Write apps
in your own words

# Ontology and business rules in practice

7 October 2025 – Bas van der Raadt – VU Guest lecture

# Who is Bas van der Raadt

**Education & Research:**

- Computer Science (Bsc) at Hogeschool Utrecht
- Business Information Science (Msc) at Vrije Universiteit
- Enterprise Architecture (PhD) at Vrije Universiteit

**Practitioner:**

- Consultancy (Capgemini and Ernst & Young)
- Management (Schiphol Airport, ABN AMRO & eBay)

**Entrepeneur:**

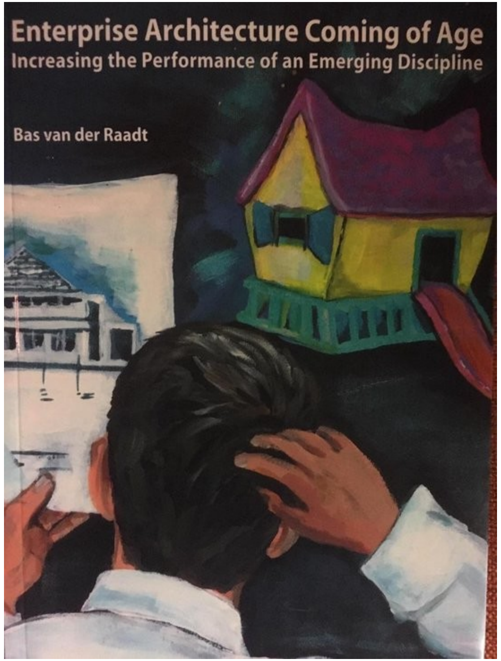
- Owner of Clear Advisory
- Founder of Hapsah.org

# Agenda part 1: “the basics”?

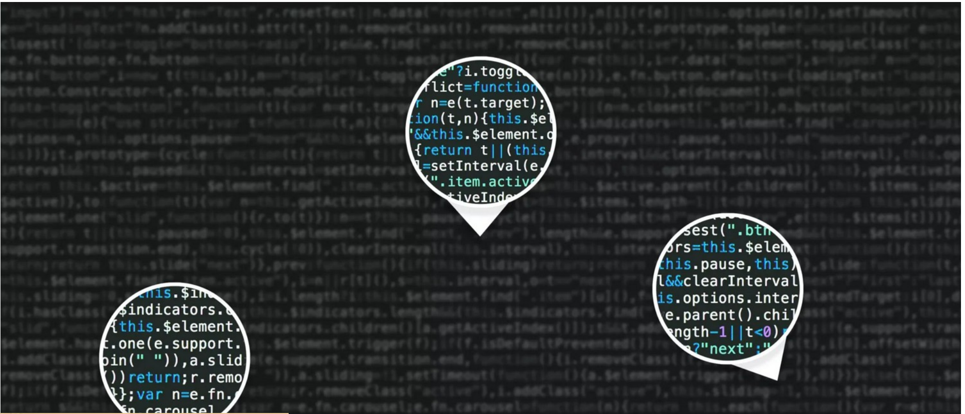
- Problems with code based systems
- Why ontology driven AI systems are needed
- What is an ontology? How to verbalize an ontology
- What is a business rule? How to verbalize business rules
- How to develop an ontology & business rules?

hapsah!

# Why Code based systems are not perfect


Enterprise Architecture Coming of Age
Increasing the Performance of an Emerging Discipline
Bas van der Raadt

Gap between design & implementation



```
e"?i.togg
flict=function
r n=e(t.target);
ion(t,n){this.$e
'&&this.$element.
{return t||(this
l=setInterval(e
(".item.activ
iveInde
```

```
sest(".btn
ors=this.$elem
this.pause,this
l&&clearInterval
is.options.inter
e.parent().chil
ngth-1||t<0)
?"next":"
```

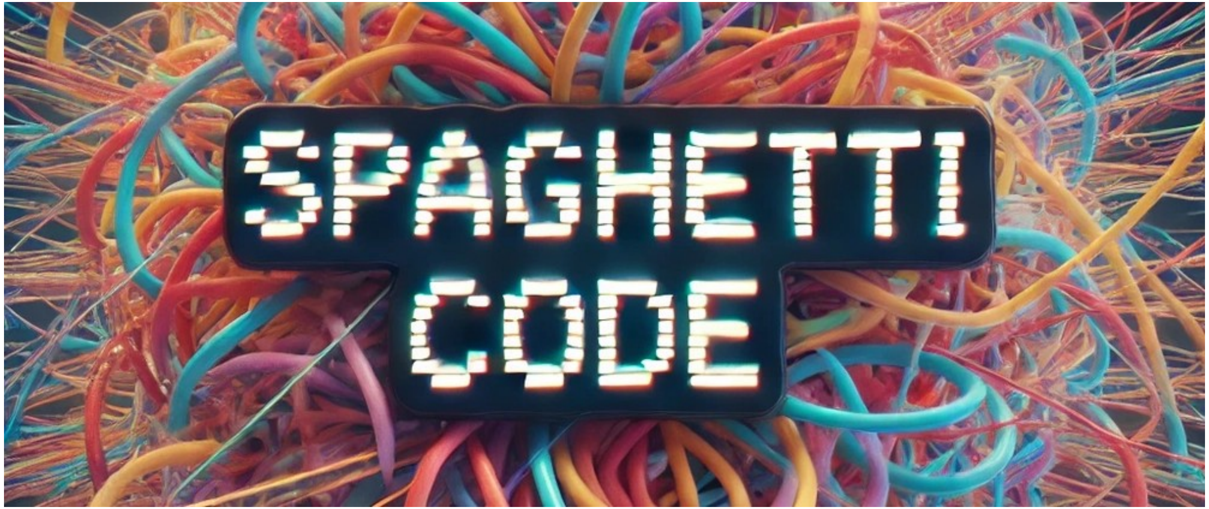
```
this.$in
$indicators.
(this.$element.
t.one(e.support.
oin(" "),a.slid
))return;r.remo
}};var n=e.fn.
n.carousel
```

Where are my business rules?

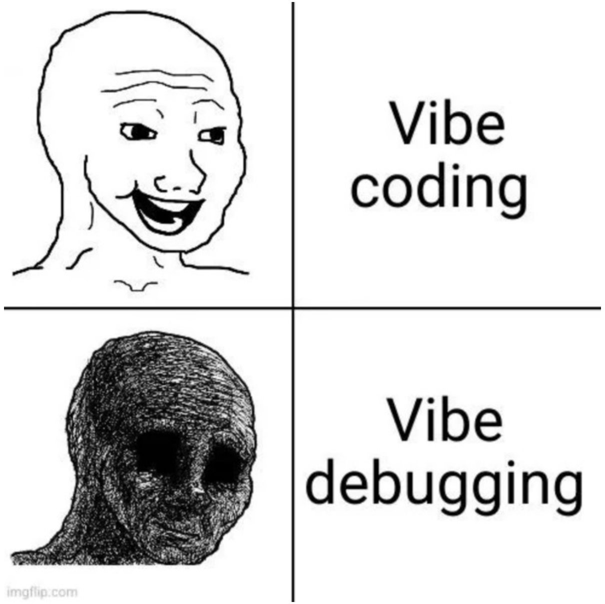
I’ve entered my data.
What happens next?

hapsah!

# Why No/Low-code & Vibe coding are not the answer



Generated code is not a pretty sight



You're on your own...

Vendor lock-in

hapsah!

# Why Ontology driven (LLM-free) AI systems are needed

Can you trust the system?

Does it respect the environment?

Can you trace its decisions?
Does it respect your privacy?

hapsah!

# What is Natural Language Execution?

A form of AI that delivers **directly executable**, **deterministic** decision-making systems, based on (controlled) **natural language** input.

1. **Natural Language**
   Specification of ontology and business rules
2. **Deterministic**
   Fully predictable and reliable outcomes
3. **Directly executable & changeable**
   Without code generation and data migrations

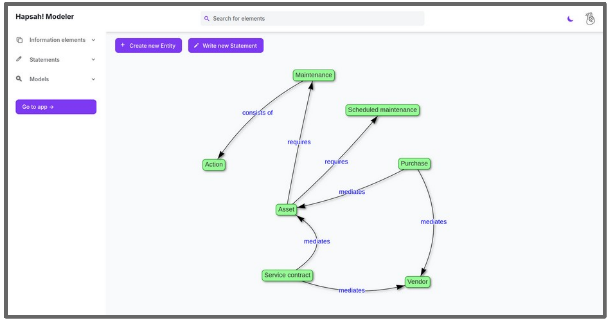
# What is Hapsah?

**Hapsah** stands for:

**H**uman
**A**imed
**P**latform for
**S**emantic
**A**pplication
**H**osting

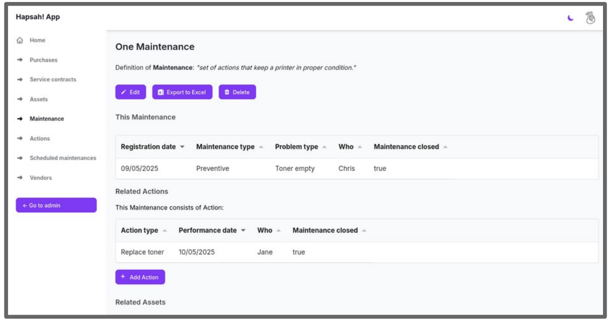
{ REST:API }
Ontology &
Business rules

## Modeler



Changes
are directly
available

## Runtime



One Maintenance

Definition of **Maintenance**: *"set of actions that keep a printer in proper condition."*

Edit | Export to Excel | Delete

This Maintenance

| Registration date | Maintenance type | Problem type | Who | Maintenance closed |
|---|---|---|---|---|
| 09/05/2025 | Preventive | Toner empty | Chris | true |

Related Actions

This Maintenance consists of Action:

| Action type | Performance date | Who | Maintenance closed |
|---|---|---|---|
| Replace toner | 10/05/2025 | Jane | true |

+ Add Action

Related Assets

{ REST:API }
App data

Commands

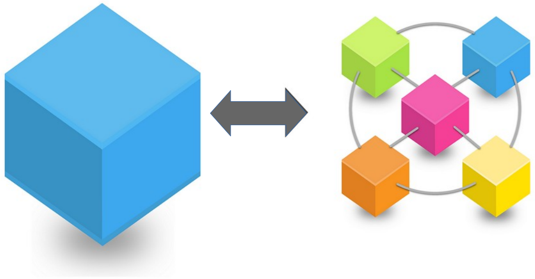
Events

hapsah!

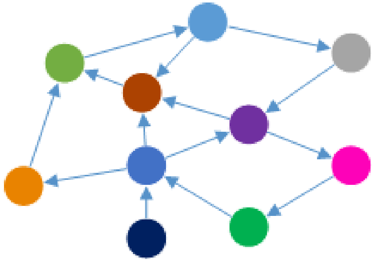
# What is Hapsah?

Monolithic

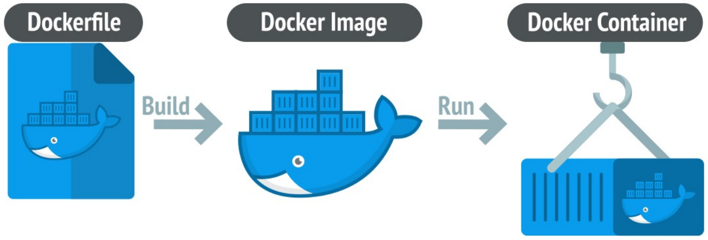
Microservices



Graph Database



Processing engine



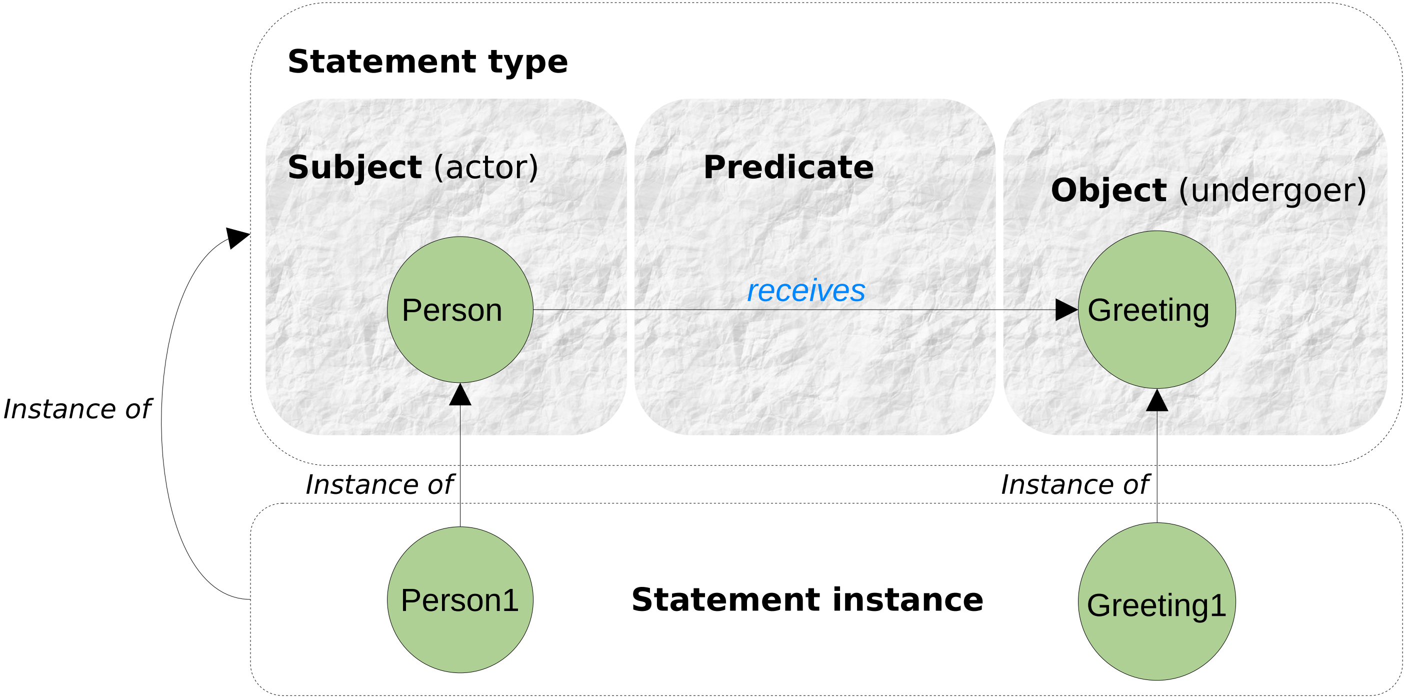
# What is the ontology language?

| Nouns (types) | Descriptions | Examples |
|---|---|---|
| Entity | • Something that exists either objectively or conceptually<br>• Has a unique identity<br>• Capable of being changed continuously (being able to hold different states) over a long period of time | Person |
| Descriptor | • Integral and immutable unit of attributes<br>• Measures, quantifies or describes a thing (e.g. entity), and has no distinct identity | Full name |
| Attribute | • Defines a property type that:<br>• May be attributed to an entity type or<br>• May be part of a descriptor type | First name |

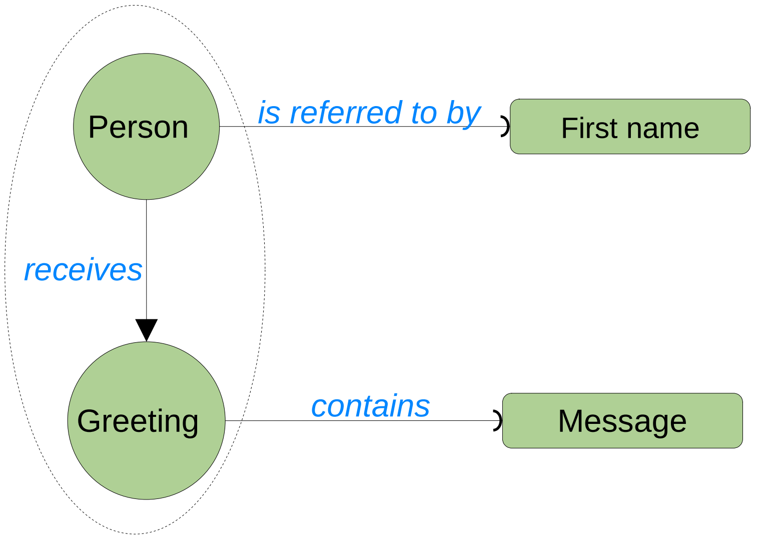
# What is the ontology language?

| Verbs (types) | Descriptions | Examples |
|---|---|---|
| *Association*<br>——▶ | • Designates a meaningful relationship between two entities<br>• Of different or the same entity types | *owns* |
| *Attribution*<br>——) | • Is the link between a subject (entity type) and a target (composite) attribute type | *is referred to by* |
| *Composition*<br>——◆ | • Indicates that a constructed type can be built from smaller, atomic types | *consists of* |
| *Extension*<br>——▷ | • Indicates that each instance of the subject entity type (subtype) is also an instance of the object entity type (base type)<br>• Subject entity type inherits all characteristics of that base or super type | *is a* |

# What is the ontology language?



# How does an ontology work in practice?



# What are action rules?

**Action rules** specify which actions need to be executed under which conditions

| Parts | Descriptions | Options |
|---|---|---|
| When | • Conditions that determine when a rule is triggered for execution | • Association condition<br>• Attribution condition<br>• Equation condition |
| Do | • Operations that manipulate values<br>• Results of operations are not stored by default | • Unary operation<br>• Binary operation |
| Then | • Actions that are executed when a rule is fired<br>• Actions may result in date being created or changed | • Find entity action<br>• Create element action<br>• Make association action<br>• Assign (value or descriptor) action |

# How does an action rule work in practice?

When:

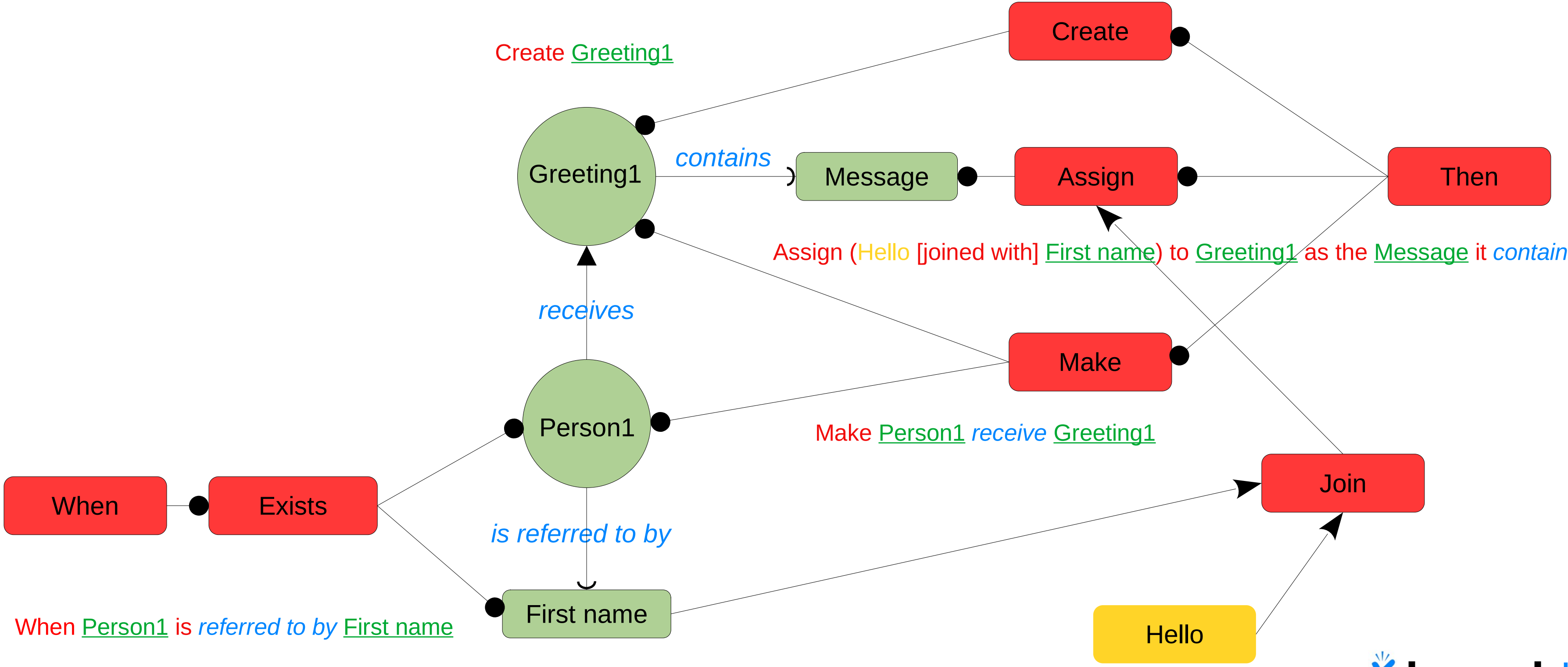
- Person1 is *referred to by* First name

Then:

- Create Greeting1
- Assign (Hello [joined with] First name) to Greeting1 as the Message it *contains*
- Make Person1 *receive* Greeting1

*The syntax and color coding are inspired by the OMG standard:*
*Semantics Of Business Vocabulary And Business Rules (SBVR)*

# What is the link between ontology and rules?



Create Greeting1

Assign (Hello [joined with] First name) to Greeting1 as the Message it *contains*

Make Person1 *receive* Greeting1

When Person1 is *referred to by* First name

hapsah!

# What are constraints?

**Constraints** specify which actions may not be executed under which conditions

| Parts | Descriptions | Options |
|---|---|---|
| When | Conditions that determine when a constraint needs to act | Association condition<br>Attribution condition<br>Equation condition |
| Do | Operations that manipulate values<br>Results of operations are not stored by default | Unary operation<br>Binary operation |
| Then | Actions that need to be blocked when constraint is activated | Create element action<br>Make association action<br>Assign (value or descriptor) action |

# How does a constraint work in practice?

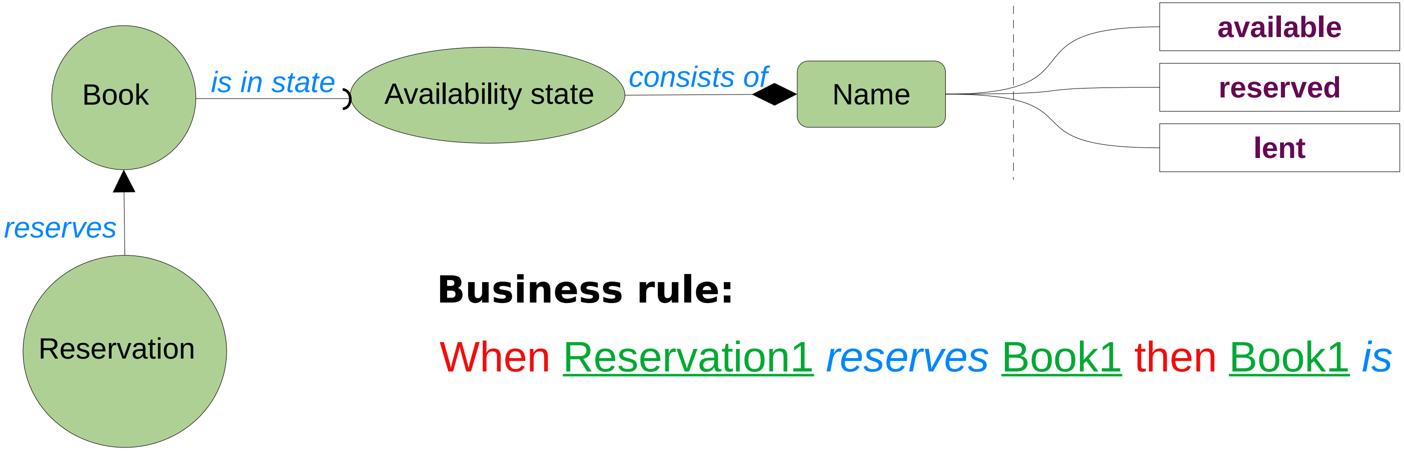
When Greeting1 does not *contain* a Message

Then Person1 may not *receive* Greeting1

# What is state modelling?

## Ontology (types):

## Values (instances):



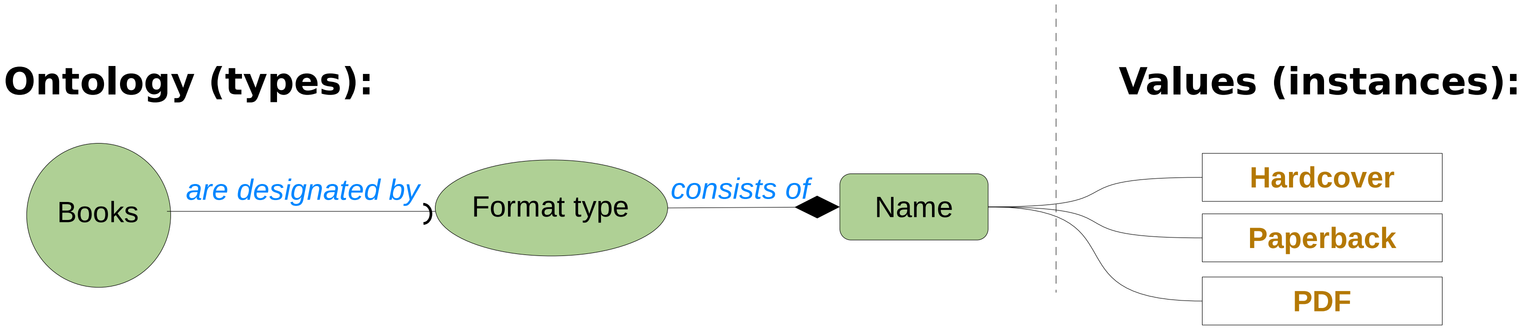
## Business rule:

When Reservation1 *reserves* Book1 then Book1 *is in state* **reserved**

| Attribution type | Description | Label |
|---|---|---|
| **State indication** | reside in a state (condition or stage) in the physical or conceptual being of something or someone | *is in state* |

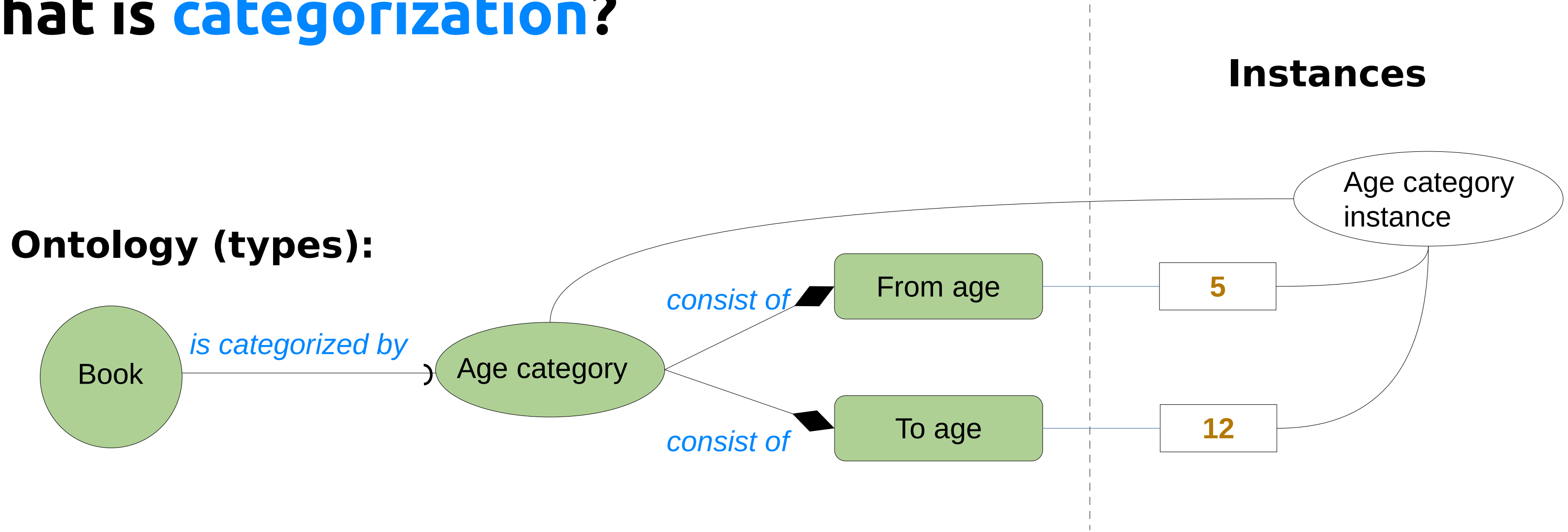
hapsah!

# What is classification? (Higher-level types)



| Attribution type | Description | Label |
|---|---|---|
| **Classification** | • Systematic arrangement based on shared properties<br>• Membership of item is determined by its characteristics<br>• An entity can be a member of one set designated by a type | *is designated by* |

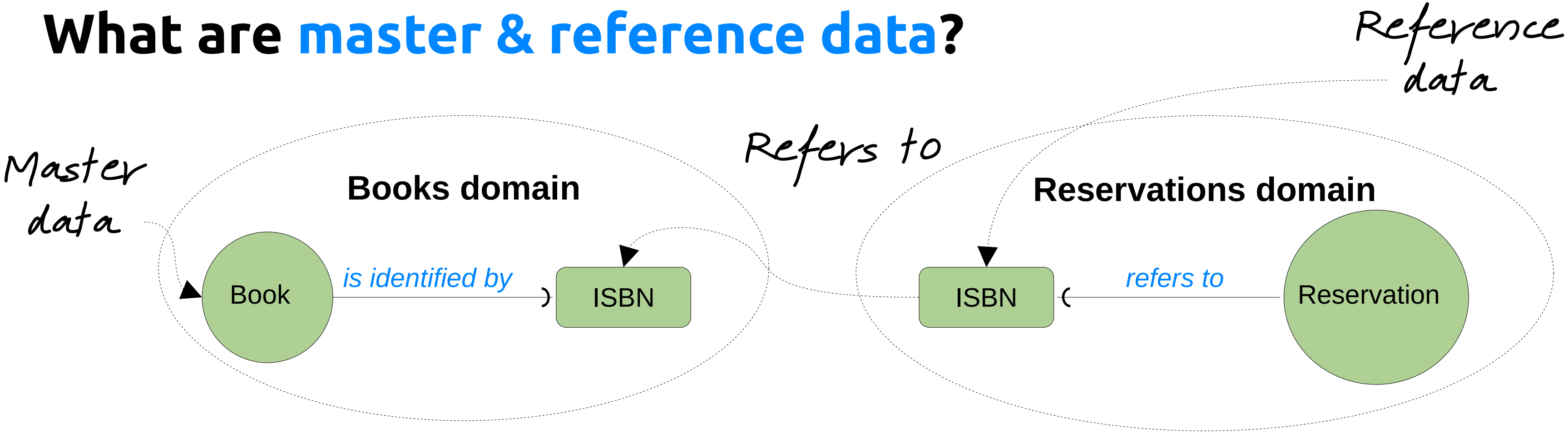
# What is categorization?



| Attribution type | Description | Label |
|---|---|---|
| **Categorization** | • Placing items based on distinguishing features<br>• Membership of item may change when definition of category changes<br>• An entity can be put into one or more categories | *is categorized by* |

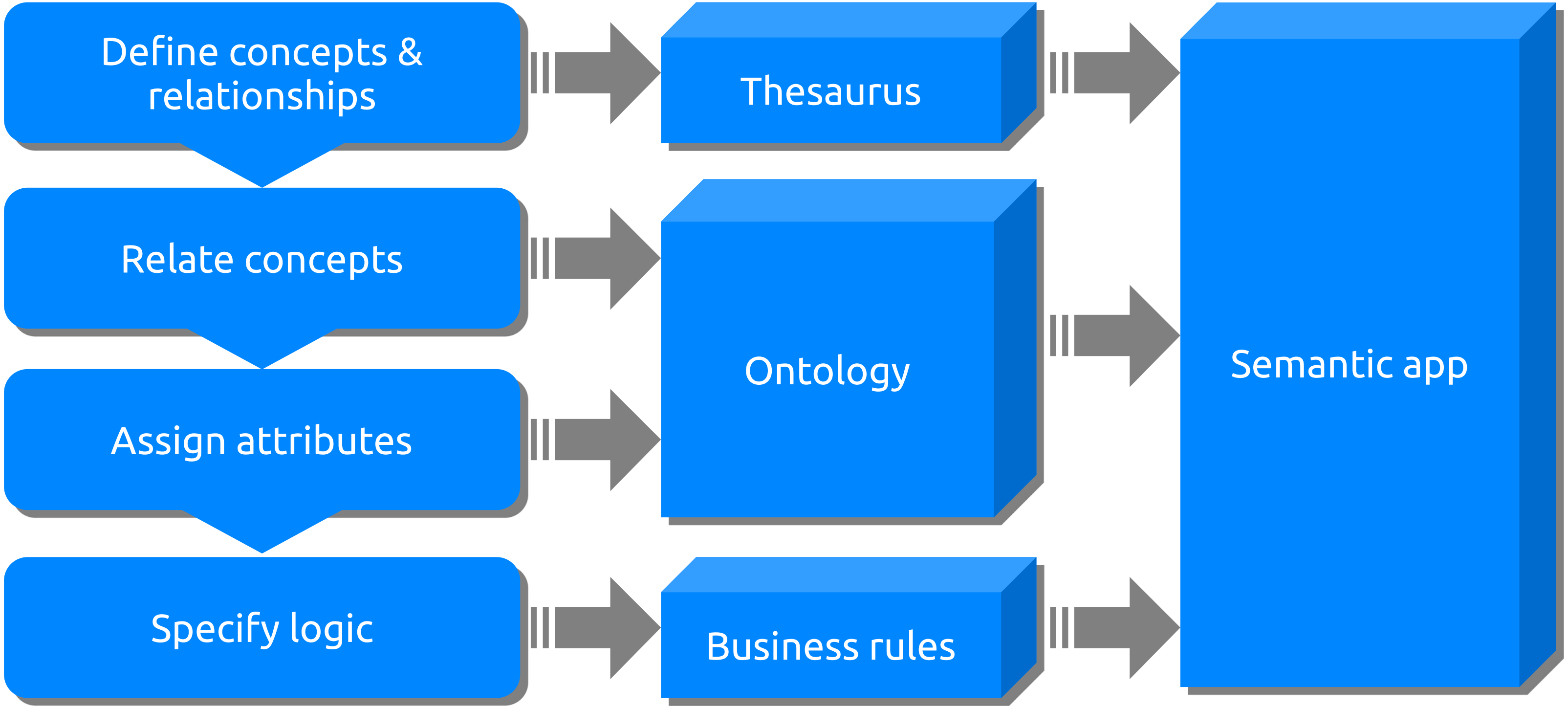
hapsah!

# What are master & reference data?



| Attribution types | Descriptions | Labels |
|---|---|---|
| **Identification** | be established or recognized as a certain person or thing by a specific characteristic | *is identified by* |
| **Reference** | An entity can mention or allude to another entity by referring to its unique identifier | *refers to* |

# How to develop an ontology & business rules?



Define concepts & relationships

Relate concepts

Assign attributes

Specify logic

Thesaurus

Ontology

Business rules

Semantic app

hapsah!

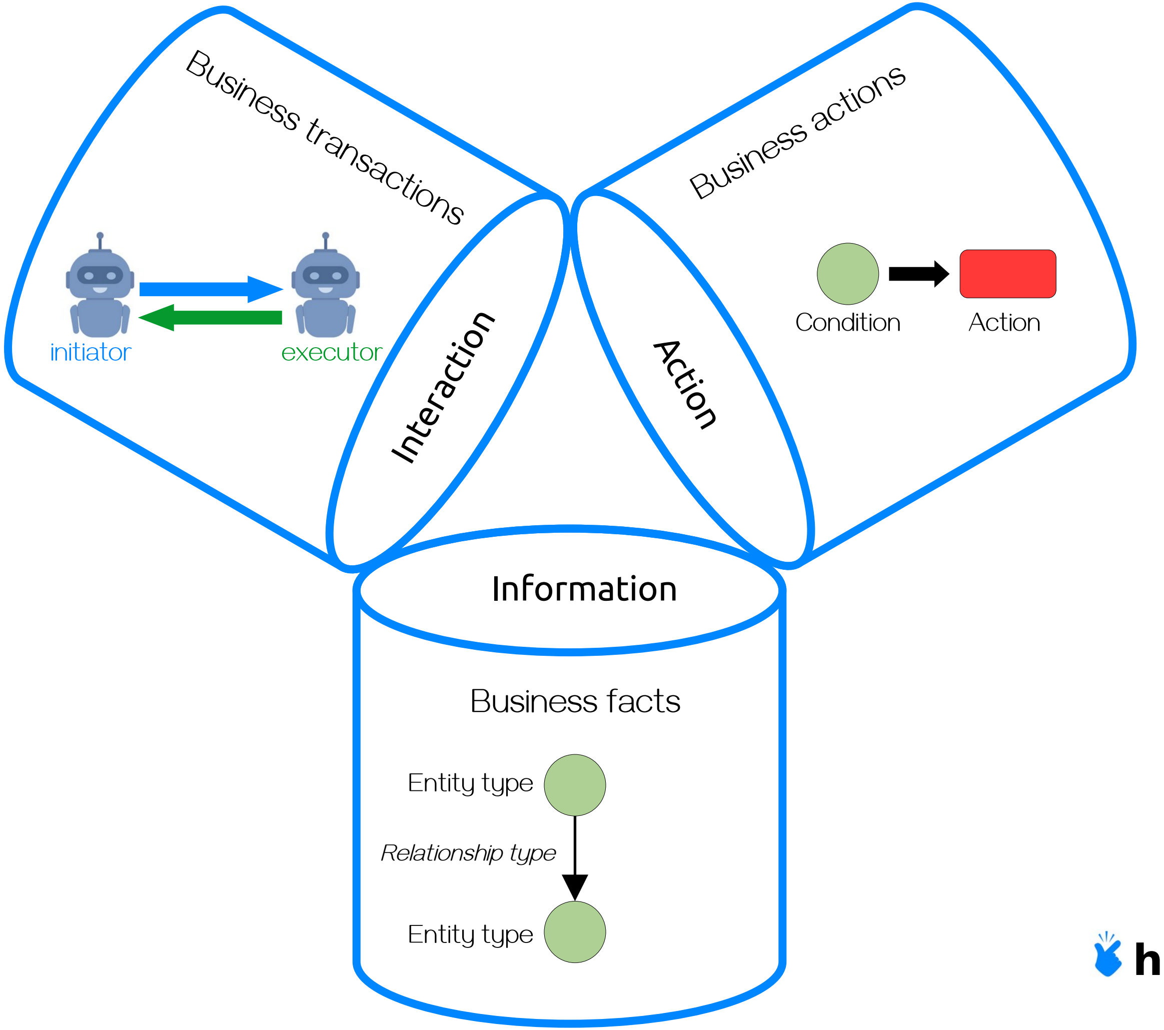
# What did we cover in part 1: “the basics”?

- Problems with code based systems
- Why ontology driven AI systems are needed
- What is an ontology? How to verbalize an ontology
- What is a business rule? How to verbalize business rules
- How to develop an ontology & business rules?

# Agenda part 2: “Taking the next step”

- Ontology goes beyond just information
- Different ontology languages:
  - DEMO
  - OntoUML
  - i*
  - e3-value
- How to combine those languages into one modeling approach:
  - Case study: 3D printer maintenance
- Questions & Answers

# Ontology is not only about information



hapsah!

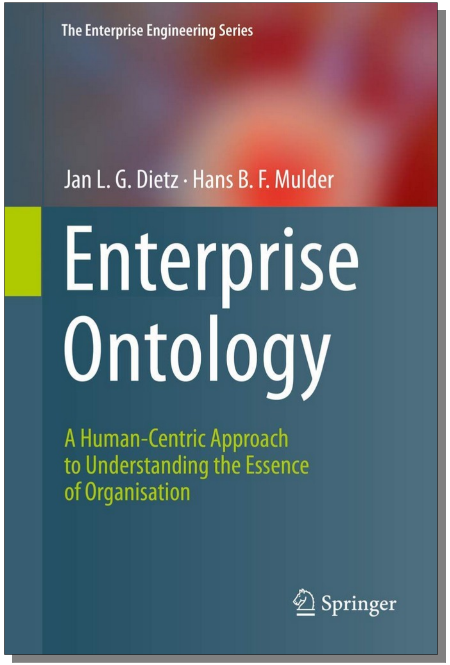
# Different ontology languages: DEMO / Enterprise Ontology

**Purpose:** Model the essence of an organization

**Scope:** Process model, Fact (information) model, Action model

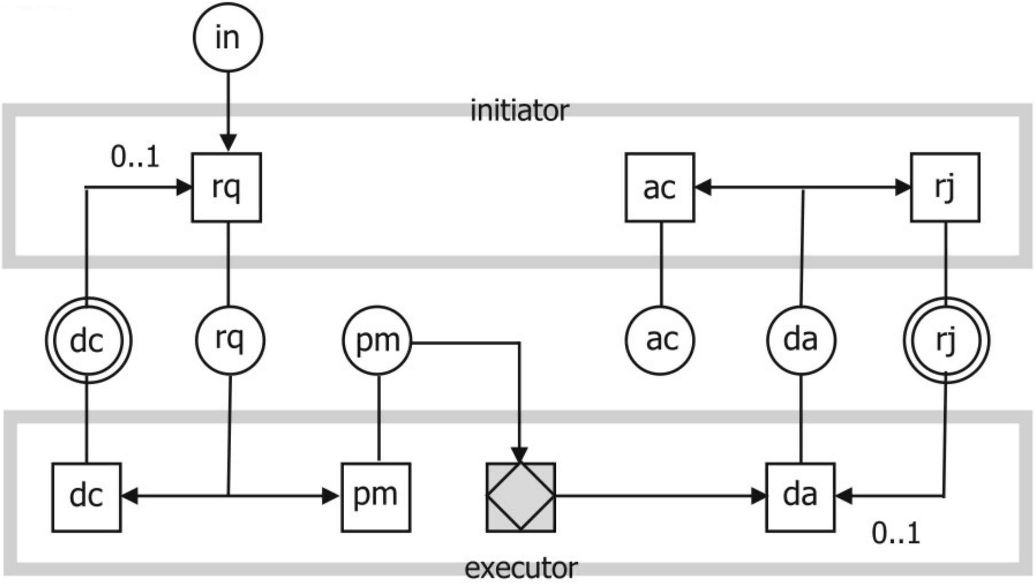
**Key concepts of Process model:**

- **Transaction:** The building block of business processes.
- **Product (result):** The result of a successfully carried out transaction is the coming into existence of a product.
- **Initiator:** The role authorised and responsible for initiating the transaction
- **Executor:** The role authorised and responsible for executing the transaction


The Enterprise Engineering Series
Jan L. G. Dietz · Hans B. F. Mulder
Enterprise Ontology
A Human-Centric Approach to Understanding the Essence of Organisation
Springer

# Different ontology languages: DEMO / Enterprise Ontology

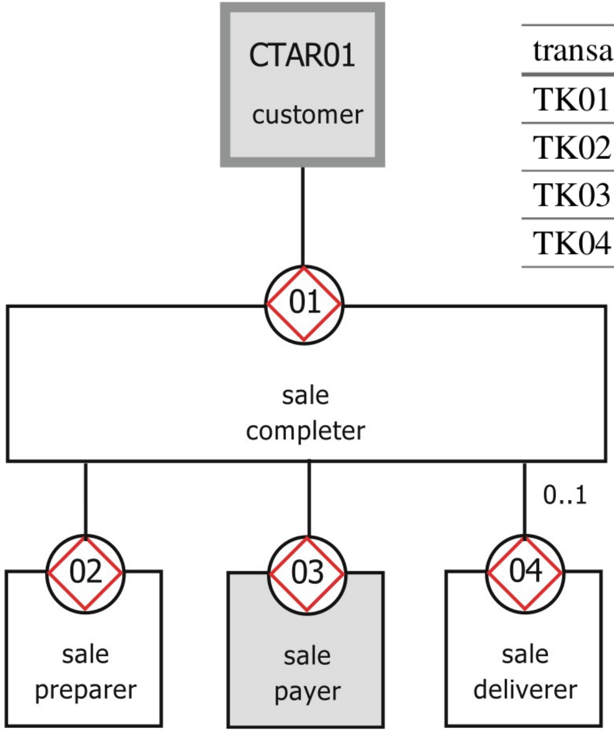
## Standard transaction pattern:



## Possible states:

- **in:** Initial state
- **rq:** Request(ed)
- **dc:** Decline(d)
- **pm:** promise(d)
- **da:** declare(d)
- **rj:** Reject(ed)

# Different ontology languages: DEMO / Enterprise Ontology



| transaction kind | product kind | executor role |
|---|---|---|
| TK01 sale completing | PK01 [sale] is completed | AR01 sale completer |
| TK02 sale preparing | PK02 [sale] is prepared | AR02 sale preparer |
| TK03 sale paying | PK03 [sale] is paid | AR03 sale payer |
| TK04 sale delivering | PK04 [sale] is delivered | AR04 sale deliverer |

hapsah!

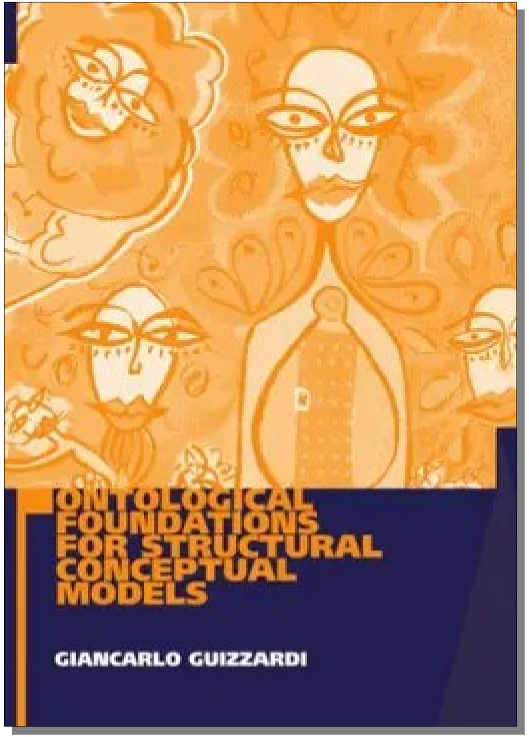
# Different ontology languages: OntoUML

**Purpose:** Language for ontology-driven conceptual modeling

**Scope:** Information modeling

**Key characteristics:**

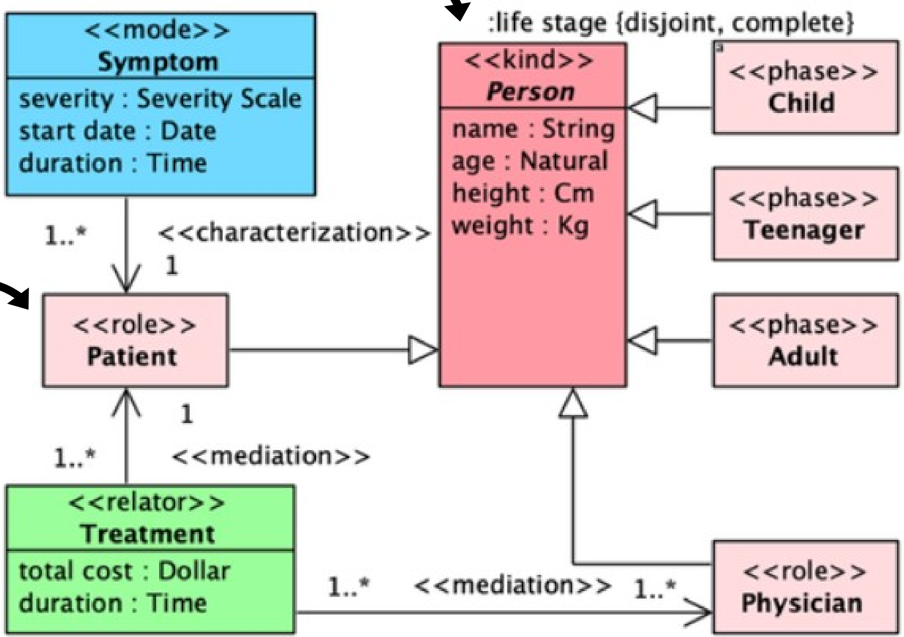
- Extension to UML class diagrams
- Extra ontological meaning based on <<stereotypes>> for classes and associations
- Provides design patterns including syntactical checking
- Based on Unified Formal Ontology (UFO)



# Different ontology languages: OntoUML

**Sortal type:**
*provides identity*

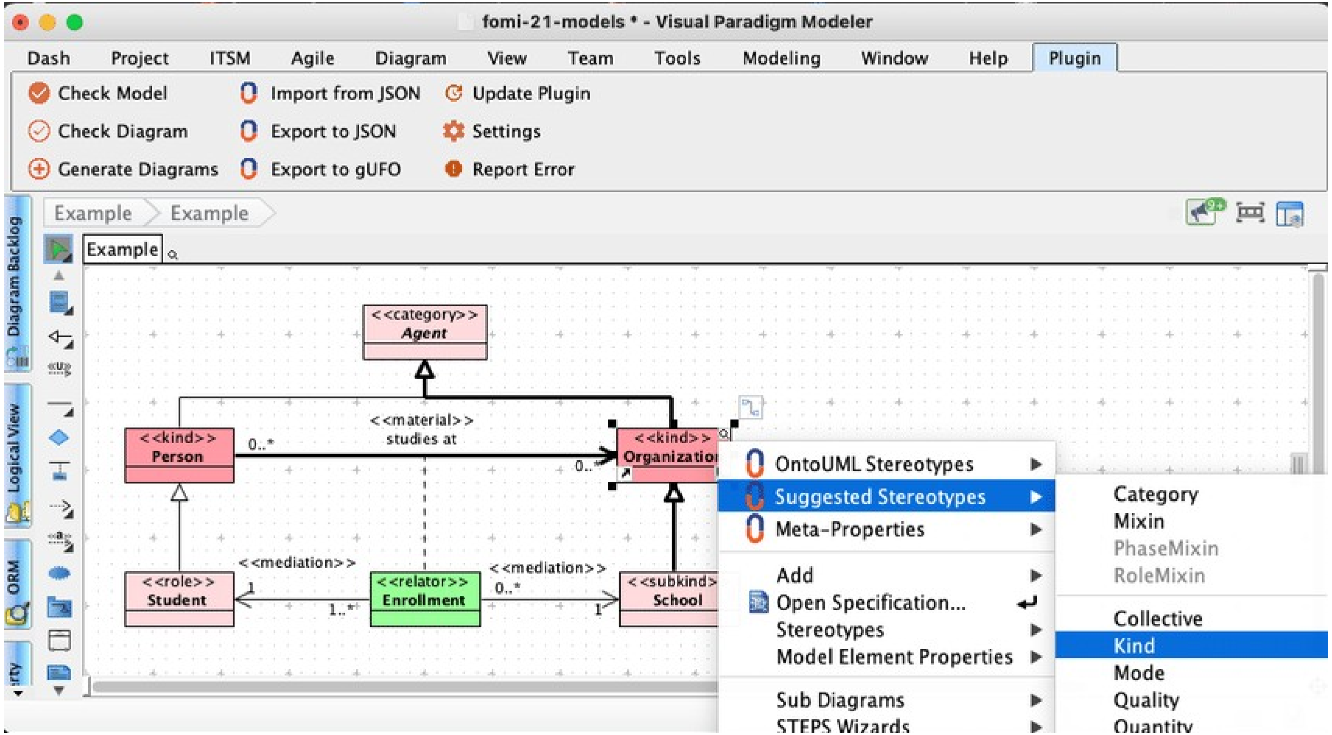
**Rigid type:**
*always exists during its entire lifecycle*



**Non-sortal type**

**Anti-rigid types**

# Different ontology languages: OntoUML



hapsah!

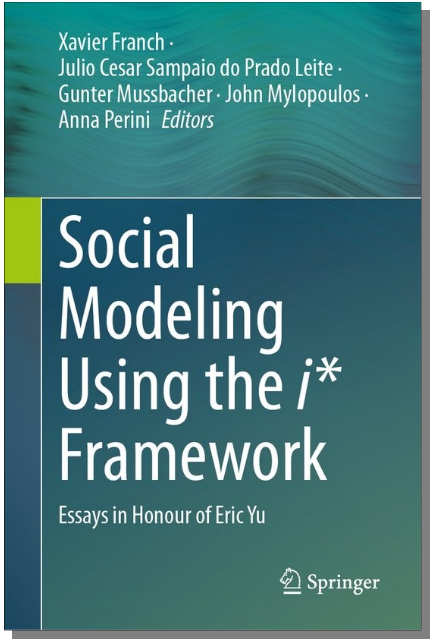
# Different ontology languages: i*

**Purpose:** Agent-oriented social modeling

**Scope:** Strategic relationships in social systems

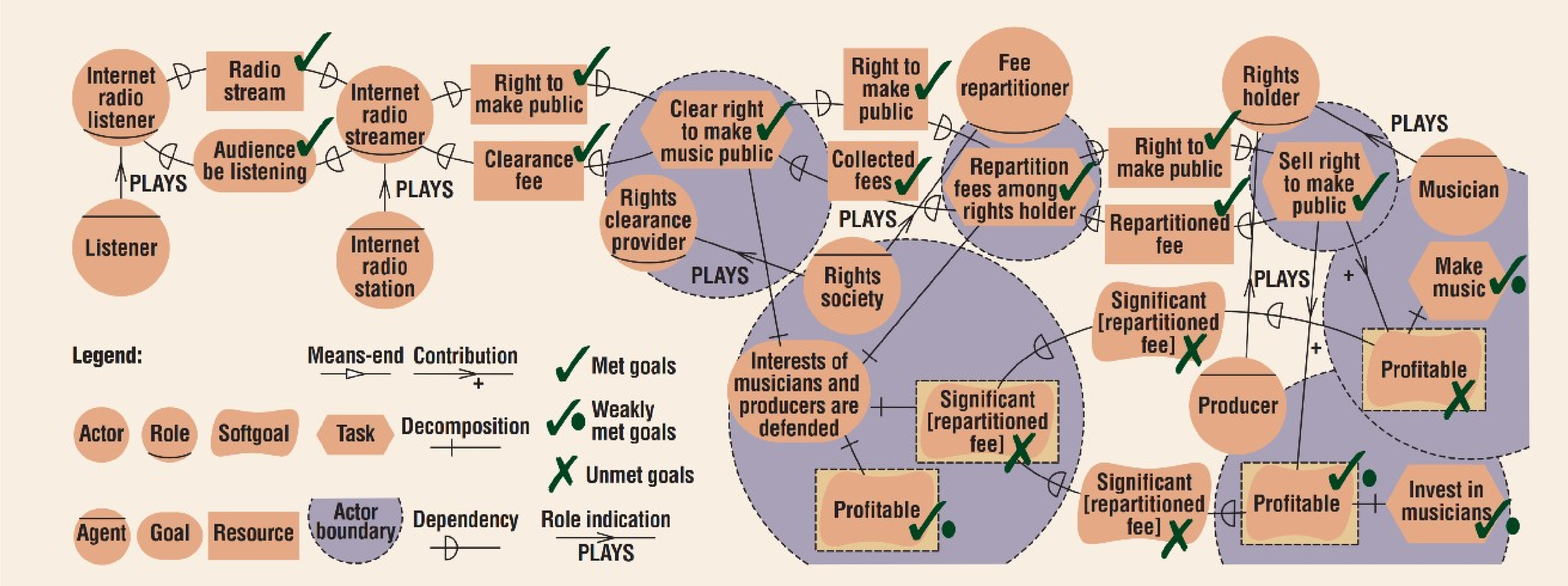
**Key concepts:**

- **Agent:** social actor with concrete manifestations
- **Role:** abstract characterization of behavior of a social actor
- **Goal:** state of affairs that the actor wants to achieve
- **Quality (Softgoal):** attribute of something for which an actor desires some level of achievement
- **Resource:** physical or informational entity that the actor requires in order to perform a task
- **Task:** action that an actor wants to be executed, usually with the purpose of achieving some goal


Xavier Franch ·
Julio Cesar Sampaio do Prado Leite ·
Gunter Mussbacher · John Mylopoulos ·
Anna Perini Editors
Social Modeling Using the i* Framework
Essays in Honour of Eric Yu
Springer

hapsah!

# Different ontology languages: i*



hapsah!

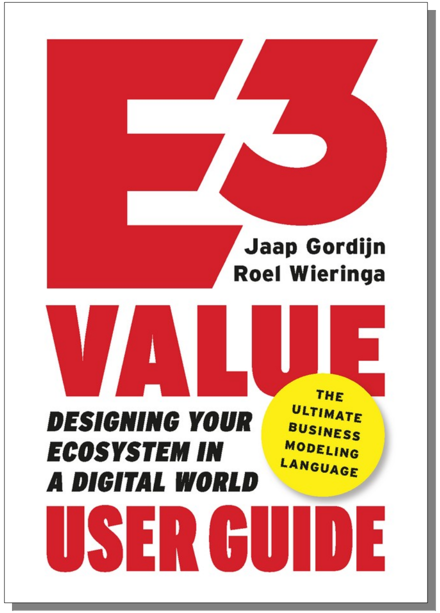
# Different ontology languages: e3-value

**Purpose:** Value modeling for business ecosystems

**Scope:** Business transactions in value networks

**Key concepts:**

- Actor: entity that is responsible for its survival and well-being
- **Value object:** object that is of economic value for at least one other actor in the network
- **Value activity:** task performed by an actor that potentially results in a benefit for the actor
- **Value transaction:** two or more actors exchange value objects to satisfy a need



E3 VALUE

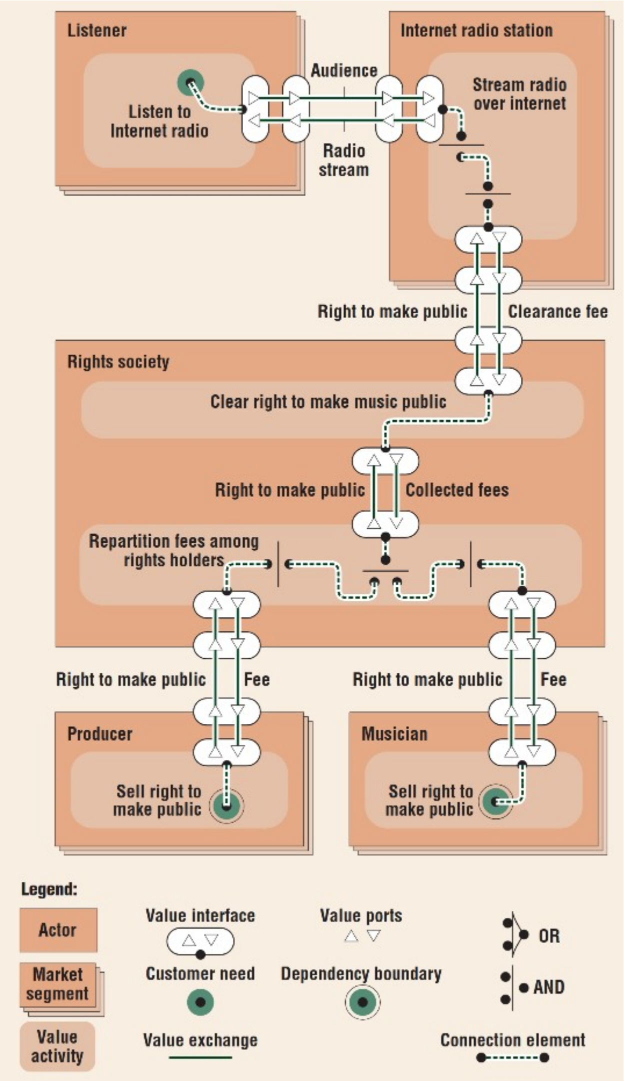
Jaap Gordijn
Roel Wieringa

DESIGNING YOUR ECOSYSTEM IN A DIGITAL WORLD

THE ULTIMATE BUSINESS MODELING LANGUAGE

USER GUIDE

# Different ontology languages: e3-value



hapsah!

# How to combine those languages into one approach

## Revisiting the DEMO Transaction Pattern with the Unified Foundational Ontology (UFO)

Tanja Poletaeva[1], Giancarlo Guizzardi[2,3], João Paulo A. Almeida[3], Habib Abdulrab[1]

[1] LITIS lab., INSA de Rouen, Rouen, France
ta.poletaeva@gmail.com, habib.abdulrab@insa-rouen.fr
[2] Free University of Bozen-Bolzano, Italy
[3] Ontology and Conceptual Modeling Research Group (NEMO), Federal University of Espírito Santo, Vitória, ES, Brazil
{gguizzardi, jpalmeida}@inf.ufes.br

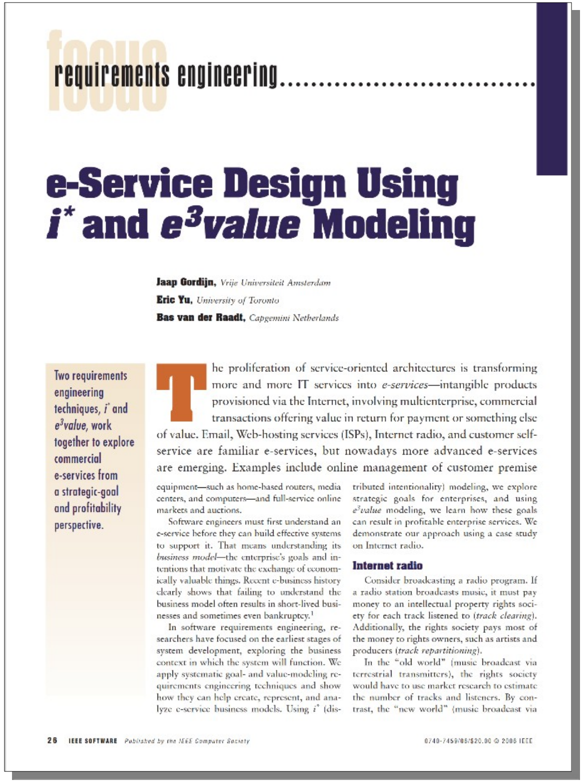
**Abstract.** In this paper, we revisit the DEMO transaction pattern in light of the domain-independent system of categories put forth by the Unified Foundational Ontology (UFO). In this process, we treat social relationships in the scope of the DEMO transactions as objectified social entities, and thereby separate the behavioural and structural aspects of the transaction pattern and clarify their interplay. Further, we represent the pattern in the OntoUML ontology-driven conceptual modeling language. The revisited pattern can be embedded in broader enterprise ontologies and reference conceptual models based in UML. The proposed OntoUML models can also be further refined to account for and consider different organizational implementations of business transactions. We demonstrate the proposed representation by applying it to OMGs EU-Rent case.

**Keywords:** Foundational ontology, enterprise ontology, DEMO transaction pattern, organizational implementation.

### 1 Introduction

Since 1960s, conceptual modeling is widely adopted for knowledge communication among human users [1]. The importance of enterprise conceptual modeling in enterprise engineering and transformation [2] has encouraged the development of various enterprise modeling methods. Nowadays, there is a growing interest in approaches that employ ontologies as theoretical tools for improving conceptual models. Among such approaches, there is a mature DEMO methodology (the Design and Engineering Methodology for Organizations) [3], which comprises the DEMO enterprise ontology, the ontology-based enterprise modeling language, and the modeling method.

Despite the conceptual quality of DEMO, we observe that there are still opportunities for clarification and generalization of its conceptual basis, in particular considering some aspects of social relationships that evolve in business transactions. In addition to that, there are little guidelines on how to integrate knowledge conceptualized with DEMO to other (non-DEMO based) organizational conceptual models that are widely employed in practice (such as, e.g., reference organizational models captured



focus

requirements engineering

## e-Service Design Using $i^*$ and $e^3value$ Modeling

**Jaap Gordijn,** *Vrije Universiteit Amsterdam*
**Eric Yu,** *University of Toronto*
**Bas van der Raadt,** *Capgemini Netherlands*

Two requirements engineering techniques, $i^*$ and $e^3value$, work together to explore commercial e-services from a strategic-goal and profitability perspective.

The proliferation of service-oriented architectures is transforming more and more IT services into *e-services*—intangible products provisioned via the Internet, involving multienterprise, commercial transactions offering value in return for payment or something else of value. Email, Web-hosting services (ISPs), Internet radio, and customer self-service are familiar e-services, but nowadays more advanced e-services are emerging. Examples include online management of customer premise equipment—such as home-based routers, media centers, and computers—and full-service online markets and auctions.

Software engineers must first understand an e-service before they can build effective systems to support it. That means understanding its *business model*—the enterprise's goals and intentions that motivate the exchange of economically valuable things. Recent e-business history clearly shows that failing to understand the business model often results in short-lived businesses and sometimes even bankruptcy.[1]

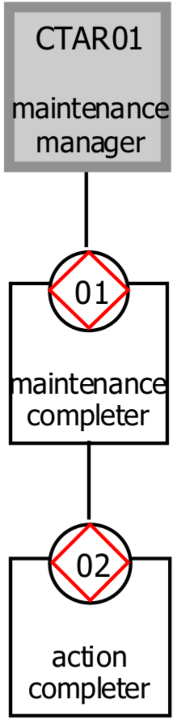
In software requirements engineering, researchers have focused on the earliest stages of system development, exploring the business context in which the system will function. We apply systematic goal- and value-modeling requirements engineering techniques and show how they can help create, represent, and analyze e-service business models. Using $i^*$ (distributed intentionality) modeling, we explore strategic goals for enterprises, and using $e^3value$ modeling, we learn how these goals can result in profitable enterprise services. We demonstrate our approach using a case study on Internet radio.

### Internet radio

Consider broadcasting a radio program. If a radio station broadcasts music, it must pay money to an intellectual property rights society for each track listened to (*track clearing*). Additionally, the rights society pays most of the money to rights owners, such as artists and producers (*track repartitioning*).

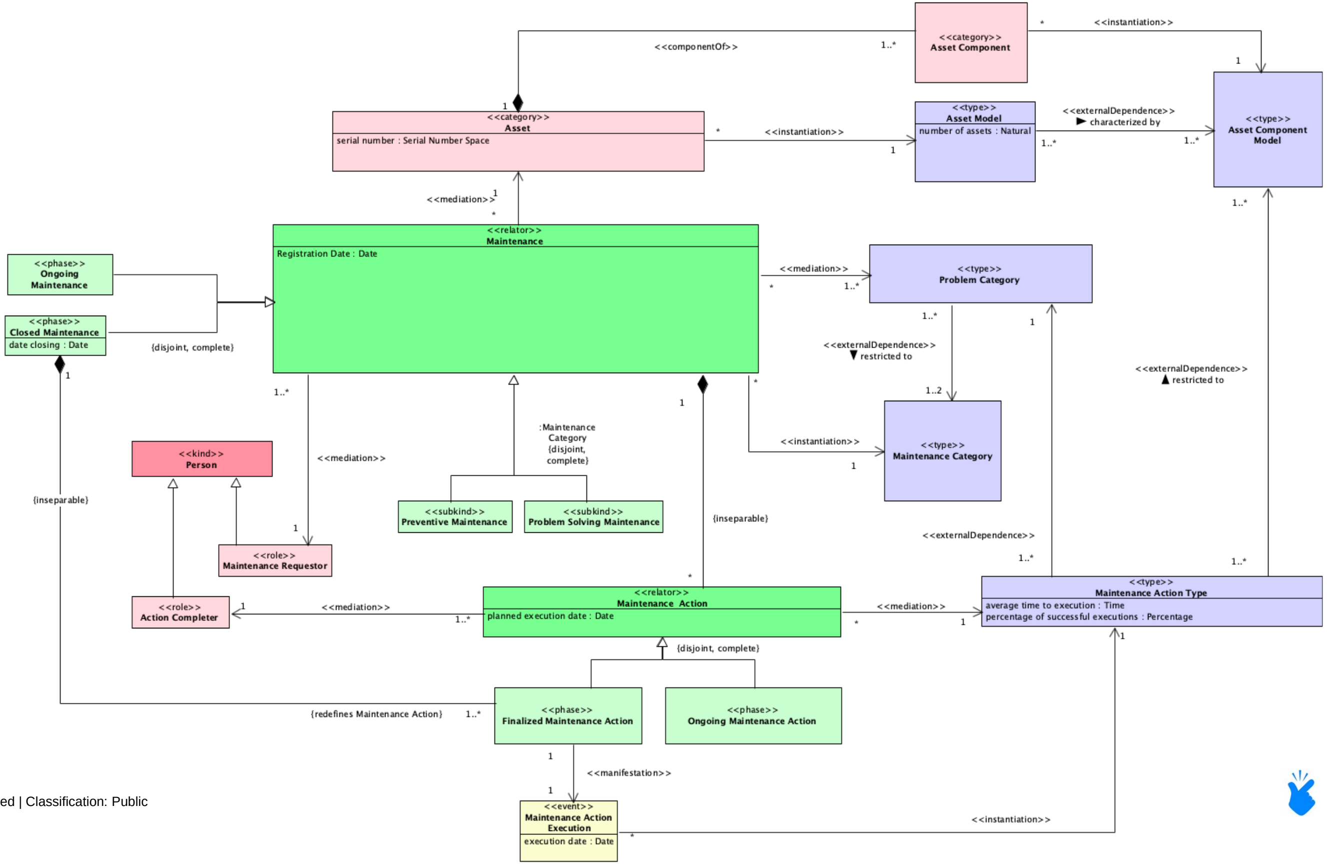
In the "old world" (music broadcast via terrestrial transmitters), the rights society would have to use market research to estimate the number of tracks and listeners. By contrast, the "new world" (music broadcast via

# Case study: 3D printer maintenance



| Transaction kind | Product kind | Executor role |
|---|---|---|
| Maintenance completing<br>Action completing | [Maintenance] is completed<br>[Action] is completed | Maintenance completer<br>Action completer |

# Case study: 3D printer maintenance



<<category>>
Asset Component

<<componentOf>>
1..*
1

*
<<instantiation>>
1

<<category>>
Asset
serial number : Serial Number Space

*
<<instantiation>>
1

<<type>>
Asset Model
number of assets : Natural

<<externalDependence>>
▶ characterized by
1..*
1..*

<<type>>
Asset Component
Model

1..*

<<mediation>>
1
*

<<phase>>
Ongoing
Maintenance

<<phase>>
Closed Maintenance
date closing : Date

{disjoint, complete}

<<relator>>
Maintenance
Registration Date : Date

<<mediation>>
*
1..*

<<type>>
Problem Category

1..*
1

<<externalDependence>>
▼ restricted to
1..2

<<externalDependence>>
▲ restricted to

1
1..*
:Maintenance
Category
{disjoint,
complete}
1
*

<<instantiation>>
1

<<type>>
Maintenance Category

<<kind>>
Person

<<mediation>>
1

{inseparable}

<<subkind>>
Preventive Maintenance

<<subkind>>
Problem Solving Maintenance

{inseparable}

<<externalDependence>>
1..*
1..*

<<role>>
Maintenance Requestor

*

<<role>>
Action Completer

1
<<mediation>>
1..*

<<relator>>
Maintenance Action
planned execution date : Date

<<mediation>>
*
1

<<type>>
Maintenance Action Type
average time to execution : Time
percentage of successful executions : Percentage

1

{disjoint, complete}

{redefines Maintenance Action}
1..*

<<phase>>
Finalized Maintenance Action

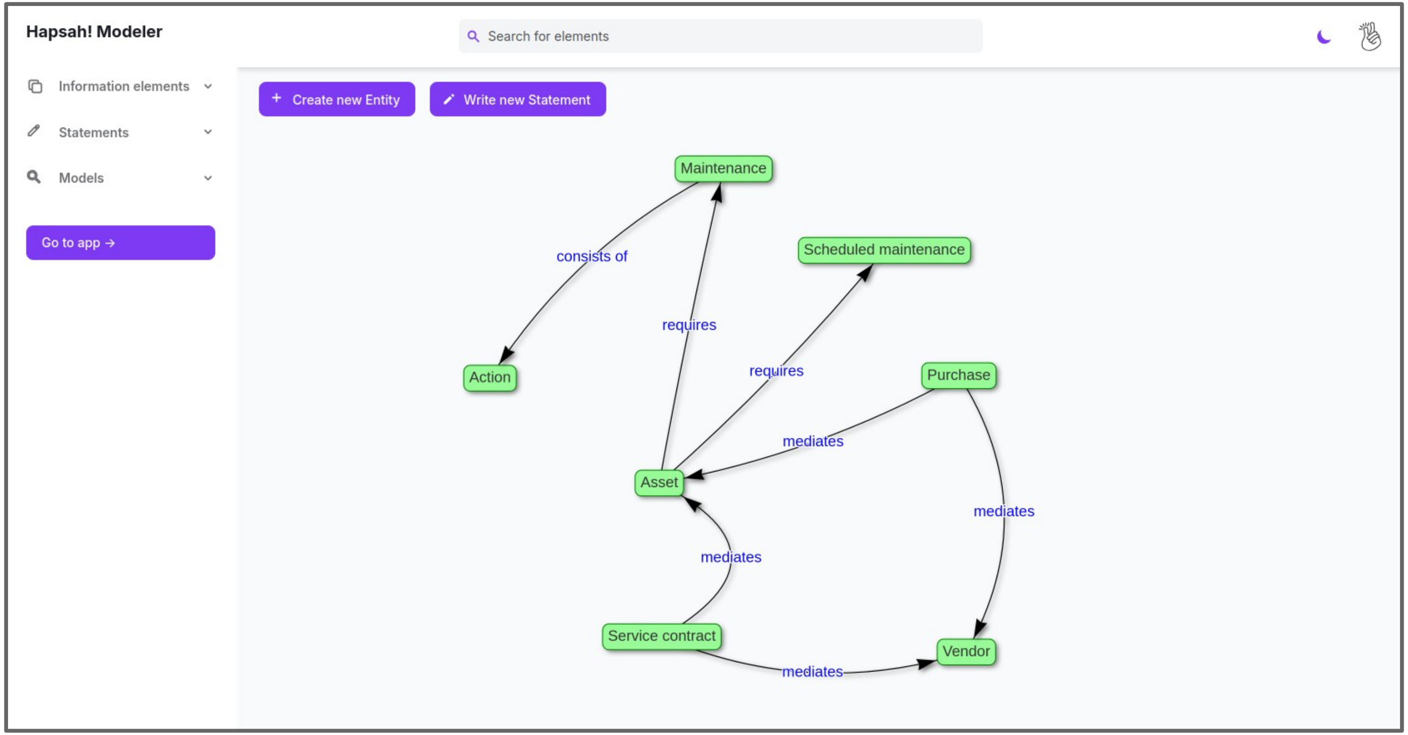
<<phase>>
Ongoing Maintenance Action

1
<<manifestation>>
1

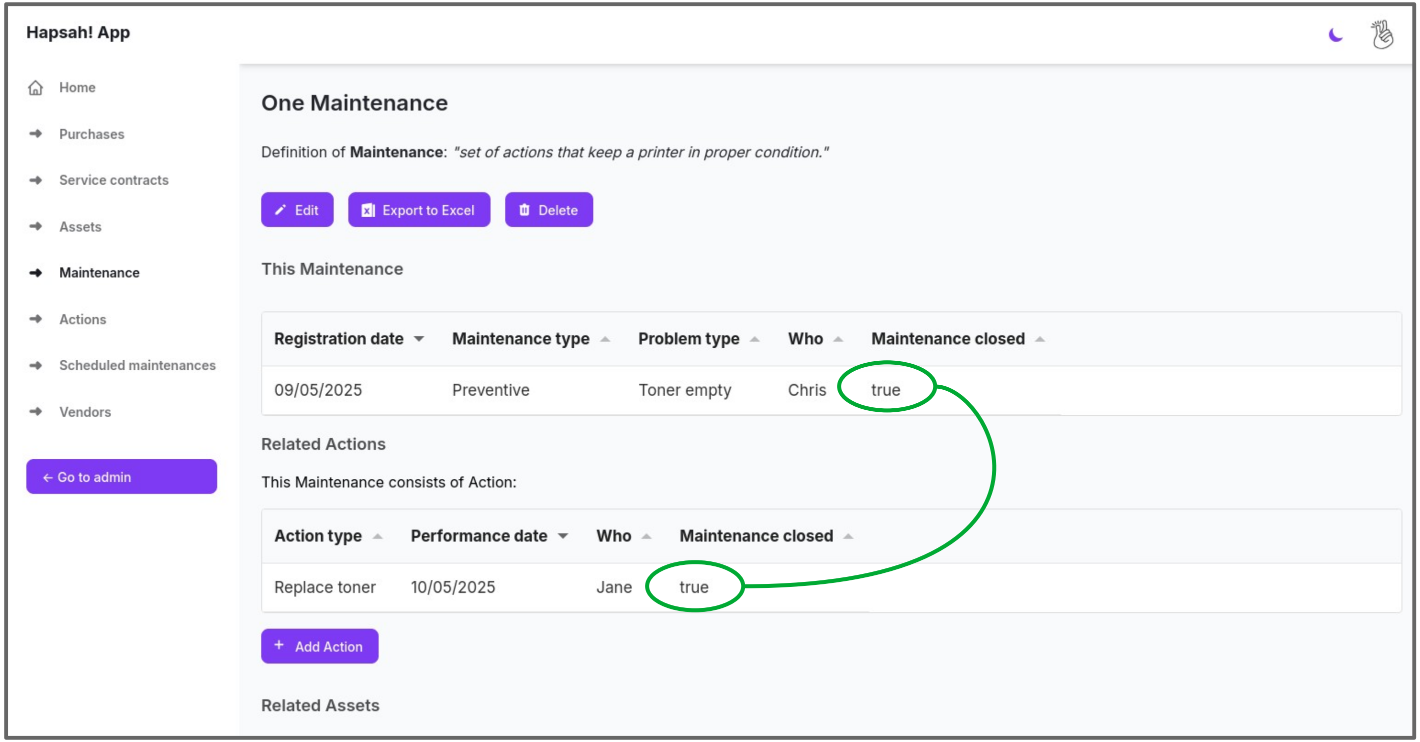
<<event>>
Maintenance Action
Execution
execution date : Date

*
<<instantiation>>

# Case study: 3D printer maintenance



# Case study: 3D printer maintenance



Hapsah! App

- Home
- Purchases
- Service contracts
- Assets
- Maintenance
- Actions
- Scheduled maintenances
- Vendors

← Go to admin

## One Maintenance

Definition of **Maintenance**: *"set of actions that keep a printer in proper condition."*

Edit | Export to Excel | Delete

### This Maintenance

| Registration date | Maintenance type | Problem type | Who | Maintenance closed |
|---|---|---|---|---|
| 09/05/2025 | Preventive | Toner empty | Chris | true |

### Related Actions

This Maintenance consists of Action:

| Action type | Performance date | Who | Maintenance closed |
|---|---|---|---|
| Replace toner | 10/05/2025 | Jane | true |

+ Add Action

### Related Assets

# Case study: 3D printer maintenance

When

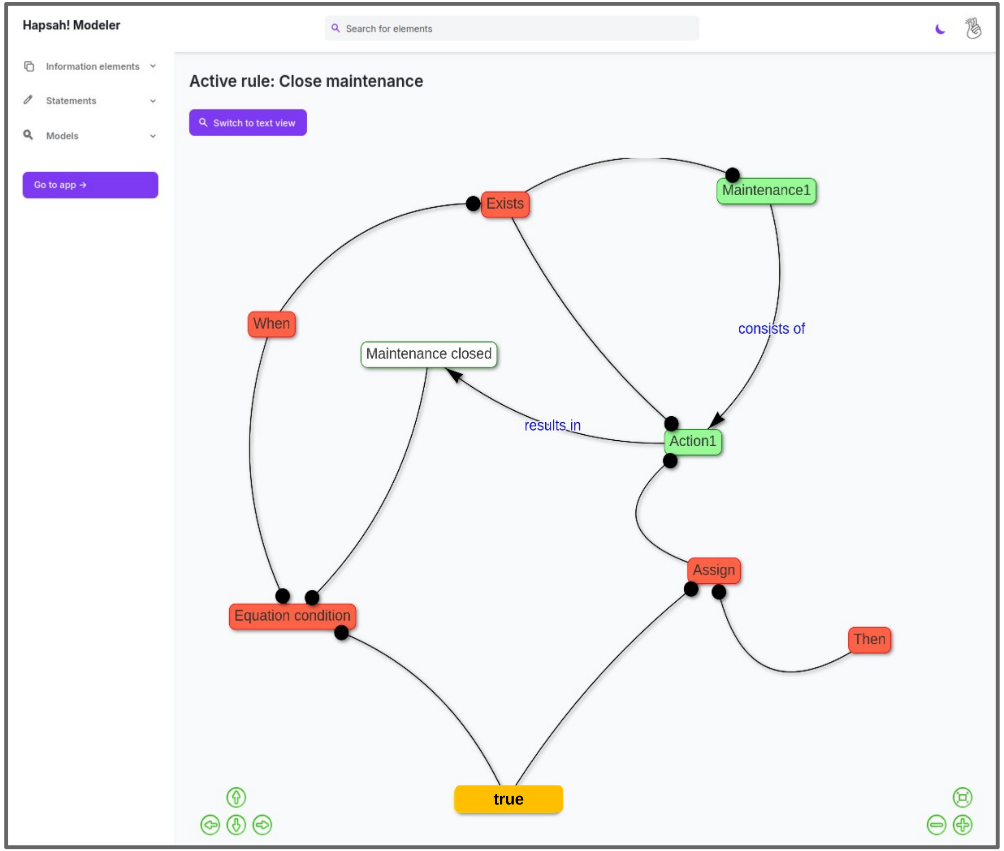
- Maintenance1 *consists of* Action1

and

- Action1 *results in* Maintenance closed
  is equal to **true**

Then

- assign **true** to Maintenance1
  as the Maintenance closed *state it is in*



Hapsah! Modeler

Active rule: Close maintenance

# What did we cover in part 2: “Taking the next step”

- Ontology goes beyond just information

- Different ontology languages:
  - DEMO
  - OntoUML
  - i*
  - e3-value

- How to combine those languages into one modeling approach:
  - Case study: 3D printer maintenance

# Questions?