenhaus der Sonderschwwestern", "Trenn-Brunnen", "Darth Vader Helm", "Ninjago City Werkstätten", "MARVEL Lego & Minifiguren", "New York City", "London", "Schloss Hogwarts Die Große Halle", "82-00", "Fast and Furious Toyota Supra MK4", "Wilde Tiere: Rosa Flamingo", "Wilde Tiere: Pandafamilie", "Ultimatives Abenteuererschloss", "Geschäft für Hausarztzubehör", "Sindor", "das Löwenjunge des Königs", "X-Men: X-Men", "Avengers Tower", "New York City", "Obi-Wan Kenobi Jedi Starfighter™", "Darth Mauls Sith Infiltrator™", "Spinjitzu-Tempel der Ninja", "Lloyds und Anns Training-Mechs"
1	"Kleinkinder"	"City"	[20..50]	true	"Einfach"	"Abenteuer - Wohnmobil"
2	"Kleinkinder"	"Classic"	[20..50]	true	"Einfach"	"Mein erster Bauernhof"
3	"Kleinkinder"	"Disney"	[20..50]	false	"Einfach"	"3-in-1-Zauberschloss"
4	"Kleinkinder"	-	[51..200]	true	"Einfach"	"Grosse interaktive Eisenbahn"
5	"Kleinkinder"	"Classic"	[51..200]	true	"Einfach"	"Grosse interaktive Eisenbahn"

Abbildung 5: Entscheidungstabelle – Lego Set

### Hit-Policy: Collect

### Mögliche Outputs: Spezifisches Lego-Set

## 1.2.7 Test Cases

Um unser Entscheidungsmodell immer wieder überprüfen und nachvollziehen zu können, haben wir insgesamt neun Use-Cases definiert. Folglich zeigen wir zwei dieser Use-Cases als Beispiel. Alle neun Test-Cases können in der DMN Datei angeschaut werden.

### Test Case Beispiele:

Test Cases

Service

Diagram Page 1

▶ ✓ Test case 1 - Ben ▶ ✎ ✕

Inputs

Alter

3

Preisbereich

30

Interessen

Simple Spielen

Anzahl gebaute Legosets

0

Soll Lizenz haben

false

Soll kompatibel sein mit weiteren Sets

true

Bauzeit in h

1

Bauteilmenge

50

Bautechnik

Anleitung

Komplexe Bausteine

false

Outputs

	Expected	Actual
Lego Set	Mein erster Bauernhof	Mein erster Bauernhof

Test Cases

Service

Diagram Page 1

▶ ✓ Test case 1 - Ben ▶ ✎ ✕

▶ ✓ Test case 2 - Finn ▶ ✎ ✕

Inputs

Alter

6

Preisbereich

40

Interessen

Ninjas

Anzahl gebaute Legosets

1

Soll Lizenz haben

true

Soll kompatibel sein mit weiteren Sets

true

Bauzeit in h

1

Bauteilmenge

150

Bautechnik

Anleitung

Komplexe Bausteine

false

Outputs

	Expected	Actual
Lego Set	Spinjitzu-Tempel der Ninja	Spinjitzu-Tempel der Ninja

## 2 Aufgabe 2

In den folgenden Abschnitten werden die konkreten Aufgabenstellungen unseres Projekts aufgeführt und die Ergebnisse dokumentiert.

### 2.1 Aufgabenstellung

« Erstellen Sie für ihr Beispiel eine Kaufberatung in Prolog.

Überlegen Sie zuerst, welche Fragen eine Kundin oder ein Kunde stellen könnte.

Überlegen Sie, welche Informationen über die Produkte, die Kunden bzw. den Bedarf in der Wissensbasis repräsentiert werden müssen, sowie geeignete Prädikate, um die Informationen darzustellen. Welche Regeln sind notwendig, um die Fragen zu beantworten? Formulieren Sie entsprechende Anfragen.

- Identifikation der häufigsten Kundenfragen.
- Sammlung der benötigten Informationen über Produkte und Kundenbedürfnisse.
- Implementierung der Wissensbasis und Regeln in Prolog.»

### 2.2 Mögliche Kundenanfragen

Wir haben hier verschiedene Subdecisions durchleuchtet, welche bei den häufigsten Kundenanfragen vorkommen könnten.

#### 2.2.1 Geschenk-Szenarien

1. **„Ich suche ein geeignetes Geschenk für ein Kind im Alter von 5 Jahren.“**  
→ Fokus: einfache Sets, spielerische Themen, niedriger Preis, kindgerechte Bauweise.
2. **„Was kann ich einem 12-jährigen Jungen schenken, der LEGO liebt und gerne baut?“**  
→ Fokus: mittlerer Schwierigkeitsgrad, kompatible Erweiterungen, passende Themenwelt wie Ninjago, Minecraft etc.



### **2.2.2 Erfahrungs- & Kompetenz-basiert**

3. „Ich habe schon viele Sets gebaut. Was ist eine neue Herausforderung für mich?“  
→ Fokus: Erwachsene – Fortgeschrittene/Experten, hohe Bauteilmenge, komplexe Bausteine, lange Bauzeit.
4. „Ich habe noch nie ein LEGO-Set gebaut. Welches ist für Anfänger geeignet?“  
→ Fokus: einfache Anleitung, Einstiegsthema (Classic, City, Friends), kleiner Preisbereich.

### **2.2.3 Themenwelt-Interessierte**

5. „Ich liebe Star Wars – was ist das beste Set dazu?“  
→ Fokus: Themenwelt Star Wars, Lizenz = true, eventuell auch Kompatibilität.
6. „Meine Tochter mag Disney-Filme, welches Set passt zu ihr?“  
→ Themenwelt Disney, altersgerecht.

### **2.2.4 Nutzerzentriert (individuelle Präferenzen)**

7. „Ich möchte ein Set, das zu anderen passt, die ich schon habe.“  
→ Kompatibilität = true, ggf. gleiche Themenwelt.
8. „Welches Set kann man zusammen mit einem Kind bauen?“  
→ mittlerer Schwierigkeitsgrad, kindgerechte Themenwelt, moderate Bauzeit.

### **2.2.5 Budget-orientiert**

10. „Was bekomme ich für unter 50 Franken?“  
→ Preisbereich 20–50 CHF, ideal für Einsteiger oder Kinder.
11. „Ich will ein grosses Technik-Set für max. 300 CHF, was empfehlst du?“  
→ Zielgruppe Erwachsene, Themenwelt Technik, Preisbereich 100–300 CHF, hoher Schwierigkeitsgrad.

### **2.2.6 Sammlung Produkt und Kundenbedürfnisse**

Für die Umsetzung einer regelbasierten Kaufberatung in Prolog mussten sowohl produktbezogene Merkmale als auch kundenseitige Anforderungen systematisch erfasst werden. Die folgende Struktur bildet die Grundlage für die Wissensbasis und dient der logischen Ableitung passender Produktempfehlungen.

### **2.2.7 Produktbezogene Informationen (Eigenschaften der LEGO-Sets)**

Um eine präzise und differenzierte Produktauswahl zu ermöglichen, wurden folgende Merkmale in der Wissensbasis für jedes LEGO-Set modelliert:

- **Set-Name:** Eindeutige Bezeichnung des Produkts.
- **Zielgruppe:** Alters- und erfahrungsbasierte Einteilung der Nutzergruppen (z. B. „Kinder – Einsteiger“, „Erwachsene – Experte“).
- **Themenwelt:** Zugehörige Kategorie wie „Technik“, „Star Wars“, „Friends“ etc.
- **Preisbereich:** Typischer Marktpreis (z. B. 20–50 CHF, 100–500 CHF).
- **Schwierigkeitsgrad:** Eingestuft in „Einfach“, „Mittel“ oder „Schwer“, basierend auf Bauteilmenge, Bauzeit und Komplexität.
- **Bauteilmenge:** Anzahl der enthaltenen Teile.
- **Bauzeit:** Geschätzte Aufbauzeit in Stunden.
- **Komplexe Bausteine:** Enthält technische oder motorisierte Bauelemente (true/false).
- **Lizenzprodukt:** Gibt an, ob es sich um ein lizenziertes Set handelt (z. B. Disney, Marvel).
- **Kompatibilität:** Ob das Set zu anderen Erweiterungen passt (true/false).

### **2.2.8 Kundenbezogene Informationen (Bedarf und Präferenzen)**

Die Anforderungen und Wünsche der Kundinnen und Kunden wurden ebenfalls als strukturierte Eingaben definiert. Diese bilden die Grundlage für die Ableitung geeigneter LEGO-Produkte:

- **Alter:** Alter der Zielperson, beeinflusst die Zielgruppenzuordnung.
- **Anzahl gebaute Sets:** Basis für die Einschätzung der Bauerfahrung.
- **Bautechnik:** Gibt an, mit welchen Bauweisen gearbeitet wird.
- **Erfahrung:** Wird aus Anzahl gebauter Sets und der angewandten Bautechnik automatisch abgeleitet.
- **Interessen:** Thematische Vorlieben wie „Fahrzeuge“, „Architektur“, „Fantasy“ etc.
- **Wunsch nach Lizenz:** Gibt an, ob ein lizenziertes Produkt bevorzugt wird.
- **Kompatibilität:** Ob das neue Set zu weiteren Sets passen soll.
- **Preisvorstellung:** Budgetrahmen für den Kauf.
- **Gewünschte Bauzeit:** Wie lange der Bauprozess dauern darf.

## 2.3 Implementierung der Wissensbasis und Regeln in Prolog

### 2.3.1 Faktenbasis

Unsere Wissensbasis bildet verschiedene Eigenschaften der LEGO-Produkte und der Kundenbedürfnisse ab. Die wichtigsten Fakten sind:

**Alter:** Das Alter der Zielperson wird als Zahl zwischen 1.5 und 100 Jahren modelliert.

```
% t_Alter: 1.5 bis 100 Jahre  
alter(Alter) :- Alter >= 1.5, Alter <= 100.
```

**Preisbereich:** Preisangaben sind auf 20 bis 1000 CHF beschränkt.

```
% t_Preisbereich: 20 bis 1000  
preis(Preis) :- Preis >= 20, Preis <= 1000.
```

**Kompatibilität:** Ein LEGO-Set kann entweder kompatibel oder nicht kompatibel mit anderen Sets sein.

```
% Soll kompatibel sein mit weiteren Sets: true/false  
kompatibel(true).  
kompatibel(fasle).
```

**Bautechnik:** Die Bauweise unterscheidet sich in 'Anleitung', 'Eigenkreation' und 'Modular'.

```
% t_Bautechnik  
bautechnik('Anleitung').  
bautechnik('Eigenkreation').  
bautechnik('Modular').
```

**Anzahl gebaute Sets:** Die bisherige Bauerfahrung wird über die Zahl der gebauten Sets abgebildet (0–1000).

```
% t_AnzahlgebauteLegosets: 0 bis 1000  
anzahl_gebaute_legosets(Anzahl) :- Anzahl >= 0, Anzahl <= 1000.
```

**Bauteilmenge und Bauzeit:** Diese beschreiben die Komplexität eines Sets.

```
% t_Bauteilmenge: 0 bis 12000  
bauteilmenge(Menge) :- Menge >= 0, Menge <= 12000.
```

**Komplexe Bausteine:** Gibt an, ob ein Set schwierige Spezialteile enthält.

```
% Komplexe Bausteine: true/false  
komplexe_bausteine(true).  
komplexe_bausteine(false).
```

**Lizenzwunsch:** Der Kunde kann an lizenzierten Themen interessiert sein (z. B. Disney, Marvel).

```
% Soll Lizenz haben: true/false  
soll_lizenz_haben(true).  
soll_lizenz_haben(false).
```

**Zielgruppen, Erfahrungen, Schwierigkeitsgrade, Themenwelten, LEGO-Sets und Interessen:**  
Diese Eigenschaften werden als Fakten hinterlegt.

**1. Set Zielgruppe:** Zu welcher Zielgruppe das Set jeweils gehört.

```
% Lego Set Zielgruppen  
set_zielgruppe('Abenteuer - Wohnmobil', 'Kleinkinder').  
set_zielgruppe('Mein erster Bauernhof', 'Kleinkinder').
```

**2. Set Themenwelt:** Zu welcher Themenwelt das Set jeweils gehört.

```
%Lego Set Themenwelt  
set_themenwelt('Abenteuer - Wohnmobil', 'City').  
set_themenwelt('Mein erster Bauernhof', 'Classic').
```

**3. Set Preisbereich:** Zu welchem Preisbereich das Set jeweils gehört.

```
% Lego Set Preisbereich  
set_preisbereich('Abenteuer - Wohnmobil', preisbereich(20,50)).  
set_preisbereich('Mein erster Bauernhof', preisbereich(20,50)).
```

**4. Set Kompatibel:** Zu welcher Kompatibilität das Set jeweils gehört.

```
% Lego Set Kompatibel  
set_kompatibel('Abenteuer - Wohnmobil', true).  
set_kompatibel('Mein erster Bauernhof', true).
```

**5. Set Schwierigkeitsgrad:** Zu welchem Schwierigkeitsgrad das Set jeweils gehört.

```
% Lego Set Schwierigkeitsgrad  
set_schwierigkeit('Abenteuer - Wohnmobil', 'Einfach').  
set_schwierigkeit('Mein erster Bauernhof', 'Einfach').
```

### 2.3.2 Regel zur Bestimmung der Zielgruppe

#### Ziel der Regel:

Basierend auf dem Alter und der Bauerfahrung einer Person wird bestimmt, welcher Zielgruppe sie zugeordnet wird (z. B. Kleinkinder, Kinder-Einsteiger, Erwachsene-Experte).

#### Prolog-Regel:

```
% Zielgruppe
zielgruppe_empfehlung(Alter, Erfahrung, 'Kleinkinder') :-
    Alter < 4,
    (Erfahrung = 'Anfänger'; Erfahrung = 'Fortgeschritten'; Erfahrung = 'Expert').
zielgruppe_empfehlung(Alter, 'Anfänger', 'Kinder - Einsteiger') :-
    Alter >= 4, Alter <= 12.
zielgruppe_empfehlung(Alter, Erfahrung, 'Kinder - Fortgeschritten') :-
    Alter >= 4, Alter <= 12,
    (Erfahrung = 'Fortgeschritten'; Erfahrung = 'Expert').
zielgruppe_empfehlung(Alter, 'Anfänger', 'Jugendliche - Einsteiger') :-
    Alter > 12, Alter <= 18.
zielgruppe_empfehlung(Alter, Erfahrung, 'Jugendliche - Fortgeschritten') :-
    Alter > 12, Alter <= 18,
    (Erfahrung = 'Fortgeschritten'; Erfahrung = 'Expert').
zielgruppe_empfehlung(Alter, 'Anfänger', 'Erwachsene - Einsteiger') :-
    Alter >= 18.
zielgruppe_empfehlung(Alter, 'Fortgeschritten', 'Erwachsene - Fortgeschritten') :-
    Alter >= 18.
zielgruppe_empfehlung(Alter, 'Expert', 'Erwachsene - Experte') :-
    Alter >= 18.

empfehle_zielgruppe(Alter, Erfahrung, Zielgruppe) :-
    zielgruppe_empfehlung(Alter, Erfahrung, Zielgruppe).
```

**zielgruppe\_empfehlung(Alter, Erfahrung, Zielgruppe):** Wendet die Entscheidungsregel an, die anhand von Alter und Erfahrung die Zielgruppe bestimmt (z. B. 'Kleinkinder', 'Jugendliche - Fortgeschritten', 'Erwachsene - Experte').

Diese Regel ruft die eigentliche Entscheidungslogik (zielgruppe\_empfehlung) auf.

### 2.3.3 Regel zur Bestimmung der Erfahrung

#### Ziel der Regel:

Basierend auf der Anzahl der gebauten Sets und der verwendeten Bautechniken wird die Bauerfahrung einer Person eingeschätzt. Die Erfahrungsstufe kann "Anfänger", "Fortgeschritten" oder "Expert" sein.

```
erfahrung_empfehlung(Bautechniken, AnzahlGebaut, 'Anfänger') :-  
    AnzahlGebaut <= 5,  
    (member('Anleitung', Bautechniken); member('Modular', Bautechniken)).  
erfahrung_empfehlung(Bautechniken, AnzahlGebaut, 'Fortgeschritten') :-  
    AnzahlGebaut > 15, AnzahlGebaut <= 25,  
    (member('Anleitung', Bautechniken); member('Eigenkreation', Bautechniken)).  
erfahrung_empfehlung(Bautechniken, AnzahlGebaut, 'Fortgeschritten') :-  
    AnzahlGebaut <= 5,  
    member('Eigenkreation', Bautechniken).  
erfahrung_empfehlung(Bautechniken, AnzahlGebaut, 'Fortgeschritten') :-  
    AnzahlGebaut > 5, AnzahlGebaut <= 15,  
    (member('Anleitung', Bautechniken);  
    member('Modular', Bautechniken);  
    member('Eigenkreation', Bautechniken)).  
erfahrung_empfehlung(Bautechniken, AnzahlGebaut, 'Expert') :-  
    AnzahlGebaut > 15, AnzahlGebaut <= 25,  
    (member('Modular', Bautechniken); member('Eigenkreation', Bautechniken)).  
erfahrung_empfehlung(Bautechniken, AnzahlGebaut, 'Expert') :-  
    AnzahlGebaut > 25,  
    member('Anleitung', Bautechniken),  
    member('Modular', Bautechniken),  
    member('Eigenkreation', Bautechniken).  
  
empfehle_erfahrung(Bautechniken, AnzahlGebaut, Erfahrung) :-  
    erfahrung_empfehlung(Bautechniken, AnzahlGebaut, Erfahrung).
```

Die Regel `empfehle_erfahrung(Bautechniken, AnzahlGebaut, Erfahrung)` verknüpft die Eingaben des Benutzers mit der Entscheidungslogik zur Einstufung der Bauerfahrung. Sie prüft, ob die Bauanzahl gültig ist und wendet darauf die passenden Regeln an.

1. **erfahrung\_empfehlung(Bautechniken, AnzahlGebaut, Erfahrung):**
2. Wendet die Regeln zur Bestimmung der Bauerfahrung an, unter Berücksichtigung der verwendeten Bauweise:
  - a. Wenige Sets + Anleitung = Anfänger
  - b. Viele Sets + Eigenkreationen = Fortgeschritten oder Expert
  - c. Besonders viele Sets mit allen Techniken = Expert

### 2.3.4 Regel zur Bestimmung des Schwierigkeitsgrads

**Ziel der Regel:** Ermittelt den Schwierigkeitsgrad eines LEGO-Sets anhand der Anzahl der Bauteile, der geschätzten Bauzeit und der Komplexität der Bausteine.

```
schwierigkeitsgrad(Bauteilmenge, BauzeitStd, false, 'Einfach') :- Bauteilmenge=<1500, BauzeitStd=<2.  
schwierigkeitsgrad(Bauteilmenge, BauzeitStd, false, 'Mittel') :- Bauteilmenge=<500, BauzeitStd>2.  
schwierigkeitsgrad(Bauteilmenge, BauzeitStd, true, 'Mittel') :- Bauteilmenge=<1500, BauzeitStd>=1,  
BauzeitStd=<5.  
schwierigkeitsgrad(Bauteilmenge, BauzeitStd, false, 'Mittel') :- Bauteilmenge>=500, BauzeitStd>=2, Bau-  
zeitStd=<5.  
schwierigkeitsgrad(Bauteilmenge, BauzeitStd, true, 'Mittel') :- Bauteilmenge>1500, BauzeitStd>=1, Bau-  
zeitStd=<5.  
schwierigkeitsgrad(Bauteilmenge, BauzeitStd, false, 'Schwer') :- Bauteilmenge >= 500, BauzeitStd > 5.  
schwierigkeitsgrad(_, BauzeitStd, true, 'Schwer') :- BauzeitStd>5.
```

Die Regel `schwierigkeitsgrad` bestimmt den Schwierigkeitsgrad eines LEGO-Sets anhand der Anzahl der Bauteile, der Bauzeit in Stunden und der Komplexität der Bausteine – je nach Kombination dieser Kriterien wird der Schwierigkeitsgrad als 'Einfach', 'Mittel' oder 'Schwer' klassifiziert.

### 2.3.5 Regel zur Bestimmung der Themenwelt

#### Ziel der Regel:

Bestimmt, welche LEGO-Themenwelt zu den Interessen des Nutzers passt abhängig davon, ob dieser Lizenzprodukte wünscht oder nicht.

```
themenwelt_empfehlung('Fahrzeuge', false, 'Technik').
themenwelt_empfehlung('Fahrzeuge', true, 'Star Wars').
themenwelt_empfehlung('Disney', false, 'Friends').
themenwelt_empfehlung('Disney', true, 'Disney').
themenwelt_empfehlung('Superheroes', false, 'Ninjabo').
themenwelt_empfehlung('Superheroes', true, 'Marvel').
themenwelt_empfehlung('Architektur', false, 'Architektur').
themenwelt_empfehlung('Architektur', true, 'Harry Potter').
themenwelt_empfehlung('Sci-Fi', false, 'Ninjabo').
themenwelt_empfehlung('Sci-Fi', true, 'Star Wars').
themenwelt_empfehlung('Kinderserie / Film', false, 'Ninjabo').
themenwelt_empfehlung('Kinderserie / Film', true, 'Disney').
themenwelt_empfehlung('Ninjas', _, 'Ninjabo').
themenwelt_empfehlung('Gaming', false, 'Creator').
themenwelt_empfehlung('Gaming', true, 'Minecraft').
themenwelt_empfehlung('Sport', _, 'City').
themenwelt_empfehlung('Tier/Natur', false, 'Friends').
themenwelt_empfehlung('Tier/Natur', true, 'Minecraft').

empfehle_themenwelt(Interesse, Lizenz, Empfohlen) :-
    themenwelt_empfehlung(Interesse, Lizenz, Empfohlen),
```

Die Regel `empfehle_themenwelt(Interesse, Lizenz, Themenwelt)` verwendet die vom Nutzer angegebenen Interessen (z. B. Gaming, Fahrzeuge, Superhelden) und prüft, ob eine Lizenz gewünscht ist, um eine passende Themenwelt zu empfehlen.

- `themenwelt_empfehlung(Interesse, Lizenz, Themenwelt)`: Führt die eigentliche Ableitung durch und bestimmt eine geeignete Themenwelt (z. B. Technik, Marvel, Friends).



### 2.3.6 Regel zur Bestimmung des Preisbereichs

#### Ziel der Regel:

Ordnet einen konkreten Preiswert (z. B. 75 CHF) in einen vordefinierten Preisbereich ein (z. B. 20–50 CHF, 51–200 CHF etc.), damit das System weiss, welche Sets innerhalb des gewünschten Budgets liegen.

#### Prolog-Regel:

```
preis_im_bereich(Preis, preisbereich(20,50)) :- Preis >= 20, Preis <= 50.  
preis_im_bereich(Preis, preisbereich(51,200)) :- Preis >= 51, Preis <= 200.  
preis_im_bereich(Preis, preisbereich(201,500)) :- Preis >= 201, Preis <= 500.  
preis_im_bereich(Preis, preisbereich(501,1000)):- Preis >= 501, Preis <= 1000
```

Die Regel `preis_im_bereich(Preis, Preisbereich)` überprüft, in welchen Bereich ein gegebener Preis fällt. Das wird benötigt, um Sets mit passenden Preisetiketten herauszufiltern.

- Preis: Der vom Benutzer angegebene Preis oder das Budget (z. B. 120 CHF).
- Preisbereich: Das daraus abgeleitete Label für den Bereich (z. B. `preisbereich(51,200)`).
- Die Regel wird verwendet, um die Auswahl in der Hauptregel (`empfehle_lego_set`) einzugrenzen.

### 2.3.7 Regel zur Auswahl eines Kunden und deren Fakten

```
kunde_praefenzen(anna, 8, 45, true, 'Anleitung', 2, 200, 1, false, true, 'Gaming').
kunde_praefenzen(ben, 3, 30, true, 'Anleitung', 0, 50, 1, false, false, 'Simples Spielen').
kunde_praefenzen(clara, 10, 150, true, 'Anleitung', 6, 500, 3, false, true, 'Disney').
kunde_praefenzen(david, 18, 80, true, 'Anleitung', 2, 800, 4, true, true, 'Sci-Fi').
kunde_praefenzen(finn, 6, 40, true, 'Anleitung', 1, 150, 1, false, true, 'Ninjas').
kunde_praefenzen(kerem, 12, 250, true, 'Eigenkreation', 8, 600, 5, true, true, 'Fantasy').
kunde_praefenzen(bastian, 40, 400, false, 'Modular', 10, 3000, 15, true, false, 'Fahrzeuge').
kunde_praefenzen(fabia, 7, 60, true, 'Anleitung', 3, 250, 2, false, false, 'Tier/Natur').
kunde_praefenzen(elena, 30, 400, true, 'Anleitung', 20, 5000, 20, true, true, 'Superheroes').
empfehle_set_fuer_kunde(Kunde, EmpfohlenesSet) :-
    kunde_praefenzen(Kunde, Alter, Preis_Kunde, Kompatibel, Bautechnik, AnzSets, Teile, Zeit,
Komplex, Lizenz, Interesse),
    empfehle_lego_set(Alter, Preis_Kunde, Kompatibel, Bautechnik, AnzSets, Teile, Zeit, Komplex, Li-
zenz, Interesse, EmpfohlenesSet).
```

Diese Regel sowie Fakten werden gebraucht, um ausfindig zu machen, welcher Kunde welche Präferenzen hat, um ein schlussendliches Lego Set für den jeweiligen Kunden vorschlagen zu können. Dafür muss **empfehle\_set\_fuer\_kunde** mit einem bestehenden Kunden aufgerufen werden. Die Anfrage dafür sieht wie folgt aus:

?- empfehle\_set\_fuer\_kunde(<bestehender Kunde>, Ausabe\_Set).

### 2.3.8 Fakten Check mit eventueller Liste

```
% Prüft, ob die gesuchte Zielgruppe im Fakt enthalten ist (Atom oder Liste)
set_zielgruppe_match(Set, GesuchteZielgruppe) :-
    set_zielgruppe(Set, ZielgruppeFakt),
    ( is_list(ZielgruppeFakt) -> member(GesuchteZielgruppe, ZielgruppeFakt)
    ; ZielgruppeFakt == GesuchteZielgruppe
    ).

% Prüft, ob der gesuchte Schwierigkeitsgrad im Fakt enthalten ist (Atom oder Liste)
set_schwierigkeit_match(Set, GesuchterSchwierigkeitsgrad) :-
    set_schwierigkeit(Set, SchwierigkeitFakt),
    ( is_list(SchwierigkeitFakt) -> member(GesuchterSchwierigkeitsgrad, SchwierigkeitFakt)
    ; SchwierigkeitFakt == GesuchterSchwierigkeitsgrad
    ).
```

Diese zwei Regeln werden bei der Lego Set Evaluation benötigt, um die Zielgruppen und die Schwierigkeitsgrade zu checken mit eventuell mehreren Ergebnissen. So kann ein Ergebnis oder mehrere zurückgegeben werden.

### 2.3.9 Regel Gesamtempfehlung eines LEGO-Sets von Kundenempfehlung

#### Ziel der Regel:

Diese Regel bringt alle vorhergehenden Entscheidungen zusammen. Sie bildet das „Hauptprogramm“ unseres Empfehlungssystems, indem sie Eingaben verarbeitet, Zwischenentscheidungen trifft und letztlich ein passendes LEGO-Set vorschlägt.

#### Prolog-Regel:

```
empfehle_lego_set(
    Alter, Preis_Eingabe, Kompatibel_Eingabe, Bautechnik, Anzahl_Legosets, BauteilMenge, Bauzeit,
    KomplexeBausteine, Lizenz, Interesse, Ausgabe_Set ) :-

    alter(Alter),
    kompatibel(Kompatibel_Eingabe),
    bautechnik(Bautechnik),
    anzahl_gebaute_legosets(Anzahl_Legosets),
    bauteilmenge(BauteilMenge),
    bauzeit(Bauzeit),

    komplexe_bausteine(KomplexeBausteine),
    soll_lizenz_haben(Lizenz),
    interessen(Interesse),
```

```
% Subdecissions ableiten
empfehle_themenwelt(Interesse, Lizenz, Themenwelt),
    schwierigkeitsgrad(BauteilMenge, Bauzeit, KomplexeBausteine, Schwierigkeitsgrad),
empfehle_erfahrung([Bautechnik], Anzahl_Legosets, Erfahrung),
empfehle_zielgruppe(Alter, Erfahrung, Zielgruppe),

erfahrung(Erfahrung),
schwierigkeitsgrad(Schwierigkeitsgrad),
zielgruppe(Zielgruppe),
themenwelt(Themenwelt),

preis_im_bereich(Preis_Eingabe, Preisbereich),

% Set suchen, das ALLEN abgeleiteten Kriterien entspricht
lego_set(Ausgabe_Set),
set_zielgruppe_match(Ausgabe_Set, Zielgruppe),
set_themenwelt(Ausgabe_Set, Themenwelt),
set_preisbereich(Ausgabe_Set, Preisbereich),
set_kompatibel(Ausgabe_Set, Kompatibel_Eingabe),
set_schwierigkeit_match(Ausgabe_Set, Schwierigkeitsgrad).
```

Die Regel `empfehle_lego_set(...)` nutzt die vollständige Benutzerabfrage (Alter, Bauweise, Interessen etc.) und durchläuft folgende Schritte:

- Gültigkeitsprüfung der Eingaben (alter, bautechnik, preis, etc.)
- Ableitung der **Erfahrung** über `empfehle_erfahrung(...)`
- Bestimmung der **Zielgruppe** über `empfehle_zielgruppe(...)`
- Zuordnung der **Themenwelt** über `empfehle_themenwelt(...)`
- Bewertung des **Schwierigkeitsgrads** über `schwierigkeitsgrad(...)`
- Gültigkeitsprüfung der Ergebnisse der Ableitungen.
- Preisbereichs-Evaluation.
- Auswahl des passenden Sets mithilfe der Set-Fakten.

Am Ende liefert sie mit `Set` die passende Empfehlung als Ergebnis der gesamten Entscheidungslogik.

### 3 Aufgabe 3

In den folgenden Abschnitten werden die konkreten Aufgabenstellungen unseres Projekts aufgeführt und die Ergebnisse dokumentiert.

#### 3.1 Aufgabenstellung

«Verwandeln Sie die Lösung von Aufgabe 2 in einen Wissensgraphen. Definieren Sie ein Schema für die Objekte, die für die Kaufberatung relevant sind, insb. über die Produkte und evtl. Nutzungen sowie deren Eigenschaften.

Für Empfehlungen erstellen Sie Regeln in SWRL und Abfragen in SQWRL oder SPARQL.»

#### 3.2 Ontologieschema

Das Ontologieschema wurde auf Grundlage der in Aufgabe 2 identifizierten Konzepte und Entscheidungsfaktoren entwickelt. Es unterscheidet zwischen Klassen, Objekt- und Dateneigenschaften:

##### 3.2.1 Zentrale Klassen:

- **Kunde:** Repräsentiert eine Person, für die eine LEGO-Empfehlung vorgeschlagen werden soll.
- **LEGO-Set:** Modelliert ein konkretes Set aus dem LEGO-Produktsortiment.
- **Themenwelt, Zielgruppe, Erfahrung, Interesse, Bautechnik, Schwierigkeitsgrad:** Dienen der strukturierten Beschreibung von Sets und Kundenprofilen. Die entsprechenden Instanzen der Klassen wurden aus den in Prolog definierten Fakten in Protégé übernommen.



Abbildung 6: Zentrale Klassen im Protégé

### 3.2.2 *Objekteigenschaften (Domains → Ranges):*

- empfiehltSetFuerKunde (Kunde → LEGOSet)
- gehoertZuThemenwelt (LEGOSet → Themenwelt)
- hatAbgeleiteteZielgruppe (Kunde → Zielgruppe)
- hatBevorzugteBautechnik (Kunde → Bautechnik)
- hatBevorzugteThemenwelt (Kunde → Themenwelt)
- hatErfahrung (Kunde → Erfahrung)
- hatGewuenschtenSchwierigkeitsgrad (Kunde → Schwierigkeitsgrad)
- hatKundenInteresse (Kunde → Interesse)
- hatSetEigeneZielgruppe (LEGOSet → Zielgruppe)
- hatSetSchwierigkeitsgrad (LEGOSet → Schwierigkeitsgrad)

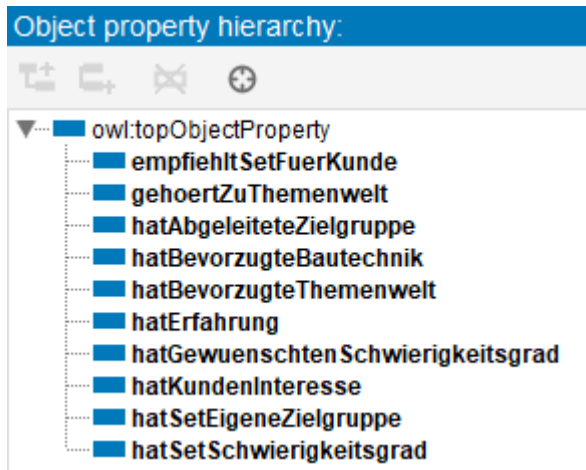


Abbildung 7: Objekteigenschaften im Protégé

### 3.2.3 Dateneigenschaften (Domains $\rightarrow$ Ranges):

- hatAlter (Kunde  $\rightarrow$  xsd:integer)
- hatAnzahlGebauterSets (Kunde  $\rightarrow$  xsd:integer)
- hatPreisMax (LEGOSet  $\rightarrow$  xsd:integer)
- hatPreisMin (LEGOSet  $\rightarrow$  xsd:integer)
- hatWunschPreis (Kunde  $\rightarrow$  xsd:integer)
- istKompatibel (LEGOSet  $\rightarrow$  xsd:boolean)
- praefertiertBauteilmenge (Kunde  $\rightarrow$  xsd:integer)
- praefertiertBauzeit (Kunde  $\rightarrow$  xsd:integer)
- praefertiertKomplexeBausteine (Kunde  $\rightarrow$  xsd:boolean)
- wuenschtKompatibel (Kunde  $\rightarrow$  xsd:boolean)
- benoetigtLizenz (LEGOSet  $\rightarrow$  xsd:boolean)

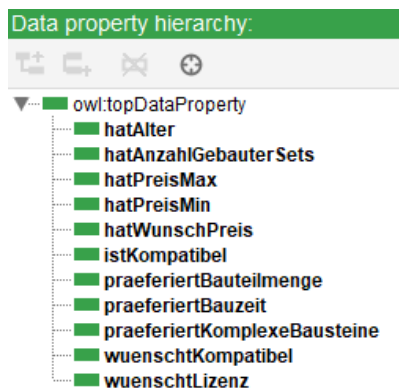


Abbildung 8: Dateneigenschaften im Protégé

Dieses semantische Modell ermöglicht eine präzise Abbildung der Relationen zwischen Kundenanforderungen und Produktmerkmalen.

### 3.2.4 Klassenhierarchie

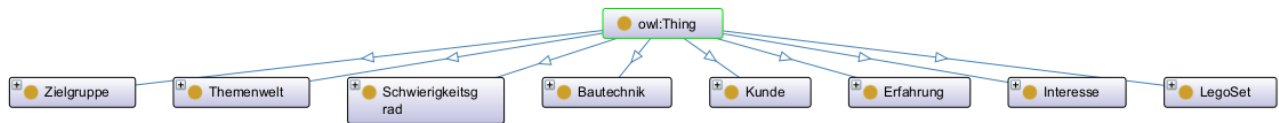


Abbildung 9: Klassenhierarchie im Protégé

In unserem Projekt haben wir uns bewusst dafür entschieden, dass alle Klassen wie Kunde, Zielgruppe, LegoSet, Themenwelt etc. direkt unter owl:Thing stehen. Das bedeutet: Wir haben keine Unterklassen wie z. B. Kinder\_Einsteiger in der Zielgruppe erstellt.

Der Hauptgrund dafür ist, dass in OWL Daten-Eigenschaften (z. B. hatAlter, hatWunschPreis) nicht automatisch an Unterklassen vererbt werden. Wenn wir also mit Unterklassen arbeiten würden, müssten wir alle Eigenschaften manuell mehrfach zuweisen was aufwändig und fehleranfällig wäre.

Ausserdem verwenden wir viele SWRL-Regeln und SPARQL-Abfragen, die direkt mit den Daten-Eigenschaften arbeiten. Eine flache Struktur macht diese Regeln und Abfragen einfacher und übersichtlicher, weil wir keine Vererbungslogik oder Klassenzugehörigkeiten prüfen müssen.

Auch für die Wartung und Erweiterung unserer Ontologie ist die flache Struktur von Vorteil: Wir behalten den Überblick und können leichter neue Klassen, Regeln oder Daten hinzufügen, ohne dass sich dies auf andere Bereiche auswirkt.

Insgesamt ist das flache Modell für unseren Anwendungsfall praktisch, effizient und gut nachvollziehbar.



## 3.3 SWRL-Regeln

### 3.3.1 Erfahrung

#### Klasse Erfahrung

Dient der Repräsentation des Erfahrungsniveaus eines Kunden im Umgang mit LEGO. Sie ist konzeptuell eng mit der Klasse Kunde verbunden, z. B. über die Property *hatErfahrung*.

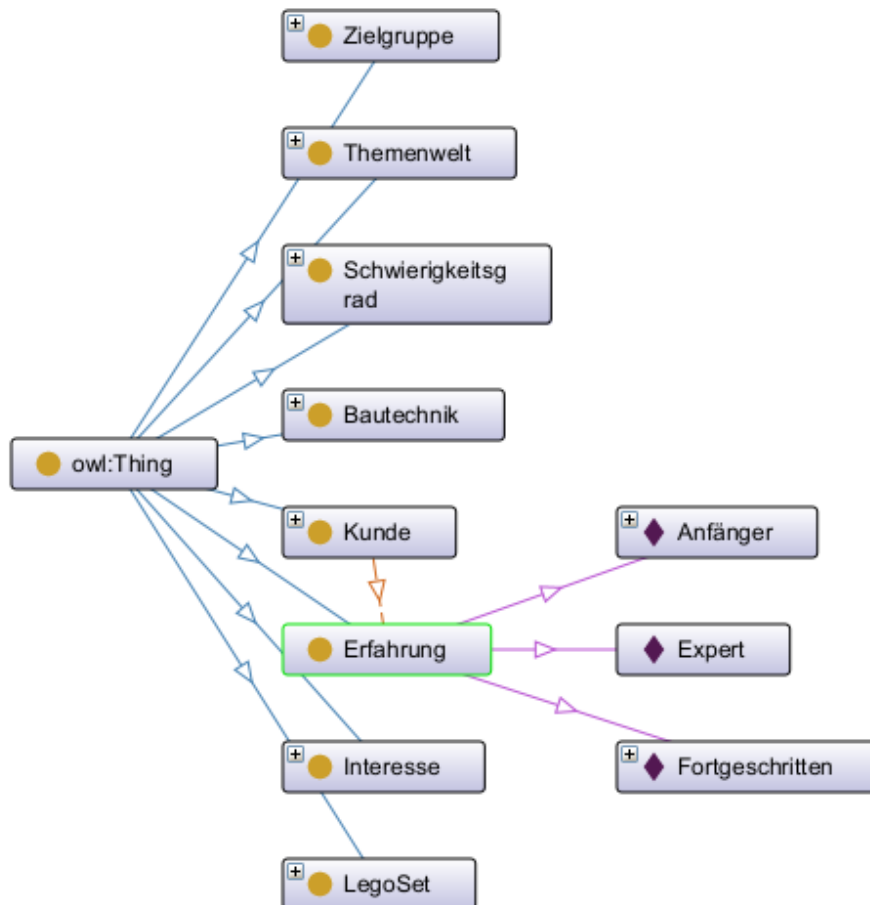


Abbildung 10: Wissensgraph "Erfahrung"

Die Klasse Erfahrung ist als eigenständige Oberklasse unterhalb von owl:Thing definiert und stellt eine qualitative Einordnung der Baufähigkeiten oder -historie eines Kunden dar.

## Struktur:

### Individuen (Ausprägungen):

- Anfänger
- Fortgeschritten
- Experte

Diese drei Individuen stellen abgestufte Erfahrungsstufen dar und ermöglichen die semantische Unterscheidung von Nutzern mit wenig bis sehr viel Bau-Erfahrung. Sie können sowohl direkt zugewiesen als auch mittels SWRL-Regeln automatisch inferiert werden – etwa basierend auf der Anzahl gebauter Sets und der bevorzugten Bautechniken.

### Regelbasierte Zuordnung der Erfahrungsstufen (SWRL)

Im Rahmen der Ontologie wurden SWRL-Regeln definiert, um das Erfahrungsniveau eines Kunden automatisch zu bestimmen. Die Einstufung erfolgt auf Basis der Anzahl gebauter Sets sowie der verwendeten Bautechniken. Der Datentyp der Anzahl (hatAnzahlGebauterSets) ist xsd:integer, weshalb sämtliche numerischen Literale in den Regeln typisiert, angegeben werden (z. B. "5"^^xsd:integer).

Nachfolgend sind die Regeln dokumentiert (Nummerierung ist gemäss dem «Namen» im Protégé SWRLTab):

---

#### Regel 11 – Anfänger: $\leq 5$ Sets & Bautechnik Anleitung

```
lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:lessThanOrEqual(?anz, 5) ^  
lego:hatBevorzugteBautechnik(?k, lego:Anleitung) -> lego:hatErfahrung(?k, lego:Anfänger)
```

---

#### Regel 12 – Anfänger: $\leq 5$ Sets & Bautechnik Modular

```
lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:lessThanOrEqual(?anz, 5) ^  
lego:hatBevorzugteBautechnik(?k, lego:Modular) -> lego:hatErfahrung(?k, lego:Anfänger)
```

---

#### Regel 13 – Fortgeschritten: 16–25 Sets & Bautechnik Anleitung

```
lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:greaterThan(?anz, 15) ^  
swrlb:lessThanOrEqual(?anz, 25) ^ lego:hatBevorzugteBautechnik(?k, lego:Anleitung) -> lego:ha-  
tErfahrung(?k, lego:Fortgeschritten)
```

---

---

**Regel 14 – Fortgeschritten: 16–25 Sets & Bautechnik Eigenkreation**

lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:greaterThan(?anz, 15) ^ swrlb:lessThanOrEqual(?anz, 25) ^ lego:hatBevorzugteBautechnik(?k, lego:Eigenkreation) -> lego:hatErfahrung(?k, lego:Fortgeschritten)

---

**Regel 16 – Fortgeschritten: ≤ 5 Sets & Bautechnik Eigenkreation**

lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:lessThanOrEqual(?anz, 5) ^ lego:hatBevorzugteBautechnik(?k, lego:Eigenkreation) -> lego:hatErfahrung(?k, lego:Fortgeschritten)

---

**Regel 22 – Fortgeschritten: 6–15 Sets & Anleitung**

lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:greaterThan(?anz, 5) ^ swrlb:lessThanOrEqual(?anz, 15) ^ lego:hatBevorzugteBautechnik(?k, lego:Anleitung) -> lego:hatErfahrung(?k, lego:Fortgeschritten)

---

**Regel 23 – Fortgeschritten: 6–15 Sets & Modular**

Lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:greaterThan(?anz, 5) ^ swrlb:lessThanOrEqual(?anz, 15) ^ lego:hatBevorzugteBautechnik(?k, lego:Modular) -> lego:hatErfahrung(?k, lego:Fortgeschritten)

---

**Regel 21 – Fortgeschritten: 6–15 Sets & Bautechnik Eigenkreation**

lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:greaterThan(?anz, 5) ^ swrlb:lessThanOrEqual(?anz, 15) ^ lego:hatBevorzugteBautechnik(?k, lego:Eigenkreation) -> lego:hatErfahrung(?k, lego:Fortgeschritten)

---

---

**Regel 18 – Experte: 16–25 Sets & Bautechnik Modular**

lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:greaterThan(?anz, 15) ^ swrlb:lessThanOrEqualTo(?anz, 25) ^ lego:hatBevorzugteBautechnik(?k, lego:Modular) -> lego:hatErfahrung(?k, lego:Expert)

---

**Regel 19 – Experte: 16–25 Sets & Bautechnik Eigenkreation**

lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:greaterThan(?anz, 15) ^ swrlb:lessThanOrEqualTo(?anz, 25) ^ lego:hatBevorzugteBautechnik(?k, lego:Eigenkreation) -> lego:hatErfahrung(?k, lego:Expert)

---

**Regel 20 – Experte: > 25 Sets & alle Bautechniken**

lego:Kunde(?k) ^ lego:hatAnzahlGebauterSets(?k, ?anz) ^ swrlb:greaterThan(?anz, 25) ^ lego:hatBevorzugteBautechnik(?k, lego:Anleitung) ^ lego:hatBevorzugteBautechnik(?k, lego:Modular) ^ lego:hatBevorzugteBautechnik(?k, lego:Eigenkreation) -> lego:hatErfahrung(?k, lego:Expert)

---

### 3.3.2 Themenwelt

#### Klasse Themenwelt:

Dient der semantischen Einordnung von LEGO-Produkten in inhaltliche Kategorien. Die Themenwelten fungieren als Bindeglied zwischen den Interessen des Kunden und konkreten LEGO-Sets.

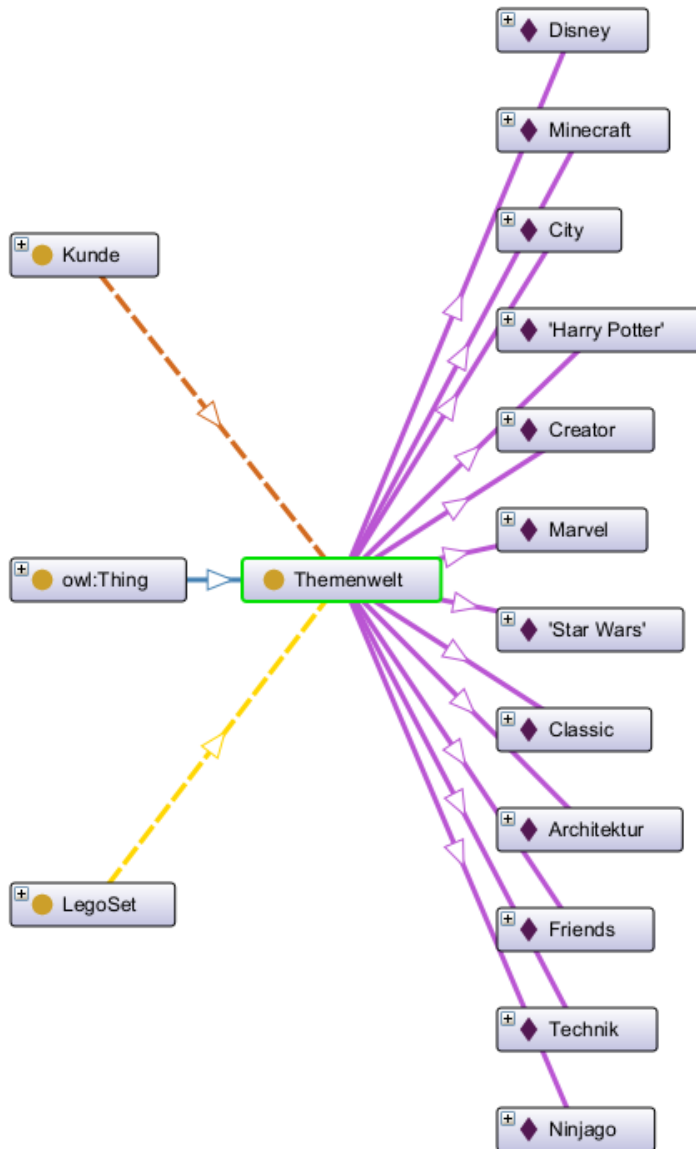


Abbildung 11: Wissensgraph "Themenwelt"

Die Klasse Themenwelt bildet die inhaltlichen Schwerpunkte ab, in welche LEGO-Sets thematisch einzuordnen sind z. B. „Star Wars“, „Minecraft“ oder „Friends“. Sie ist konzeptuell eng mit der Klasse **Kunde** verbunden, da Kunden je nach Interesse und Lizenzpräferenz einer bevorzugten Themenwelt zugewiesen werden können. Diese Zuweisung erfolgt automatisiert über die SWRL-Regeln mithilfe der Property **hatBevorzugteThemenwelt**.

## Struktur:

### Individuen (Ausprägungen):

- Technik
- Minecraft
- Classic
- Disney
- StarWars
- Ninjago
- HarryPotter
- Friends
- Marvel

### Regelbasierte Zuweisung der bevorzugten Themenwelt (SWRL)

Für die Zuweisung wurden konkrete SWRL-Regeln definiert. Die Regeln basieren jeweils auf einer Kombination von Kundeninteresse (hatKundenInteresse) und Lizenzpräferenz (wuenschtLizenz). Die daraus abgeleitete Themenwelt wird über das Objektproperty **hatBevorzugteThemenwelt** mit dem jeweiligen Kunden verknüpft.

Nachfolgend sind die Regeln dokumentiert (Nummerierung ist gemäss dem «Namen» im Protégé SWRLTab):

#### Regel 39 – Interesse: Gaming & Lizenz: true → Minecraft

```
lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:Gaming) ^ lego:wuenschtLizenz(?k, true) ->  
lego:hatBevorzugteThemenwelt(?k, lego:Minecraft)
```

---

#### Regel 40 – Interesse: SimpleSpielen & Lizenz: egal → Classic

```
lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:SimpleSpielen) ->  
lego:hatBevorzugteThemenwelt(?k, lego:Classic)
```

---

#### Regel 41 – Interesse: Disney & Lizenz: true → Disney

```
lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:Disney) ^ lego:wuenschtLizenz(?k, true)  
-> lego:hatBevorzugteThemenwelt(?k, lego:Disney)
```

---

#### Regel 42 – Interesse: SciFi & Lizenz: true → StarWars

```
lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:SciFi) ^ lego:wuenschtLizenz(?k, true)  
-> lego:hatBevorzugteThemenwelt(?k, lego:StarWars)
```

---

**Regel 43 – Interesse: Ninjas & Lizenz: egal → Ninjago**

lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:Ninjas)

-> lego:hatBevorzugteThemenwelt(?k, lego:Ninjago)

---

**Regel 44 – Interesse: Fantasy & Lizenz: true → HarryPotter**

lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:Fantasy) ^ lego:wuenschtLizenz(?k, true)

-> lego:hatBevorzugteThemenwelt(?k, lego:HarryPotter)

---

**Regel 45 – Interesse: TierNatur & Lizenz: false → Friends**

lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:TierNatur) ^ lego:wuenschtLizenz(?k, false)

-> lego:hatBevorzugteThemenwelt(?k, lego:Friends)

---

**Regel 46 – Interesse: Superheroes & Lizenz: true → Marvel**

lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:Superheroes) ^ lego:wuenschtLizenz(?k, true)

-> lego:hatBevorzugteThemenwelt(?k, lego:Marvel)

---

**Regel 47 – Interesse: Fahrzeuge & Lizenz: false → Technik**

lego:Kunde(?k) ^ lego:hatKundenInteresse(?k, lego:Fahrzeuge) ^ lego:wuenschtLizenz(?k, false)

-> lego:hatBevorzugteThemenwelt(?k, lego:Technik)

---

### 3.3.3 Zielgruppe

#### Klasse Zielgruppe

Dient der semantischen Einordnung von Kunden in eine Zielgruppe.

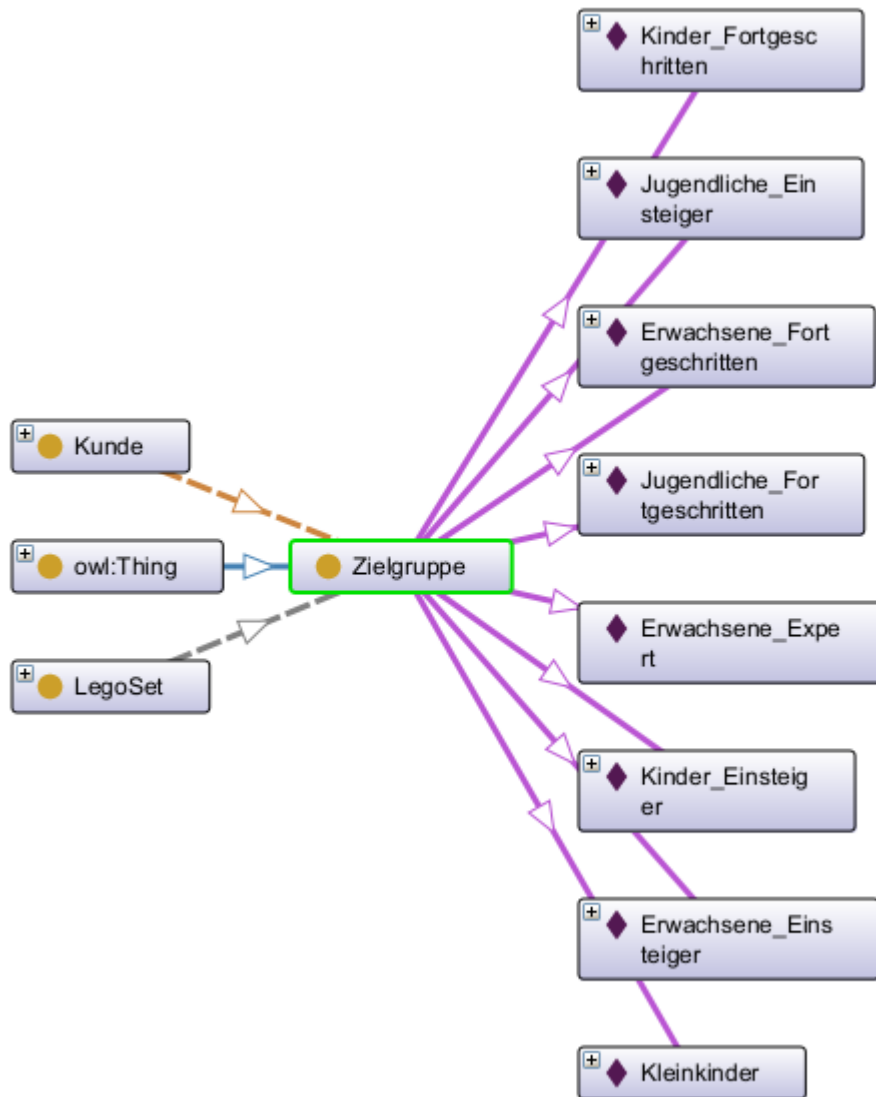


Abbildung 12: Wissensgraph "Zielgruppe"

Die Klasse **Zielgruppe** entspricht, wie der Name schon sagt, der Zielgruppen des Kunden. Etwa „Kleinkinder“, „Kinder\_Einsteiger“ oder „Erwachsene\_Experte“. Auf Basis von Alter (`hatAlter`) und Erfahrungsstufe (`hatErfahrung`) wird jeder Kunde einer von acht Gruppen zugewiesen. Die Zuweisung erfolgt automatisiert über SWRL-Regeln mithilfe der Objekt-Property **hatAbgeleiteteZielgruppe**.



## Struktur:

### Individuen (Ausprägungen)

- Kleinkinder
- Kinder\_Einsteiger
- Kinder\_Fortgeschritten
- Jugendliche\_Einsteiger
- Jugendliche\_Fortgeschritten
- Erwachsene\_Einsteiger
- Erwachsene\_Fortgeschritten
- Erwachsene\_Experte

Nachfolgend sind die Regeln dokumentiert (Nummerierung ist gemäss dem «Namen» im Protégé SWRLTab):

#### **Regel 31 – Zielgruppe: Kleinkinder (Alter < 4)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:lessThan(?a, 4)

-> lego:hatAbgeleiteteZielgruppe(?k, lego:Kleinkinder)

---

#### **Regel 32 – Zielgruppe: Kinder – Einsteiger ( $4 \leq \text{Alter} \leq 12$ & Erfahrung = Anfänger)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThanOrEqual(?a, 4)

^ swrlb:lessThanOrEqual(?a, 12) ^ lego:hatErfahrung(?k, lego:Anfänger)

-> lego:hatAbgeleiteteZielgruppe(?k, lego:Kinder\_Einsteiger)

---

#### **Regel 33.1 – Zielgruppe: Kinder – Fortgeschritten ( $4 \leq \text{Alter} \leq 12$ & Erfahrung = Fortgeschritten)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThanOrEqual(?a, 4)

^ swrlb:lessThanOrEqual(?a, 12) ^ lego:hatErfahrung(?k, lego:Fortgeschritten)

-> lego:hatAbgeleiteteZielgruppe(?k, lego:Kinder\_Fortgeschritten)

---

**Regel 33.2 – Zielgruppe: Kinder – Fortgeschritten ( $4 \leq \text{Alter} \leq 12$  & Erfahrung = Experte)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThanOrEqualTo(?a, 4)  
 ^ swrlb:lessThanOrEqualTo(?a, 12) ^ lego:hatErfahrung(?k, lego:Experte)  
 -> lego:hatAbgeleiteteZielgruppe(?k, lego:Kinder\_Fortgeschritten)

---

**Regel 34 – Zielgruppe: Jugendliche – Einsteiger ( $12 < \text{Alter} < 18$  & Erfahrung = Anfänger)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThan(?a, 12)  
 ^ swrlb:lessThanOrEqualTo(?a, 18) ^ lego:hatErfahrung(?k, lego:Anfänger)  
 -> lego:hatAbgeleiteteZielgruppe(?k, lego:Jugendliche\_Einsteiger)

---

**Regel 35.1 – Zielgruppe: Jugendliche – Fortgeschritten ( $12 < \text{Alter} < 18$  & Erfahrung = Fortgeschritten)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThan(?a, 12)  
 ^ swrlb:lessThanOrEqualTo(?a, 18) ^ lego:hatErfahrung(?k, lego:Fortgeschritten)  
 -> lego:hatAbgeleiteteZielgruppe(?k, lego:Jugendliche\_Fortgeschritten)

---

**Regel 35.2 – Zielgruppe: Jugendliche – Fortgeschritten ( $12 < \text{Alter} < 18$  & Erfahrung = Experte)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThan(?a, 12)  
 ^ swrlb:lessThanOrEqualTo(?a, 18) ^ lego:hatErfahrung(?k, lego:Experte)  
 -> lego:hatAbgeleiteteZielgruppe(?k, lego:Jugendliche\_Fortgeschritten)

---

**Regel 36 – Zielgruppe: Erwachsene – Einsteiger ( $\text{Alter} \geq 18$  & Erfahrung = Anfänger)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThanOrEqualTo(?a, 18)  
 ^ lego:hatErfahrung(?k, lego:Anfänger)  
 -> lego:hatAbgeleiteteZielgruppe(?k, lego:Erwachsene\_Einsteiger)

---

**Regel 37 – Zielgruppe: Erwachsene – Fortgeschritten (Alter  $\geq 18$  & Erfahrung = Fortgeschritten)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThanOrEqual(?a, 18)

^ lego:hatErfahrung(?k, lego:Fortgeschritten)

-> lego:hatAbgeleiteteZielgruppe(?k, lego:Erwachsene\_Fortgeschritten)

---

**Regel 38 – Zielgruppe: Erwachsene – Experte (Alter  $\geq 18$  & Erfahrung = Experte)**

lego:Kunde(?k) ^ lego:hatAlter(?k, ?a) ^ swrlb:greaterThanOrEqual(?a, 18)

^ lego:hatErfahrung(?k, lego:Experte)

-> lego:hatAbgeleiteteZielgruppe(?k, lego:Erwachsene\_Experte)

---

### 3.3.4 Schwierigkeitsgrad

#### Klasse Schwierigkeitsgrad

Dient der semantischen Beschreibung des Anspruchsniveaus eines LEGO-Sets.

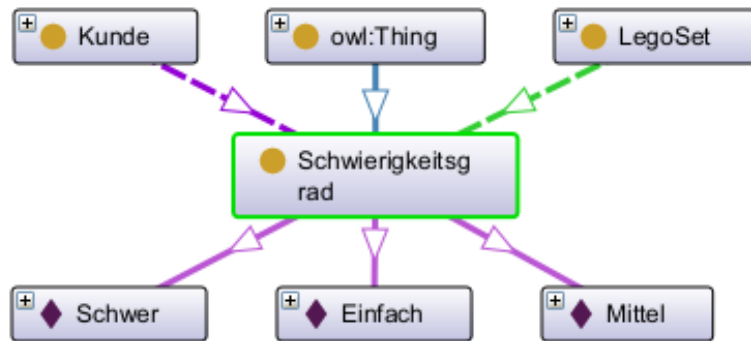


Abbildung 13: Wissensgraph "Schwierigkeitsgrad"

Die Klasse Schwierigkeitsgrad dient der semantischen Einordnung von LEGO-Sets nach ihrem Schwierigkeitsniveau. Die Einstufung basiert auf der Anzahl der Bauteile (praeferiertBauteilmenge), der geschätzten Bauzeit in Stunden (praeferiertBauzeit) sowie der Information, ob komplexe Spezialbausteine enthalten sind (praeferiertKomplexeBausteine).

Die Einordnung erfolgt automatisiert über SWRL-Regeln, wobei der abgeleitete Schwierigkeitsgrad über die Objekt-Property hatGewuenschtenSchwierigkeitsgrad zugewiesen wird.

#### Struktur:

##### Individuen (Ausprägungen)

- Einfach
- Mittel
- Schwer

Nachfolgend sind die Regeln dokumentiert (Nummerierung ist gemäss dem «Namen» im Protégé SWRLTab):

#### Regel 24 – Schwierigkeitsgrad: Einfach - $0 < \text{Bauteilmenge} \leq 1500 + \text{Bauzeit} \leq 2$

```
lego:Kunde(?k) ^ lego:praeferiertBauteilmenge(?k, ?anz) ^ swrlb:greaterThan(?anz, 0) ^
swrlb:lessThanOrEqual(?anz, 1500) ^ lego:praeferiertBauzeit(?k, ?h) ^ swrlb:lessThanOrEqual(?h,
2) ^ lego:praeferiertKomplexeBausteine(?k, false) -> lego:hatGewuenschtenSchwierigkeitsgrad(?k,
lego:Einfach)
```

**Regel 25 – Schwierigkeitsgrad: Mittel - Bauteilmenge  $\leq 500$  + Bauzeit  $> 2$  + Komplexe Bausteine "false"**

lego:Kunde(?k) ^ lego:praeferiertBauteilmenge(?k, ?anz) ^ swrlb:lessThanOrEqualTo(?anz, 500) ^ lego:praeferiertBauzeit(?k, ?h) ^ swrlb:greaterThan(?h, 2) ^ lego:praeferiertKomplexeBausteine(?k, false) -> lego:hatGewuenschtenSchwierigkeitsgrad(?k, lego:Mittel)

---

**Regel 26 – Schwierigkeitsgrad: Mittel -  $0 < \text{Bauteilmenge} \leq 1500$  +  $1 \leq \text{Bauzeit} \leq 5$  + Komplexe Bausteine "true"**

lego:Kunde(?k) ^ lego:praeferiertBauteilmenge(?k, ?anz) ^ swrlb:greaterThan(?anz, 0) ^ swrlb:lessThanOrEqualTo(?anz, 1500) ^ lego:praeferiertBauzeit(?k, ?h) ^ swrlb:greaterThanOrEqualTo(?h, 1) ^ swrlb:lessThanOrEqualTo(?h, 5) ^ lego:praeferiertKomplexeBausteine(?k, true) -> lego:hatGewuenschtenSchwierigkeitsgrad(?k, lego:Mittel)

---

**Regel 27 – Schwierigkeitsgrad: Mittel - Bauteilmenge  $\geq 500$**

lego:Kunde(?k) ^ lego:praeferiertBauteilmenge(?k, ?anz) ^ swrlb:greaterThanOrEqualTo(?anz, 500) ^ lego:praeferiertBauzeit(?k, ?h) ^ swrlb:greaterThanOrEqualTo(?h, 2) ^ swrlb:lessThanOrEqualTo(?h, 5) ^ lego:praeferiertKomplexeBausteine(?k, false) -> lego:hatGewuenschtenSchwierigkeitsgrad(?k, lego:Mittel)

---

**Regel 28 – Schwierigkeitsgrad: Mittel - Bauteilmenge  $> 1500$  + Bauzeit  $> 5$  + Komplexe Bausteine "true"**

lego:Kunde(?k) ^ lego:praeferiertBauteilmenge(?k, ?anz) ^ swrlb:greaterThan(?anz, 1500) ^ lego:praeferiertBauzeit(?k, ?h) ^ swrlb:greaterThanOrEqualTo(?h, 1) ^ swrlb:lessThanOrEqualTo(?h, 5) ^ lego:praeferiertKomplexeBausteine(?k, true) -> lego:hatGewuenschtenSchwierigkeitsgrad(?k, lego:Mittel)

---

**Regel 29 – Schwierigkeitsgrad: Schwer - Bauteilmenge  $\geq 500$  + Bauzeit  $> 5$  + Komplexe Bausteine "false"**

lego:Kunde(?k) ^ lego:praeferiertBauteilmenge(?k, ?anz) ^ swrlb:greaterThanOrEqualTo(?anz, 500) ^ lego:praeferiertBauzeit(?k, ?h) ^ swrlb:greaterThan(?h, 5) ^ lego:praeferiertKomplexeBausteine(?k, false) -> lego:hatGewuenschtenSchwierigkeitsgrad(?k, lego:Schwer)

---

**Regel 31 – Schwierigkeitsgrad: Schwer - Bauteilmenge nicht relevant + Bauzeit  $> 5$  + Komplexe Bausteine "true"**

lego:Kunde(?k) ^ lego:praeferiertBauzeit(?k, ?h) ^ swrlb:greaterThan(?h, 5) ^ lego:praeferiertKomplexeBausteine(?k, true) -> lego:hatGewuenschtenSchwierigkeitsgrad(?k, lego:Schwer)

---

### 3.3.5 LegoSet Empfehlungen Regeln

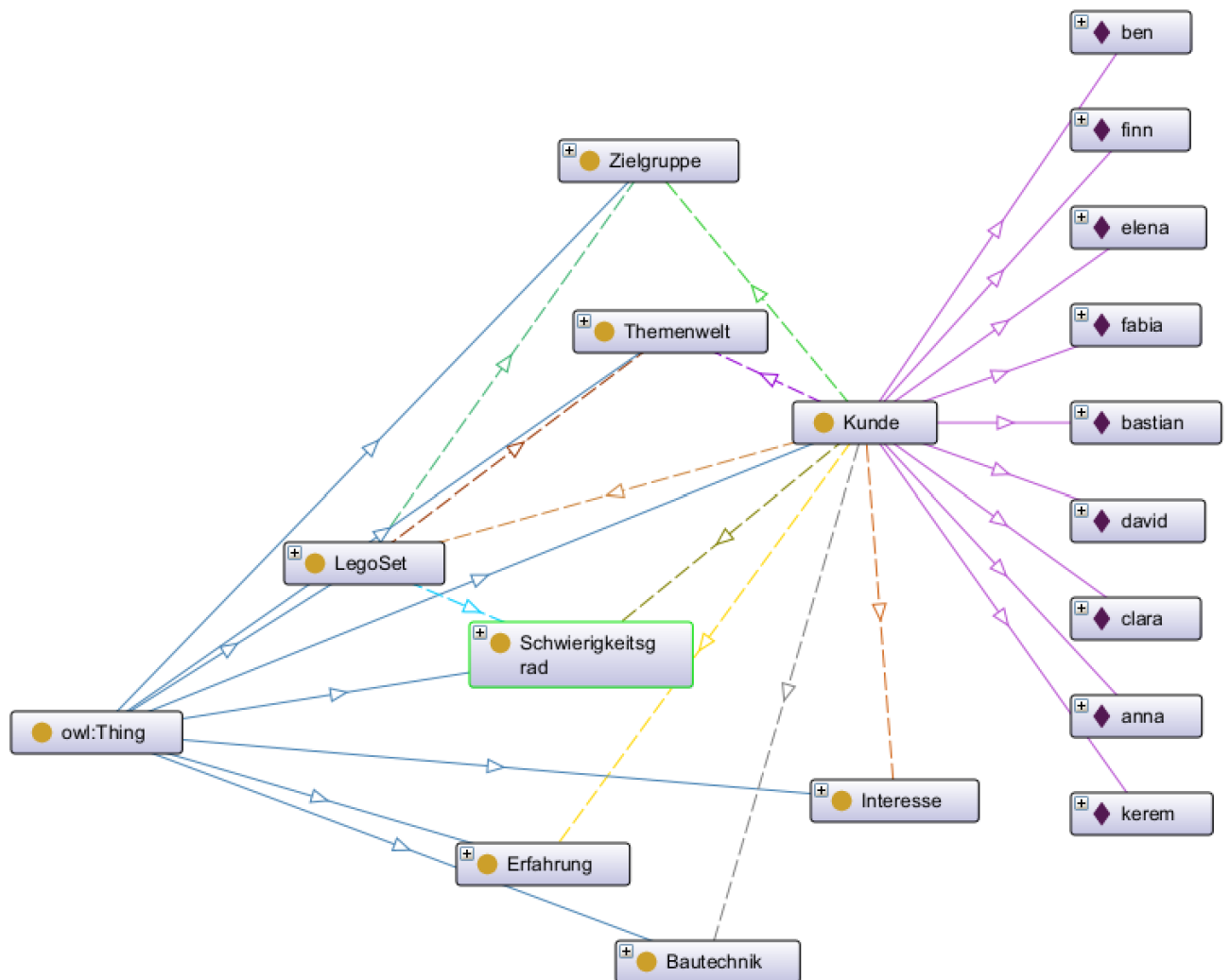


Abbildung 14: OntoGraf-Darstellung der LEGO-Ontologie für Beziehungen zu Kunden

Die Klasse Kunde steht im Zentrum der Ontologie und fasst alle Personen zusammen für die wir, basierend auf ihren Präferenzen und Daten, Empfehlungen generieren. Über SWRL-Regeln werden ihre individuell erfassten Eigenschaften (Alter, Preis, Gebaute Sets in der Vergangenheit, Interessen etc.) schrittweise zu Aussagen zusammengeschaltet. Von der Themenwelt über den Schwierigkeitsgrad und die Zielgruppe bis hin zur finalen Set-Empfehlung.

## Struktur:

Die Klasse Kunde repräsentiert einen einzelnen Nutzer bzw. Interessenten in unserem System.

### Data-Properties (Attribute eines Kunden)

- hatAlter
- hatWunschPreis
- wuenschtKompatibel
- hatAnzahlGebauterSets
- praeferiertBauteilmenge
- praeferiertBauzeit
- praeferiertKomplexeBausteine
- wuenschtLizenz

### Object-Properties (Verknüpfungen zu anderen Konzepten)

- hatErfahrung → Verbindung zur Klasse **Erfahrung**
- hatGewuenschtenSchwierigkeitsgrad → zu **Schwierigkeitsgrad**
- hatKundenInteresse → zu **Interesse**
- hatBevorzugteBautechnik → zu **Bautechnik**
- hatBevorzugteThemenwelt → zu **Themenwelt**
- hatAbgeleiteteZielgruppe → zu **Zielgruppe**
- empfiehltSetFuerKunde → zu **LegoSet** (endgültige Set-Empfehlung)

## Individuen

- anna
- ben
- clara
- david
- elena
- fabia
- finn
- kerem
- bastian

Nachfolgend sind die Regeln dokumentiert (Nummerierung ist gemäss dem «Namen» im Protégé SWRLTab):

### **Regel 48 – LegoSet: Minecraft\_mini\_Höhle für Anna**

lego:Kunde(lego:anna) ^ lego:hatAlter(lego:anna, 8) ^ lego:hatWunschPreis(lego:anna, 45) ^ lego:wuenschtKompatibel(lego:anna, true) ^ lego:hatBevorzugteBautechnik(lego:anna, lego:Anleitung) ^ lego:hatAnzahlGebauterSets(lego:anna, 2) ^ lego:praeferiertBauteilmenge(lego:anna, 200) ^ lego:praeferiertBauzeit(lego:anna, 1) ^ lego:praeferiertKomplexeBausteine(lego:anna, false) ^ lego:wuenschtLizenz(lego:anna, true) ^ lego:hatKundenInteresse(lego:anna, lego:Gaming) -> lego:empfehlSetFuerKunde(lego:anna, lego:Minecraft\_Mini\_Höhle)

---

### **Regel 49 – LegoSet: Mein\_erster\_Bauernhof für Ben**

lego:Kunde(lego:ben) ^ lego:hatAlter(lego:ben, 3) ^ lego:hatWunschPreis(lego:ben, 30) ^ lego:wuenschtKompatibel(lego:ben, true) ^ lego:hatBevorzugteBautechnik(lego:ben, lego:Anleitung) ^ lego:hatAnzahlGebauterSets(lego:ben, 0) ^ lego:praeferiertBauteilmenge(lego:ben, 50) ^ lego:praeferiertBauzeit(lego:ben, 1) ^ lego:praeferiertKomplexeBausteine(lego:ben, false) ^ lego:wuenschtLizenz(lego:ben, false) ^ lego:hatKundenInteresse(lego:ben, lego:Simple\_Spielen) -> lego:empfehlSetFuerKunde(lego:ben, lego:Mein\_erster\_Bauernhof)

---



### Regel 50 – LegoSet: Ultimatives\_Abenteuerschloss für Clara

lego:Kunde(lego:clara) ^ lego:hatAlter(lego:clara, 10) ^ lego:hatWunschPreis(lego:clara, 150) ^ lego:wuenschtKompatibel(lego:clara, true) ^ lego:hatBevorzugteBautechnik(lego:clara, lego:Anleitung) ^ lego:hatAnzahlGebauterSets(lego:clara, 6) ^ lego:praeferiertBauteilmenge(lego:clara, 500) ^ lego:praeferiertBauzeit(lego:clara, 3) ^ lego:praeferiertKomplexeBausteine(lego:clara, false) ^ lego:wuenschtLizenz(lego:clara, true) ^ lego:hatKundenInteresse(lego:clara, lego:Disney) -> lego:empfiehltSetFuerKunde(lego:clara, lego:Ultimatives\_Abenteuerschloss)

---

### Regel 51 – LegoSet: Darth\_Mauls\_Sith\_Infiltrator für David

lego:Kunde(lego:david) ^ lego:hatAlter(lego:david, 18) ^ lego:hatWunschPreis(lego:david, 80) ^ lego:wuenschtKompatibel(lego:david, true) ^ lego:hatBevorzugteBautechnik(lego:david, lego:Anleitung) ^ lego:hatAnzahlGebauterSets(lego:david, 2) ^ lego:praeferiertBauteilmenge(lego:david, 800) ^ lego:praeferiertBauzeit(lego:david, 4) ^ lego:praeferiertKomplexeBausteine(lego:david, true) ^ lego:wuenschtLizenz(lego:david, true) ^ lego:hatKundenInteresse(lego:david, lego:Sci\_Fi) -> lego:empfiehltSetFuerKunde(lego:david, lego:Darth\_Mauls\_Sith\_Infiltrator)

---

### Regel 52 – LegoSet: Spinjitzu\_Tempel\_der\_Ninja für Finn

lego:Kunde(lego:finn) ^ lego:hatAlter(lego:finn, 6) ^ lego:hatWunschPreis(lego:finn, 40) ^ lego:wuenschtKompatibel(lego:finn, true) ^ lego:hatBevorzugteBautechnik(lego:finn, lego:Anleitung) ^ lego:hatAnzahlGebauterSets(lego:finn, 1) ^ lego:praeferiertBauteilmenge(lego:finn, 150) ^ lego:praeferiertBauzeit(lego:finn, 1) ^ lego:praeferiertKomplexeBausteine(lego:finn, false) ^ lego:wuenschtLizenz(lego:finn, true) ^ lego:hatKundenInteresse(lego:finn, lego:Ninjas) -> lego:empfiehltSetFuerKunde(lego:finn, lego:Spinjitzu\_Tempel\_der\_Ninja)

---

### Regel 53 – LegoSet: Schloss\_Hogwarts\_mit\_Schlossgelände für Kerem

lego:Kunde(lego:kerem) ^ lego:hatAlter(lego:kerem, 12) ^ lego:hatWunschPreis(lego:kerem, 250) ^ lego:wuenschtKompatibel(lego:kerem, true) ^ lego:hatBevorzugteBautechnik(lego:kerem, lego:Eigenkreation) ^ lego:hatAnzahlGebauterSets(lego:kerem, 8) ^ lego:praeferiertBauteilmenge(lego:kerem, 600) ^ lego:praeferiertBauzeit(lego:kerem, 5) ^ lego:praeferiertKomplexeBausteine(lego:kerem, true) ^ lego:wuenschtLizenz(lego:kerem, true) ^ lego:hatKundenInteresse(lego:kerem, lego:Fantasy) -> lego:empfiehltSetFuerKunde(lego:kerem, lego:Schloss\_Hogwarts\_mit\_Schlossgelände)

---

#### **Regel 54 – LegoSet: McLaren\_P1 für Bastian**

lego:Kunde(lego:bastian) ^ lego:hatAlter(lego:bastian, 40) ^ lego:hatWunschPreis(lego:bastian, 400) ^ lego:hatBevorzugteBautechnik(lego:bastian, lego:Modular) ^ lego:hatAnzahlGebauterSets(lego:bastian, 10) ^ lego:praeferiertBauteilmenge(lego:bastian, 3000) ^ lego:praeferiertBauzeit(lego:bastian, 15) ^ lego:praeferiertKomplexeBausteine(lego:bastian, true) ^ lego:wuenschtLizenz(lego:bastian, false) ^ lego:hatKundenInteresse(lego:bastian, lego:Fahrzeuge) -> lego:empfiehlSetFuerKunde(lego:bastian, lego:McLaren\_P1)

---

#### **Regel 55 – LegoSet: Paisleys\_Haus für Fabia**

lego:Kunde(lego:fabia) ^ lego:hatAlter(lego:fabia, 7) ^ lego:hatWunschPreis(lego:fabia, 60) ^ lego:wuenschtKompatibel(lego:fabia, true) ^ lego:hatBevorzugteBautechnik(lego:fabia, lego:Anleitung) ^ lego:hatAnzahlGebauterSets(lego:fabia, 3) ^ lego:praeferiertBauteilmenge(lego:fabia, 250) ^ lego:praeferiertBauzeit(lego:fabia, 2) ^ lego:praeferiertKomplexeBausteine(lego:fabia, false) ^ lego:wuenschtLizenz(lego:fabia, false) ^ lego:hatKundenInteresse(lego:fabia, lego:Tier\_or\_Natur) -> lego:empfiehlSetFuerKunde(lego:fabia, lego:Paisleys\_Haus)

---

#### **Regel 56 – LegoSet: X\_Men\_X\_Mansion für Elena**

lego:Kunde(lego:elena) ^ lego:hatAlter(lego:elena, 30) ^ lego:hatWunschPreis(lego:elena, 400) ^ lego:wuenschtKompatibel(lego:elena, true) ^ lego:hatBevorzugteBautechnik(lego:elena, lego:Anleitung) ^ lego:hatAnzahlGebauterSets(lego:elena, 20) ^ lego:praeferiertBauteilmenge(lego:elena, 5000) ^ lego:praeferiertBauzeit(lego:elena, 20) ^ lego:praeferiertKomplexeBausteine(lego:elena, true) ^ lego:wuenschtLizenz(lego:elena, true) ^ lego:hatKundenInteresse(lego:elena, lego:Superheroes) -> lego:empfiehlSetFuerKunde(lego:elena, lego:X\_Men\_\_X\_Mansion)

---

### 3.4 SPARQL-Abfragen

Zur gezielten Analyse und Auswertung der Ontologie wurden verschiedene SPARQL-Abfragen definiert. Diese ermöglichen es, Daten aus der Wissensbasis strukturiert abzufragen, zu filtern und logisch zu verknüpfen. Ziel war es, sowohl Inhalte zur Kunden- und Setstruktur abzuleiten als auch die Funktionsfähigkeit der inferenzbasierten Empfehlungen zu prüfen.

Im Rahmen dieses Projekts wurden sieben Abfragen ausgewählt, um unterschiedliche Aspekte der Ontologie zu beleuchten:

#### 1. *Finde alle Kunden und deren empfohlene LEGO-Sets:*

SPARQL query:	kunde	set
PREFIX : <http://www.example.org/lego-ontology#>	david	Darth Mauls Sith Infiltrator
SELECT ?kunde ?set	anna	Minecraft Mini-Höhle
WHERE {	elena	X_Men_X_Mansion
?kunde a :Kunde .	fabia	Paisleys Haus
?kunde :empfiehltSetFuerKunde ?set .	clara	Ultimates Abenteuerschloss
}	kerem	Schloss Hogwarts mit Schlossgelände
	bastian	McLaren P1
	ben	Mein erster Bauernhof
	finn	Spinjitzu-Tempel der Ninja

Abbildung 15: SPARQL-Abfrage und Ergebnis – empfohlene Sets je Kunde

#### 2. *Finde alle Sets, die zur Zielgruppe „Kinder - Einsteiger“ gehören:*

SPARQL query:	set
PREFIX : <http://www.example.org/lego-ontology#>	Minecraft Mini-Höhle
SELECT ?set	Donut Truck
WHERE {	Paisleys Haus
?set :hatSetEigeneZielgruppe :Kinder_Einsteiger .	Kreativer Reisekoffer
}	Bunte Bausteine-Box

Abbildung 16: SPARQL-Abfrage – Sets mit Zielgruppe „Kinder – Einsteiger“

**3. Gib alle LEGO-Sets mit Themenwelt „Disney“ zurück:**

SPARQL query:	set
<pre>PREFIX : &lt;http://www.example.org/lego-ontology#&gt;  SELECT ?set WHERE {   ?set :gehörtZuThemenwelt :Disney . }</pre>	Disney Hocus Pocus: Das Hexenhaus der Sanderson-Schwwestern 3-in-1-Zauberschloss Ultimatives Abenteuerschloss

Abbildung 17: SPARQL-Abfrage und Ergebnis – LEGO-Sets mit Themenwelt „Disney“

**4. Finde Kunden mit Interesse an „Gaming“ und Lizenzwunsch = true:**

SPARQL query:	kunde
<pre>PREFIX : &lt;http://www.example.org/lego-ontology#&gt;  SELECT ?kunde WHERE {   ?kunde a :Kunde ;     :hatKundenInteresse :Gaming ;     :wuenschtLizenz true . }</pre>	anna

Abbildung 18: SPARQL-Abfrage und Ergebnis – Kunden mit Interesse an Gaming und Lizenzwunsch

**5. Gib alle Kunden und deren Zielgruppe (abgeleitet):**

SPARQL query:	kunde	zielgruppe
<pre>PREFIX : &lt;http://www.example.org/lego-ontology#&gt;  SELECT ?kunde ?zielgruppe WHERE {   ?kunde a :Kunde ;     :hatAbgeleiteteZielgruppe ?zielgruppe . }</pre>	anna	Kinder_Einsteiger
	elena	Erwachsene_Fortgeschritten
	finn	Kinder_Einsteiger
	bastian	Erwachsene_Fortgeschritten
	ben	Kleinkinder
	david	Erwachsene_Einsteiger
	kerem	Kinder_Fortgeschritten
	fabia	Kinder_Einsteiger
	clara	Kinder_Fortgeschritten

Abbildung 19: SPARQL-Abfrage und Ergebnis – Kunden und ihre abgeleitete Zielgruppe

## 6. Finde alle Sets mit Schwierigkeitsgrad „Einfach“:

SPARQL query:	
PREFIX : <http://www.example.org/lego-ontology#>	
SELECT ?set	
WHERE {	
?set :hatSetSchwierigkeitsgrad :Einfach .	
}	
set	
Trevi-Brunnen	
Geschäft für Haustierzubehör	
Abenteuer - Wohnmobil	
Minecraft Mini-Höhle	
Darth Vader Helm	
Obi-Wan Kenobis Jedi Startfighter	
3-in-1-Zauberschloss	
Wilde Tiere: Rosa Flamingo	
Donut Truck	
Paisleys Haus	
Bunte Bausteine-Box	
London	
Fast and Furious Toyota Supra MK4	
Grosse interaktive Eisenbahn	
R2-D2	
Mein erster Bauernhof	
Kreativer Reisekoffer	

Abbildung 20: SPARQL-Abfrage und Ergebnis – LEGO-Sets mit Schwierigkeitsgrad „Einfach“

## 7. Suche alle Kunden mit Wunschpreis unter < 110:

SPARQL query:		kunde	preis
PREFIX : <http://www.example.org/lego-ontology#>		ben	"30"^^<http://www.w3.org/2001/XMLSchema#integer>
SELECT ?kunde ?preis		david	"80"^^<http://www.w3.org/2001/XMLSchema#integer>
WHERE {		fabia	"60"^^<http://www.w3.org/2001/XMLSchema#integer>
?kunde a :Kunde ;		anna	"45"^^<http://www.w3.org/2001/XMLSchema#integer>
:hatWunschPreis ?preis .		finn	"40"^^<http://www.w3.org/2001/XMLSchema#integer>
FILTER (?preis < "110"^^xsd:integer)			
}			

Abbildung 21: SPARQL-Abfrage und Ergebnis – Kunden mit Wunschpreis unter 110 CHF

Die gewählten Abfragen decken sowohl Regelkontrolle, Zielgruppenlogik, als auch Produkttypen ab und helfen dabei, die Ontologie inhaltlich sowie technisch zu validieren.



### 3.5 Visualisierung des Wissensgraphen

Zur besseren Nachvollziehbarkeit wurde der strukturierte Aufbau der Ontologie mittels OntoGraf in Protégé visualisiert. Der Graph zeigt die Klassenhierarchie sowie die semantischen Beziehungen zwischen allen Klassen.

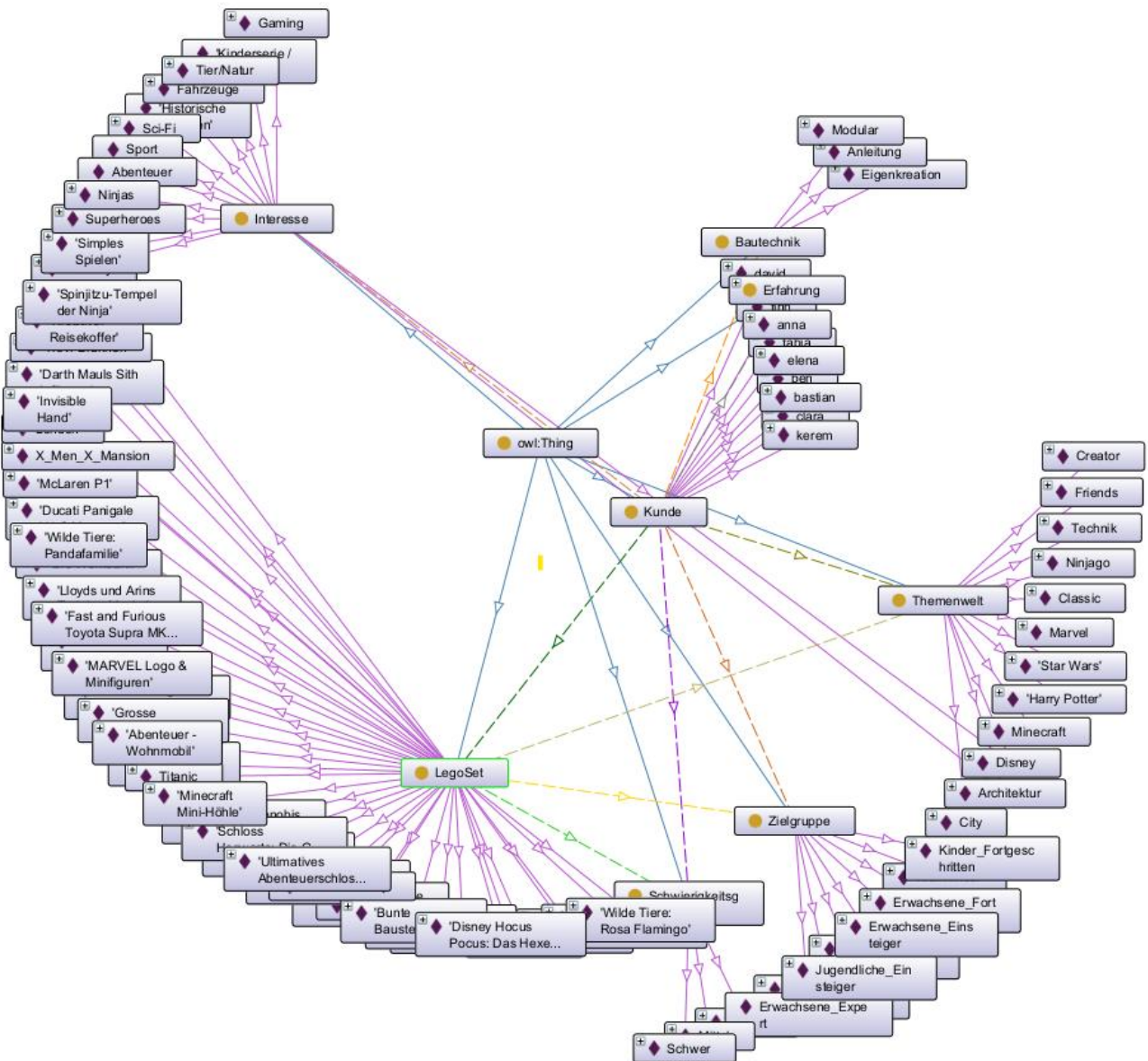


Abbildung 22: Gesamtübersicht der LEGO-Ontologie mit Instanzen, Klassen und Beziehungen

Die Abbildung zeigt die komplette Struktur unserer Ontologie in Protégé, visualisiert mit dem Tool OntoGraf. Im Zentrum steht die Klasse Kunde, also die LEGO-Käuferinnen und -Käufer. Um sie herum sind die anderen wichtigen Begriffe (Klassen) angeordnet, wie z. B. LegoSet, Interesse, Bautechnik, Zielgruppe, Themenwelt, Erfahrung und Schwierigkeitsgrad.

Die Pfeile in unterschiedlichen Farben und Formen stellen sogenannte Object Properties dar, also semantische Verbindungen zwischen Klassen oder Instanzen:

- Kunden sind z. B. über die Beziehung `hatKundenInteresse` mit bestimmten Interessen wie „Gaming“ oder „Superheroes“ verbunden.
- Über `hatBevorzugteBautechnik` wird gespeichert, wie jemand bevorzugt baut – etwa mit Anleitung oder frei als Eigenkreation.
- Die Empfehlung eines konkreten LEGO-Sets erfolgt über die Property `empfiehltSetFuerKunde`.

Rund um die zentrale Klasse `Kunde` sind die konkreten Kundeninstanzen wie `anna`, `finn`, `elena` oder `bastian` zu sehen. Jeder dieser Kunden ist mit individuellen Eigenschaften verknüpft, die als Grundlage für die spätere Set-Empfehlung dienen.

Auf der linken Seite befinden sich die LEGO-Sets, zum Beispiel „Minecraft Mini-Höhle“ oder „Trevi-Brunnen“. Diese sind mit ihren jeweiligen Themenwelten (z. B. „Disney“), Zielgruppen (z. B. „Kinder\_Einsteiger“) und Schwierigkeitsgraden (z. B. „Einfach“) verbunden.

Die OntoGraf-Darstellung macht deutlich, wie die Ontologie als semantisches Netzwerk aufgebaut ist:

Durch die Kombination aus Kundeneigenschaften, Setmerkmalen und regelbasierter Logik (SWRL) entsteht ein intelligentes Empfehlungssystem, das strukturiert und nachvollziehbar passende LEGO-Produkte vorschlagen kann.

### 3.6 Umsetzung in Protégé

Durch die Übertragung der Prolog-Regeln in eine OWL-Ontologie Datei wurde eine gut strukturierte und verständliche Wissensbasis aufgebaut. Diese ermöglicht sowohl manuelle Auswertungen als auch automatische Empfehlungen. Die Verwendung von OWL, SWRL und SPARQL bietet dabei eine solide Grundlage, um das System bei Bedarf weiter auszubauen.

## 4 Abschluss

Im Verlauf dieses Projekts haben wir ein intelligentes Empfehlungssystem für LEGO-Sets entwickelt, das schön zeigt, wie verschiedene Methoden der Wissensrepräsentation ineinandergreifen können. Dabei war uns wichtig, die theoretischen Konzepte praxisnah von der Entscheidungslogik über regelbasierte Systeme bis hin zur Ontologie umzusetzen.

Den Anfang bildete das Entscheidungsmodell in Trisotech, das wir anhand realer Use Cases aus dem LEGO-Online-Shop entwickelt haben. Durch Subdecisions konnten wir eine modulare Logik aufbauen, die es ermöglicht, individuelle Präferenzen wie Interessen, Erfahrung oder Budget schrittweise in die Produktempfehlung einfließen zu lassen. Dabei wurde deutlich, wie wichtig eine klare Strukturierung von Entscheidungsfaktoren ist, um spätere Ableitungen konsistent zu gestalten.

Anschliessend haben wir diese Logik in die deklarative Programmiersprache Prolog übersetzt. Die Herausforderung bestand darin, unsere Überlegungen aus dem DMN-Modell so zu formalisieren, dass sie in Form von Regeln und Fakten automatisiert auswertbar sind. Dabei entwickelten wir ein tieferes Verständnis für logikbasierte Systeme und sahen, wie sich mit einem geschickten Regelwerk aus wenigen Informationen differenzierte Empfehlungen ableiten lassen.

Zum Abschluss setzten wir unser System in Protégé als Wissensgraph um. Mit OWL, SWRL und SPARQL konnten wir nicht nur die Beziehungen zwischen Kunden, Interessen und Produkten sichtbar machen, sondern auch formalisieren. Die Arbeit mit Protégé zeigte uns, wie mächtig und zugleich komplex Ontologien sein können. Dies besonders wenn es darum geht, Regeln und semantische Beziehungen zu kombinieren.

Was das Projekt besonders gemacht hat, war unsere starke Teamarbeit. Wir haben effizient zusammengearbeitet, Arbeitspakete klar verteilt und konnten uns jederzeit aufeinander verlassen. Ideen wurden gemeinsam diskutiert, Vorschläge konstruktiv eingebracht, und bei Problemen fanden wir stets im Team eine Lösung. Insgesamt konnten wir mit unserem LEGO-Modell ein vollständiges und funktionierendes Beispiel für ein wissensbasiertes Empfehlungssystem entwickeln. Es zeigt, wie man theoretische Grundlagen praxisnah anwenden kann und dass auch scheinbar einfache Fragen wie «Welches LEGO-Set passt zu mir?» eine überraschend komplexe Logik im Hintergrund erfordern. Wir nehmen aus diesem Projekt nicht nur technische Skills mit, sondern auch ein besseres Verständnis für strukturierte Entscheidungsprozesse und gute Zusammenarbeit.



## Abbildungen

Abbildung 1: DMN Entscheidungsmodell für LEGO Sets.....	7
Abbildung 2: Entscheidungstabelle – Subdecision Themenwelt .....	11
Abbildung 3: Entscheidungstabelle – Subdecision Erfahrung .....	12
Abbildung 4: Entscheidungstabelle – Subdecision Zielgruppe .....	13
Abbildung 5: Entscheidungstabelle – Lego Set.....	14
Abbildung 6: Zentrale Klassen im Protégé.....	29
Abbildung 7: Objekteigenschaften im Protégé .....	30
Abbildung 8: Dateneigenschaften im Protégé.....	31
Abbildung 9: Klassenhierarchie im Protégé .....	32
Abbildung 10: Wissensgraph "Erfahrung" .....	33
Abbildung 11: Wissensgraph "Themenwelt" .....	37
Abbildung 12: Wissensgraph "Zielgruppe".....	40
Abbildung 13: Wissensgraph "Schwierigkeitsgrad" .....	44
Abbildung 14: OntoGraf-Darstellung der LEGO-Ontologie für Beziehungen zu Kunden .....	46
Abbildung 15: SPARQL-Abfrage und Ergebnis – empfohlene Sets je Kunde .....	51
Abbildung 16: SPARQL-Abfrage – Sets mit Zielgruppe „Kinder – Einsteiger“ .....	51
Abbildung 17: SPARQL-Abfrage und Ergebnis – LEGO-Sets mit Themenwelt „Disney“ .....	52
Abbildung 18: SPARQL-Abfrage und Ergebnis – Kunden mit Interesse an Gaming und Lizenzwunsch.....	52
Abbildung 19: SPARQL-Abfrage und Ergebnis – Kunden und ihre abgeleitete Zielgruppe .....	52
Abbildung 20: SPARQL-Abfrage und Ergebnis – LEGO-Sets mit Schwierigkeitsgrad „Einfach“ ....	53
Abbildung 21: SPARQL-Abfrage und Ergebnis – Kunden mit Wunschpreis unter 110 CHF .....	53
Abbildung 22: Gesamtübersicht der LEGO-Ontologie mit Instanzen, Klassen und Beziehungen...	54





# Anhang/Anhänge

## Anhang 1: Eigenständigkeitserklärung

Wir erklären hiermit,

- dass wir den vorliegenden Leistungsnachweis selbst und selbständig verfasst habe;
- dass wir sämtliche nicht von mir selbst stammenden Textstellen gemäss gängigen wissenschaftlichen Zitierregeln korrekt zitiert und die verwendeten Quellen gut sichtbar erwähnt habe;
- dass wir in einem Verzeichnis alle verwendeten Hilfsmittel (z. B. KI-Assistenzsysteme wie Chatbots [z. B. ChatGPT], Übersetzungs-, Paraphrasier-Tools) oder Programmierapplikationen [z. B. Github Copilot] deklariert und ihre Verwendung bei den entsprechenden Textstellen angegeben habe;
- dass wir sämtliche immateriellen Rechte an von mir allfällig verwendeten Materialien wie Bildern oder Grafiken erworben habe oder dass diese Materialien von mir selbst erstellt wurden;
- dass das Thema, die Arbeit oder Teile davon nicht bei einem Leistungsnachweis eines anderen Moduls verwendet wurden, sofern dies nicht ausdrücklich mit der Dozentin oder dem Dozenten im Voraus vereinbart wurde und in der Arbeit ausgewiesen wird;
- dass wir uns bewusst sind, dass die Arbeit auf Plagiate und auf Drittautorschaft menschlichen oder technischen Ursprungs (künstliche Intelligenz) überprüft werden kann;
- dass wir uns bewusst sind, dass die Hochschule für Wirtschaft FHNW einen Verstoss gegen diese Eigenständigkeitserklärung bzw. die ihr zugrundeliegenden Pflichten der Studien- und Prüfungsordnung der Hochschule für Wirtschaft FHNW verfolgt. Zu den Pflichten gehört insbesondere die Wahrung von Urheberrechten und das Unterlassen von Plagiaten (§ 29 Abs. 1 lit. d StuPO). Ein diesbezüglicher Verstoss wird mit der Note 1 bewertet und es können zusätzlich disziplinarische Folgen (Verweis/Ausschluss aus dem Studiengang) resultieren.

Samstag, 01. Juni 2025

Vor- und Nachname	Unterschrift
Bastian Büeler	
Fabia Holzer	
Elena Pineider	
Kerem Akkaya	

## Anhang 2: Hilfsmittelverzeichnis

Hilfsmittel	Verwendung	Betroffene Stellen
ChatGPT u/o DeepL Write	Korrektur auf Rechtschreibfehler	Ganzes Dokumentation
ChatGPT	ChatGPT wurde genutzt um bei den Programmen wie Visual Studio Code, Protégé Fehler zu erruiren und zu lösen.	Siehe die einzelnen Prompts weiter unten.
Literatur: A Practical Guide to Building OWL Ontologies Using Protégé 5.5 and Plugins	Diente der Fehleranalyse bei ausbleibender Inferenz durch die SWRL-Regeln.	Durch die Literaturrecherche wurde deutlich, dass für die erfolgreiche Ausführung der SWRL-Regeln in Protégé das Plugin „Pellet Reasoner“ erforderlich ist. Erst mit dessen Installation lassten sich die definierten Regeln korrekt anwenden und Inferenzprozesse auslösen.

## ChatGPT Prompt 1: «“Wieso kann ich im visual studio code nichts eingeben. Ich habe einen Pull gemacht aber wenn ich berarbeiten möchte steht editor add the file is read only.»

Das Problem, dass du im Visual Studio Code nichts eingeben kannst und eine "read-only" Fehlermeldung siehst, tritt auf, weil die Datei entweder schreibgeschützt ist oder durch Git als unverändert markiert wurde. Dies passiert häufig nach einem Pull, bei dem die Datei mit einem schreibgeschützten Status geladen wurde.

Hier sind einige Schritte, die du ausprobieren kannst:

### 1. Überprüfe die Datei-Eigenschaften:

- Stelle sicher, dass die Datei nicht als "schreibgeschützt" markiert ist. Rechtsklicke auf die Datei, wähle "Eigenschaften" und überprüfe, ob die Option "Schreibgeschützt" aktiviert ist. Wenn ja, deaktiviere sie.

### 2. Überprüfe den Git-Status:

- Öffne das Terminal in VS Code und führe den Befehl `git status` aus, um zu sehen, ob es Konflikte oder Änderungen gibt. Wenn die Datei schreibgeschützt ist, könnte es sein, dass ein Konflikt oder eine gesperrte Datei vorliegt.

### 3. Überprüfe die Berechtigungen der Datei:

- Wenn du auf einem Unix-basierten System (wie macOS oder Linux) arbeitest, überprüfe die Berechtigungen der Datei mit dem Befehl `ls -l`. Stelle sicher, dass du Schreibrechte hast.

### 4. Bearbeite die Datei manuell:

- Falls die Datei nach dem Pull als schreibgeschützt markiert wurde, kannst du mit `git checkout -- <filename>` versuchen, die Datei wieder in den bearbeitbaren Zustand zu versetzen.

### 5. Neustart von VS Code:

- Manchmal kann ein Neustart von Visual Studio Code helfen, den "read-only" Modus zu beenden, besonders wenn es durch einen Git-Pull oder eine falsche Einstellung verursacht wurde.

Probier diese Schritte aus, und schau, ob das Problem dadurch behoben wird.

## ChatGPT Prompt 2: «Wie kann ich ein mein Wissensgraphen vergössern?»

### ✓ 1. In Protégé: OntoGraf

Zeigt: Klassen & Object Properties (TBox)

So geht's:

1. `Window > Tabs > OntoGraf` aktivieren
2. Im „OntoGraf“-Tab:
  - Klicke auf das Plus-Symbol „Add root class“
  - Wähle `Person`, `LEGOSet`, etc.
  - Mit Rechtsklick auf Klassen: „Add children“, „Add neighbors“ aktivieren

📘 So baust du Schritt für Schritt ein visuelles Netz aus Klassen und ihren Beziehungen auf

⚠️ Achtung: OntoGraf zeigt keine Instanzen, nur Struktur (Schema)

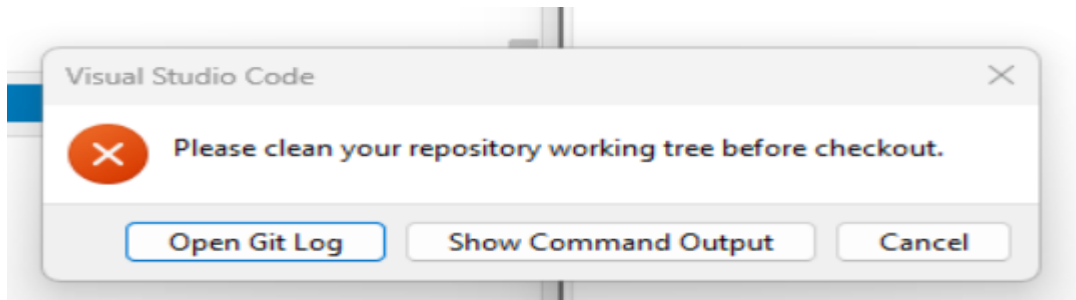
### ✓ 2. In Protégé: Individuals + Reasoner

Zeigt: Instanzen & inferierte Verbindungen (ABox)

So geht's:

1. `Window > Tabs > Individuals by Class`
2. Reasoner starten: `Reasoner > SWRLAPI (oder Pellet)`
3. Wähle z.B. `testKunde` → in der rechten Spalte siehst du:
  - seine Klassen
  - Object Properties → z.B. `empfehlSet` → `LEGOSetX`

### ChatGPT Prompt 3: «Was muss ich in Visual Studio Code tun?»



Die Fehlermeldung in Visual Studio Code lautet:

"Please clean your repository working tree before checkout."

Das bedeutet: Du hast **uncommittete oder unstaged Änderungen** in deinem Arbeitsverzeichnis (working tree), und Git lässt dich deshalb **nicht auf einen anderen Branch wechseln (checkout)**.

#### So löst du das:

##### ✅ Möglichkeit 1: Änderungen committen

Wenn du deine Änderungen behalten willst:

```
bash
git add .
git commit -m "Zwischenspeicher"
```

Kopieren Bearbeiten

Dann kannst du den Branch wechseln.

##### ✅ Möglichkeit 2: Änderungen stashen (zwischenlagern)

Wenn du deine Änderungen behalten, aber (noch) nicht committen willst:

```
bash
git stash
```

Kopieren Bearbeiten

Dann kannst du `git checkout <branchname>` machen.

Später mit `git stash pop` wiederherstellen.

##### ⚠️ Möglichkeit 3: Änderungen verwerfen

Wenn du **sicher** bist, dass du deine Änderungen nicht brauchst:

```
bash
git reset --hard
```

Kopieren Bearbeiten

Achtung: Damit gehen **nicht gespeicherte Änderungen** verloren.

## ChatGPT Prompt 4: «Was bedeutet diese Fehlermeldung im Protégé?

Exception running SQWRL query Set5\_Eigenkreation\_Experte: error running SQWRL queries: invalid query name Set5\_Eigenkreation\_Experte: invalid query name Set5\_Eigenkreation\_Experte»

Danke für die Rückmeldung – der Fehler

"invalid query name Set5\_Eigenkreation\_Experte"

tritt in Protégé/SQWRL in der Regel aus einem der folgenden drei Gründe auf:

### ✓ **Checkliste zur Fehlerbehebung**

#### 1. ✗ **Sonderzeichen oder ungültige Zeichen**

- Auch wenn der Name "sauber" aussieht: Umlautzeichen, Bindestriche, Leerzeichen oder Sonderzeichen verursachen Probleme.
- **Lösung:** Benutze nur ASCII-Zeichen, am besten CamelCase:
  - ✓ Set5EigenkreationExperte
  - ✗ Set5\_Eigenkreation\_Experte
  - ✗ Set5-Eigenkreation-Experte

#### 2. ⚠ **Query nicht gespeichert oder aktiv**

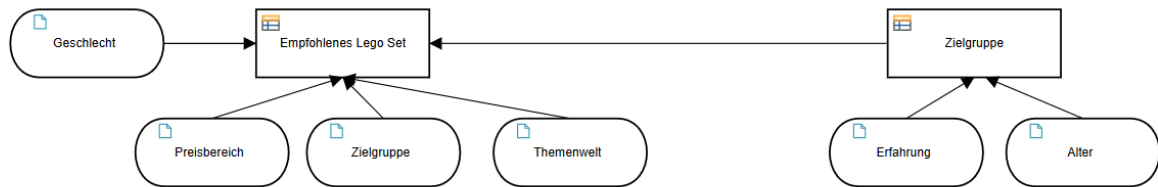
- Manchmal wird der Name nicht korrekt übernommen, wenn du ihn nur eintippst und dann gleich ausführst.
- **Lösung:**
  - Ändere den Namen (z. B. in Test123 ), drücke Enter.
  - Aktiviere die Checkbox links neben der Regel.
  - Klicke dann auf „Run“.

#### 3. 🗨 **Reasoner nicht aktiv**

- Ohne aktiven Reasoner (z. B. Pellet), läuft keine Regel.
- **Lösung:**
  - Menü oben → Reasoner > Pellet auswählen.
  - Dann → Reasoner > Start Reasoner .

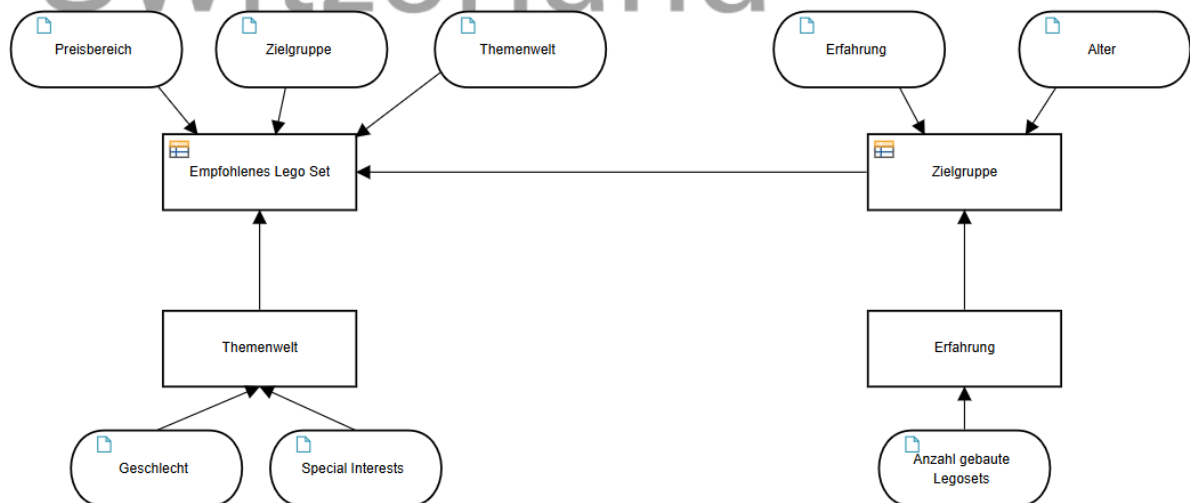
## Anhang 3: DMN Versionenübersicht mit iterativen Erläuterungen

### Version 1 – erste Modellüberlegungen



Im ersten Entwurf unseres Entscheidungsmodells wollten wir möglichst einfach darstellen, wie passende Lego-Sets empfohlen werden können. Die Entscheidung „Zielgruppe“ wurde dabei direkt aus den Faktoren Alter und Erfahrung abgeleitet. Die Produktempfehlung basierte auf Zielgruppe, Preisbereich, Themenwelt und Geschlecht. Dieses Modell war funktional, aber in vielen Bereichen noch sehr direkt aufgebaut – es gab wenig Zwischenschritte oder Herleitungen, wie genau die Entscheidungsinputs zustande kommen.

### Version 2 – erste Weiterentwicklung

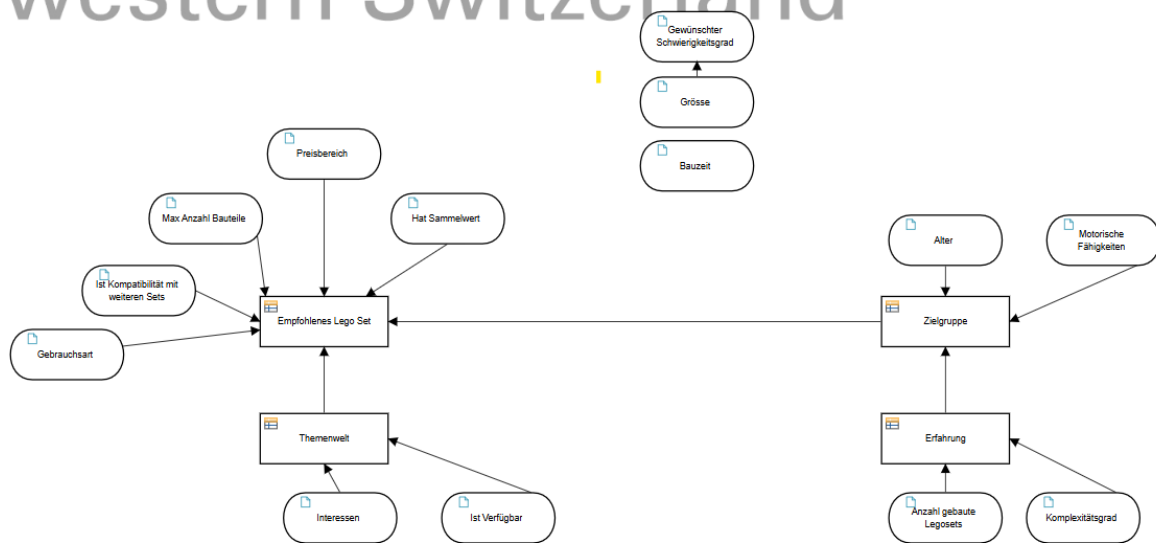


Wir haben daraufhin begonnen, unser Modell weiter zu strukturieren. Erste Ideen waren, zusätzliche Entscheidungsfaktoren wie Interessen oder Sammelwert einzubauen. Ziel war es, die Empfehlung realistischer zu gestalten, ähnlich wie in einem Beratungsgespräch im Laden. Trotzdem war unser Modell zu diesem Zeitpunkt noch stark auf direkte Input-Output-Zuordnungen ausgelegt.



## Version 3

# nwestern Switzerland



Unsere Dozierenden gaben uns den Hinweis, mehr Subdecisions einzubauen und die Inputs besser herzuleiten. Besonders die Entscheidungen „Erfahrung“ und „Themenwelt“ sollten nicht direkt festgelegt, sondern logisch aus anderen Faktoren abgeleitet werden.

Daraufhin haben wir unser Modell überarbeitet. Die Entscheidung „Erfahrung“ wird nun aus der Anzahl gebauter Legosets und dem Komplexitätsgrad abgeleitet. Diese fliesst gemeinsam mit Alter und motorischen Fähigkeiten in die Zielgruppenentscheidung ein. Die Themenwelt wird neu aus den Interessen der Person und der Verfügbarkeit der Sets bestimmt. Auf das Kriterium Geschlecht haben wir bewusst verzichtet, da Interessen und Spielverhalten relevanter sind.

Zusätzlich haben wir weitere Entscheidungsfaktoren ergänzt, um eine möglichst differenzierte Produktempfehlung zu ermöglichen. Dazu gehören unter anderem: Gebrauchsart (z. B. Spielen, Ausstellen, Sammeln), Sammelwert, maximale Anzahl Bauteile, Kompatibilität mit anderen Sets sowie der Preisbereich. Auch der gewünschte Schwierigkeitsgrad wurde als zusätzliche Entscheidung aufgenommen, der sich aus der Bauzeit und der Grösse des Sets ableitet. Insgesamt ist ein strukturiertes und realistisches Entscheidungsmodell entstanden, das verschiedene Bedürfnisse und Interessen berücksichtigt und eine nachvollziehbare Produktempfehlung ermöglicht.

## Anhang 4: Coaching Protokoll vom 04. April 2025

Online (Teams): Emanuele Laurenzi, Elena Pineider, Fabia Holzer, Bastian Bühler, Kerem Akkaya

Coaching Protokoll:

- Einleitung anpassen - Aufgabe erklären, Business Perspektive: Was ist das Business? Welche Entscheidung wollen wir treffen?
- Hit Policies anpassen schauen ob andere angewandt werden können und sicherstellen, dass alle Inputs zu einem Output führen.
- Decision "Empfehlung" komplett löschen
- Dokumentation: Das Finale DMN und die Entscheidungen genauer erklären, nicht die verschiedenen Versionen. Jede Entscheidung einzeln erklären und warum es Sinn macht, die einzelnen Inputs erklären und jeweils die der Hit-Policy erläutern.
- Eventuell mehr Lego Sets beim Output einer Regel hinzufügen, dass es eine Auswahl von Sets Empfohlen wird.

## Anhang 5: Coaching Protokoll vom 30. April 2025

Online (Teams): Knut Hinkelmann, Elena Pineider, Fabia Holzer, Bastian Bühler, Kerem Akkaya

Coaching Protokoll:

- Ziel: Aufteilung der Prädikate zur besseren Lesbarkeit und Wartbarkeit
- Statt alles in ein Prädikat wie `lego_set_empfehlung/6` zu packen, lieber aufteilen
- Kunden als eigene Fakten modellieren, z. B.:
  - o `kunde(anna).`
  - o `zielgruppe(anna, erwachsene).`
  - o `interesse(anna, city).`
  - o `budget(anna, 50).`
- Empfehlung nicht direkt als Datenrelation, sondern über eine Regel mit Logik:
  - o `empfehle_set(Kunde, Set) :- ... prüft Interessen, Zielgruppe, Budget etc.`
- Vorteil: Anfrage wie `?- empfehle_set(anna, Set).` funktioniert direkt
- Empfehlung: Default-Fälle (z. B. wenn keine Kriterien zutreffen) durch alternative Regeln abfangen, nicht in einer überladenen Entscheidungstabelle

- Zielgruppen kann zu mehreren Produkten zugeordnet werden
- Die Hauptregel empfehle\_set soll alle Einzelregeln zusammenführen, z. B. Thema, Preis, Zielgruppe, Kompatibilität

## Anhang 6: Coaching Protokoll vom 21. Mai 2025

Online (Teams): Emanuele Laurenzi, Elena Pineider, Fabia Holzer, Bastian Bühler, Kerem Akkaya

Coaching Protokoll:

- Ziel: Funktionierende Inferenz durch korrekt befüllte Instanzen und funktionale Regeln sicherstellen
- Aktuell feuern keine Regeln, da notwendige Property-Werte in den Individuen fehlen
- Empfehlungen:
  - Regeln einzeln debuggen, jeweils passende Test-Individuen anlegen
  - Reasoner Pellet verwenden, da Hermit instabil ist
  - Bei jedem Individuum prüfen:
    - Sind alle Vorbedingungen der Regel erfüllt (z. B. Alter, Erfahrung, Interesse)?
    - Wenn ja: gelbe Kästchen als Resultat sichtbar? (→ dann Regel funktioniert)
- Beispielregel prüfen:
  - Regel S1: Wenn  $\text{Alter} > 12 \wedge < 18$  und  $\text{Erfahrung} = \text{fortgeschritten}$  → Zielgruppe = Jugendliche
  - Test-Instanz:
    - kunde(david)
    - alter(david, 15)
    - erfahrung(david, fortgeschritten)
    - → Erwartetes Resultat: zielgruppe(david, jugendliche) (inferred)
- Noch keine konkrete Produktempfehlung für Kunden modelliert
  - Regeln existieren bisher nur für Ableitungen wie Zielgruppe, Schwierigkeitsgrad, Themenwelt

- Empfehlung:
  - Finalregel ergänzen:
    - Kombination aus Preis, Zielgruppe, Interesse, Schwierigkeitsgrad → emp-  
fehle\_set(Kunde, Set)
- Hinweise zur Toolnutzung:
  - Reasoner testen (Pellet bevorzugt)
  - SWRL-Regeln einzeln prüfen
  - Screenshots für alle inferierten Fakten ins Dokument einfügen (inkl. OntoGraph, Regelansicht, Query-Resultate)