



SEMANTIC
TREEHOUSE

Shaping application profiles with STH

IDSA Tech Talk | 27 June 2024



Contact



Michiel Stornebrink

 [/michielstornebrink](#)

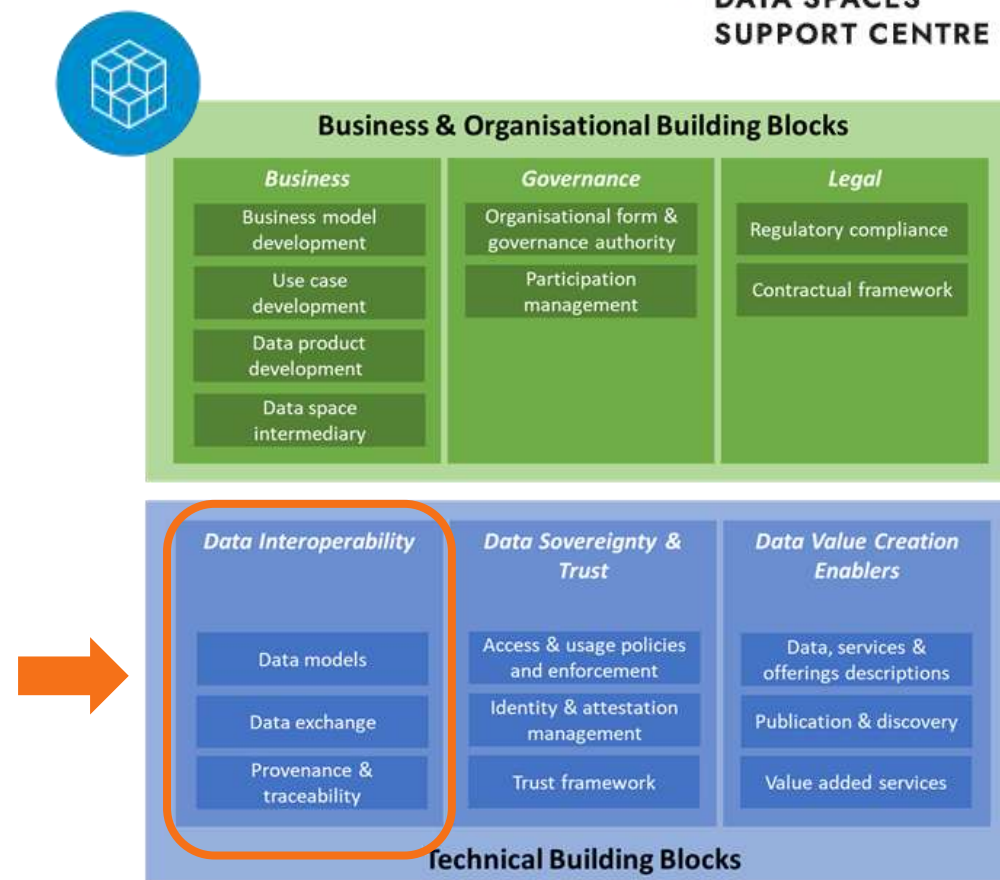
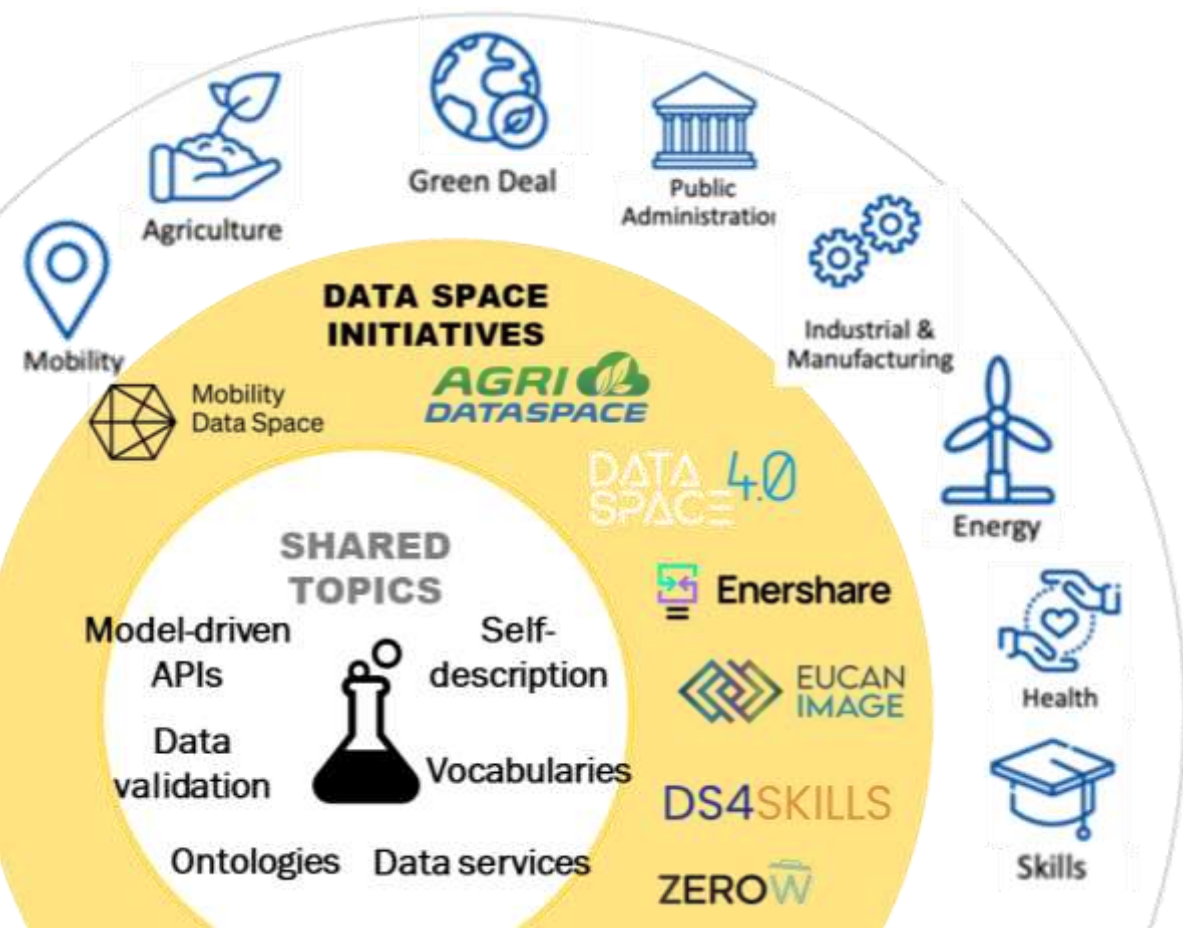
 michiel.stornebrink@tno.nl



Relevant experience on Semantic Interoperability for data spaces:

- DSSC Expert on Data interoperability
- DSSC Task lead to develop the toolbox (catalogue of tools)
- Active in IDSA community
- WP lead in ENSHARE project; shaping a common European Energy Data space
- Steering group member of the PLDN community (network of linked data experts in NL)
- Product owner of Semantic Treehouse at TNO

Data spaces require solutions for data interoperability



A vocabulary hub to support data spaces with data model use and -management



Semantic
Treehouse

Online community platform
for **business & IT**
to work collaboratively
on **data models**

GitHub



Online
community platform
for **developers**
to collaboratively work
on **software**

SwaggerHub



Online
platform
for **developers**
to collaboratively on
APIs

Communities using Semantic Treehouse

Dutch sectoral initiatives and standard development organizations (SDOs)

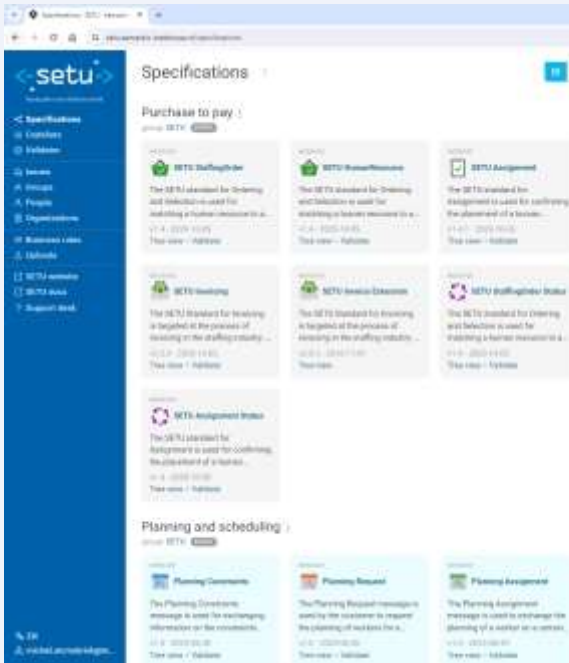


European data space / data sharing initiatives

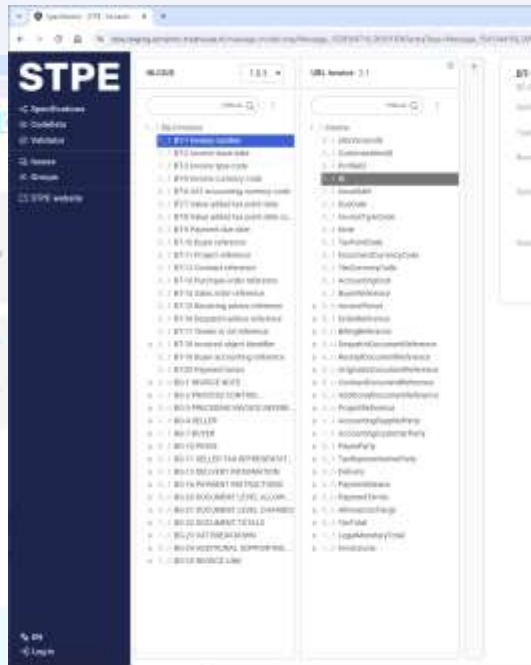


With STH we support SDOs with their data model management processes

<https://setu.semantic-treehouse.nl>



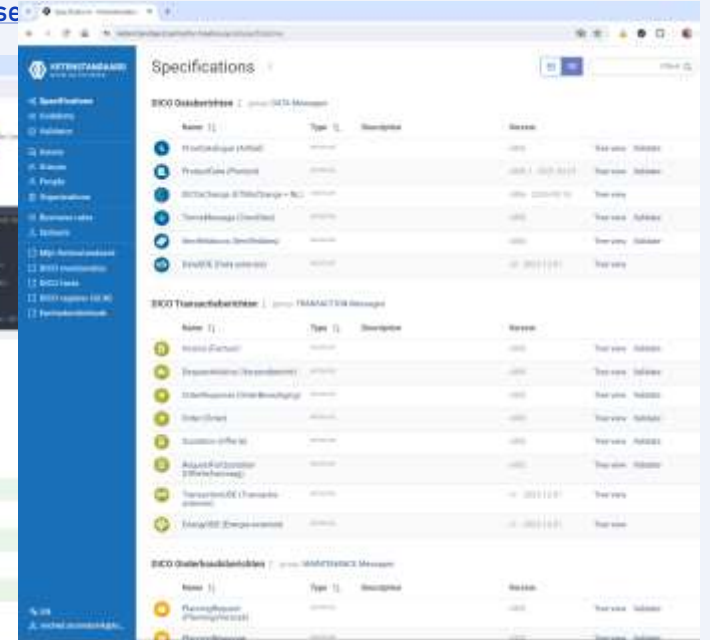
<https://stpe.semantic-treehouse.nl>



<https://smartconnected.semantic-treehouse.nl>



<https://ketenstandaard.semantic-treehouse.nl>

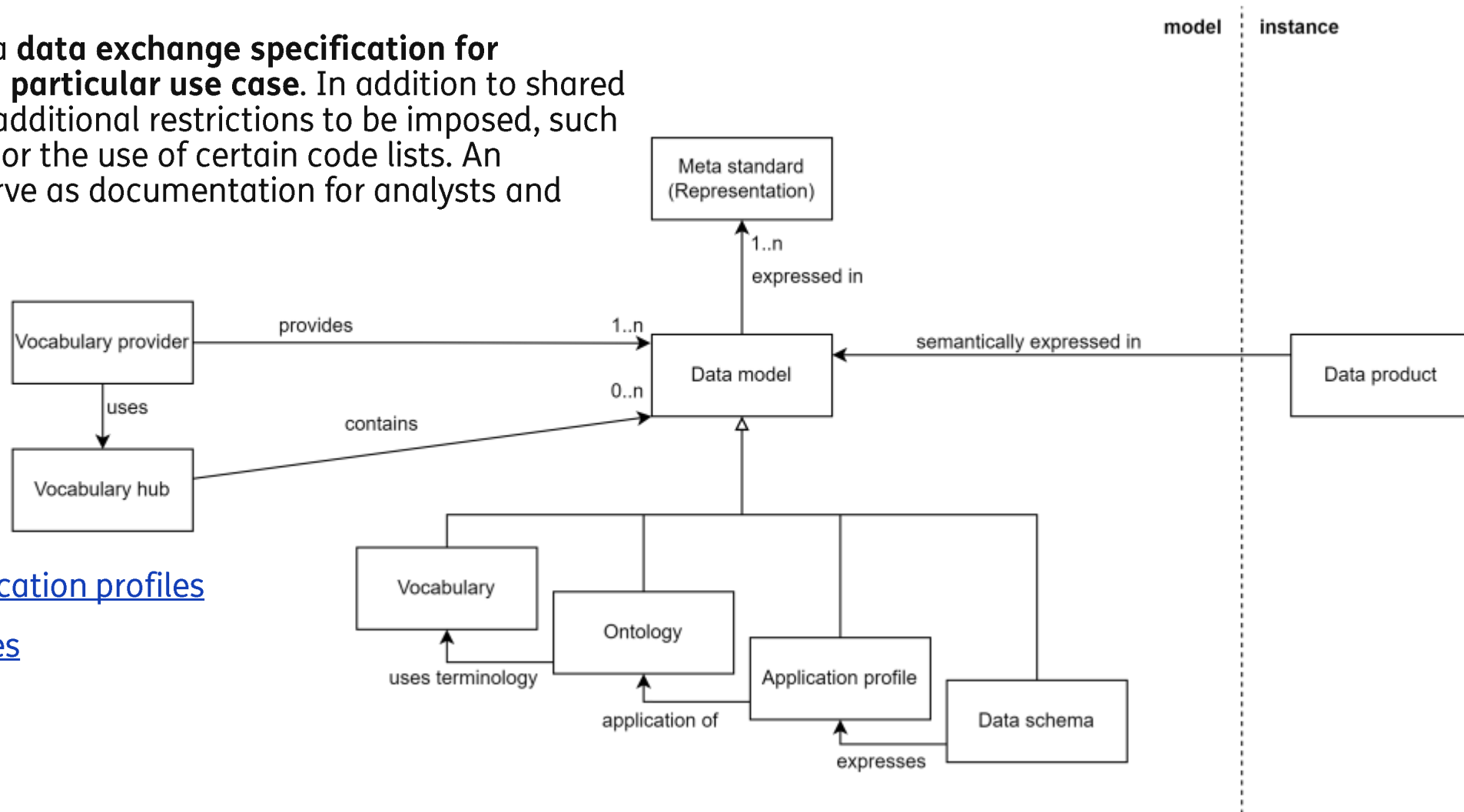


Application profiles

“An application profile is a **data exchange specification for applications that fulfill a particular use case**. In addition to shared semantics, it also allows additional restrictions to be imposed, such as recording cardinalities or the use of certain code lists. An application profile can serve as documentation for analysts and developers.”¹

Examples:

- [DCAT-AP](#)
- [EN16931 CIUS](#)
- [Data Vlaanderen application profiles](#)
- [Open Trip Model profiles](#)
- Many more...



1. Source: data.vlaanderen.be

2. Source figure: [DSSC Blueprint v1.0 – Data models building block](#)

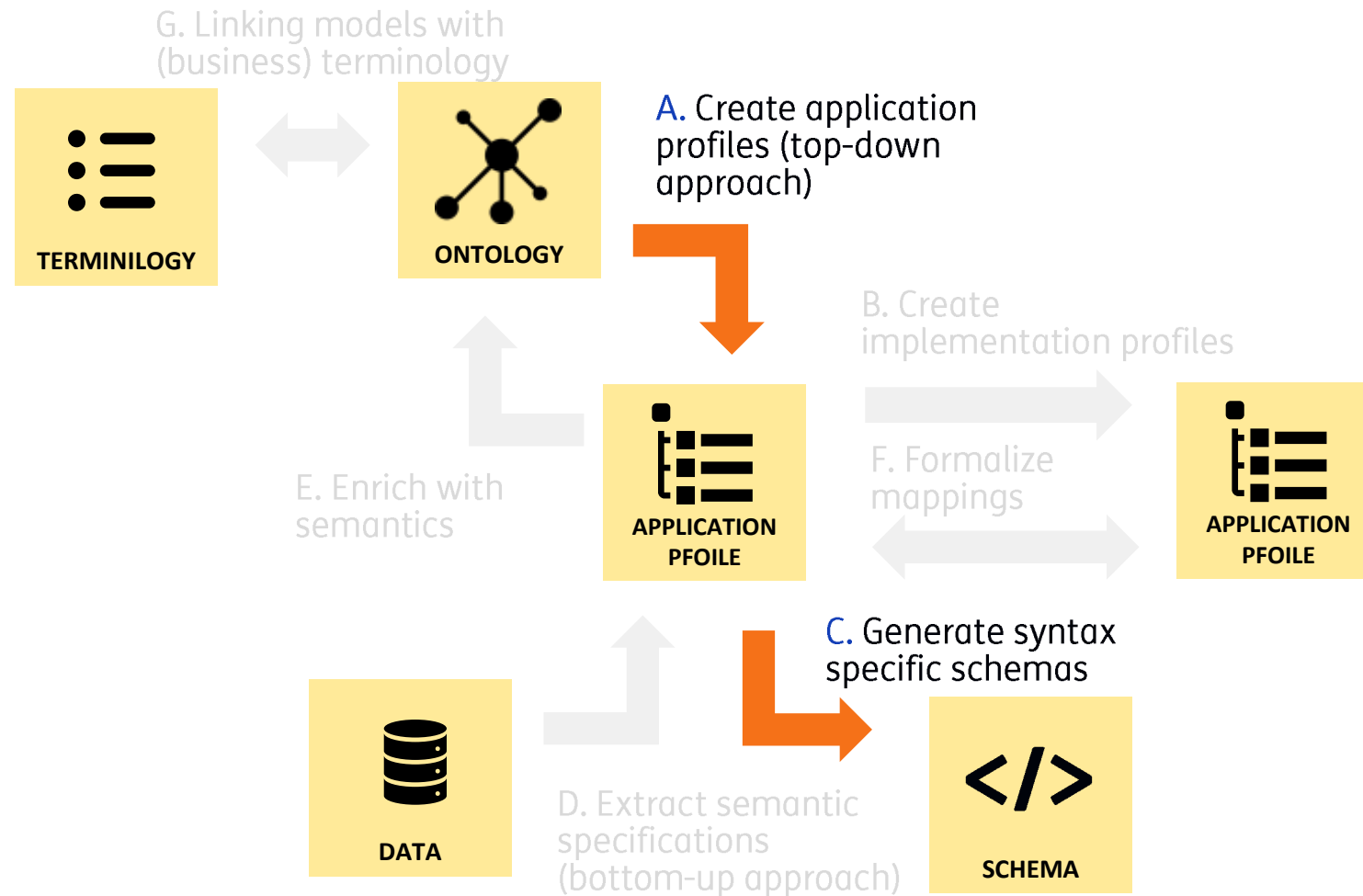
STH wizard to create, publish and maintain application profiles

We designed a wizard to:

- Lower barriers of using existing semantic models / standards
- Shape application profiles based on these semantic models
- Generate machine readable specifications according to open standards that can be used for software development (JSON schema, XSD, SHACL)

Learn more:

- Documentation about the [STH wizard](#)
- [Support to use JSON schemas](#) as input (the ontology/conceptual model)



Use case 1: OTM Transport order profile

1. Reuse Open Trip Model
2. Create OTM profile for Transport order. Information requirement + constraints are articulated by group of stakeholders
3. Visualize as tree structure for human understanding
4. Export in machine readable format to adopt profile in tool chain. Format of choice is JSON schema
5. Configure validator for developers to check compliance

Live environment:
<https://sutc.semantic-treehouse.nl> (You need access rights of *maintainer*)

The screenshot shows the SUTC Semantic Treehouse interface. The left sidebar contains navigation options: Specifications, Codelists, Validator, Issues, Group, People, Organizations, Accounts, Business rules, Message mappings, Uploads, and SUTC website. The main content area is titled 'Specifications' and shows the 'Open Trip Model' (group: OTM kernteam) and 'OTM profiles' (group: OTM kernteam). The 'Open Trip Model' card is labeled 'EXTERNAL' and 'Open Trip Model', with a description: 'OpenTripModel is a simple, free, lightweight and easy-to-use data model, used to exchange real-tim...'. The 'OTM profiles' section shows several cards, including 'OTM Profile - Transport Order' (MESSAGE, v1, Tree view), 'TMS - FMS profile' (MESSAGE, v0.1, Tree view), 'Test2' (MESSAGE, v0.1, Tree view), 'OTM profile - Actual Roadworks' (MESSAGE, v0.1, Tree view), and 'Test' (MESSAGE, v0.1, Tree view). An orange arrow labeled 'Input' points to the 'Open Trip Model' card, and another orange arrow labeled 'Result' points to the 'OTM Profile - Transport Order' card.

Creating an application profile – 3 step wizard

The image displays a three-step wizard for creating an application profile in the SUTC tool. The interface is divided into three main panels, connected by orange arrows indicating the flow from left to right.

Step 1: Create message model

- Specification:** Specification name: "OTM profile - Transport Order", Project: "Open Trip Model".
- Version:** Model version: "0.1", Status: "WIP".
- Message:** Message name: "TransportOrder", Message namespace URI: (empty).
- Message basis:** JSON schema, Ontology, Sample data.
- Import JSON schemas:** JSON Schemas: "OTM schema...", Based on class (full URI): (empty).
- Buttons:** "Create message model", "Read help".

Step 2: Edit message model version

- Tree View:** OTM Profile - Transport Order (1) > 1..1 transportOrder > 1..n consignments > 1..1 associationType > 1..1 entity > 1..n goods.
- Edit element panel:** Element: "entity", Value: (empty), Label: "entity", Element name: "entity", Namespace: "test", Definition: (empty), Min multiplicity: "1", Max multiplicity: "1", Ref element to: "Add Message", Sub elements: (empty).
- Buttons:** "Add sub element", "Add all", "Add next level descendants".

Step 3: Export

- Export options:** XML, JSON, RDF, OAS.
- Schema format:** JSON, YAML.
- JSON Schema:**

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "OTM Profile - Transport Order version 1",
  "description": "Generated by Semantic Treehouse on 2024-12-12",
  "required": [
    "consignments",
    "id"
  ],
  "additionalProperties": false,
  "properties": {
    "consignments": {
      "type": "array",
      "items": {
        "type": "object",
        "required": [
          "associationType",
          "entity"
        ],
        "properties": {
          "associationType": {
            "type": "string",
            "enum": [
              "inline"
            ]
          },
          "entity": {
            "type": "object",
            "required": [
              "goods"
            ],
            "properties": {
              "goods": {
                "type": "array",
                "items": {
                  "type": "object",
                  "required": [
                    "associationType",
                    "entity"
                  ],
                  "properties": {
                    "associationType": {
                      "type": "string",
                      "enum": [
                        "inline"
                      ]
                    },
                    "entity": {
                      "type": "object",
                      "required": [
                        "type"
                      ],
                      "properties": {
                        "type": {
                          "type": "string",
                          "enum": [
                            "Items"
                          ]
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```
- Example:**

```
{
  "consignments": [
    {
      "associationType": "inline",
      "entity": {
        "goods": [
          {
            "associationType": "inline",
            "entity": {
              "type": "Items"
            }
          }
        ]
      }
    }
  ]
}
```
- RML:**

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

()
rml:logicalSource [
  rml:source "http://www.example.com/root" ;
```

Use case 2: combining different semantic models from different SDOs

- Specify event message type for Estimated Time of Arrival (ETA)
- Combining generic wrapper and domain specific payloads
 - Generic: **FEDeRATED** event
 - Domain specific: **DCSA** payload
- Explicitly model the payload for these event types
- Output schema's for different syntaxes (including XML, JSON and RDF)
- Schema can be used in API specs to **configure data planes** (Dataspace Protocol)

4. ETA 0.1 ▾

CTRL+K 🔍 ⋮

1...1 ETA Event

- 1...1 UUID
- 1...1 Event Type
- 1...1 Has Time Classification
- ▾ 1...1 Container
 - 1...1 Container Number
- 1...1 Location
- ▾ 1...1 Vessel
 - 0...1 Vessel Name
- 1...1 Timestamp
- ▾ 0...1 Transport Journey
 - 1...1 Datetime
 - 1...1 Event ID
 - 1...1 TimelineStatus
 - 1...1 Transport event kind
 - 1...1 At kind of facility
 - 0...1 Per transport mode
- 0...1 Equipment Journey
- 0...1 Shipment Journey
- ⊕ 0...n Involves Business Identifier
- 0...n Involves Actor...
- 0...n Involves Transport Means...
- ⊕ Add all

FEDeRATED

DCSA



#	Event	Event Message			
		Generic BDI Wrapper		Domain Specific Payload Data	
		Format	Data	Format	Data
1	Pallet is loaded into container into truck A	BDI	Event ID, Event Type: Loading Event Object : Pallet, Pallet ID Object : Container, Container ID Location : Warehouse A, Transport Means: Truck Time	GS1	EPCIS event Product data GTIN
2	Estimated time of arrival of truck A at port A	BDI	Event ID, Event Type: ETA Object : Container, Container ID Location : Port A, Transport Means: Truck Time	OTM	OTM event Truck + Trailer ID Vehicle Data, Cargo Data Route Data, Shipment Data
3	Truck has arrived at port A, gate in	BDI	Event ID, Event Type: Gate in Object : Container, Container ID Location : Port A, Transport Means: Truck Time	DCSA	DCSA event Transport Journey Equipment Journey Shipment Journey
4	Estimated time of arrival of vessel at port B	BDI	Event ID, Event Type: ETA Object : Container, Container ID Location : Port B, Transport Means: Vessel Time	DCSA	DCSA event Transport Journey, Itinerary Equipment Journey Shipment Journey
5	Estimated discharge time of container	BDI	Event ID, Event Type: ETA Object : Container, Container ID Location : Port B Time	DCSA	DCSA event Transport Journey Equipment Journey Shipment Journey
6	Customs cleared, ready for pickup	BDI	Event ID, Event Type: Customs Cleared Object : Container, Container ID Location : Port B Time	DCSA	DCSA event Transport Journey, Customs Data Equipment Journey Shipment Journey
7	Gate out, actual time of departure of truck B	BDI	Event ID, Event Type: Gate Out Object : Container, Container ID Location : Port B, Transport Means: Truck Time	DCSA	DCSA event Transport Journey Equipment Journey Shipment Journey
8	Estimated time of arrival of truck B at warehouse B	BDI	Event ID, Event Type: ETA Object : Container, Container ID Location : Warehouse B, Transport Means: Truck Time	OTM	OTM event Truck + Trailer ID Vehicle Data, Cargo Data Route Data, Shipment Data
9	Pallet unloaded from container at warehouse B	BDI	Event ID, Event Type: Unloading event Object : Pallet, Pallet ID Location : Warehouse B Time	GS1	EPCIS event Product Data GTIN

Supplier
Warehouse A

1

Transporter A

2

Port A

3

Shipper

4

Port B

5

Transporter B

6

Supplier

7

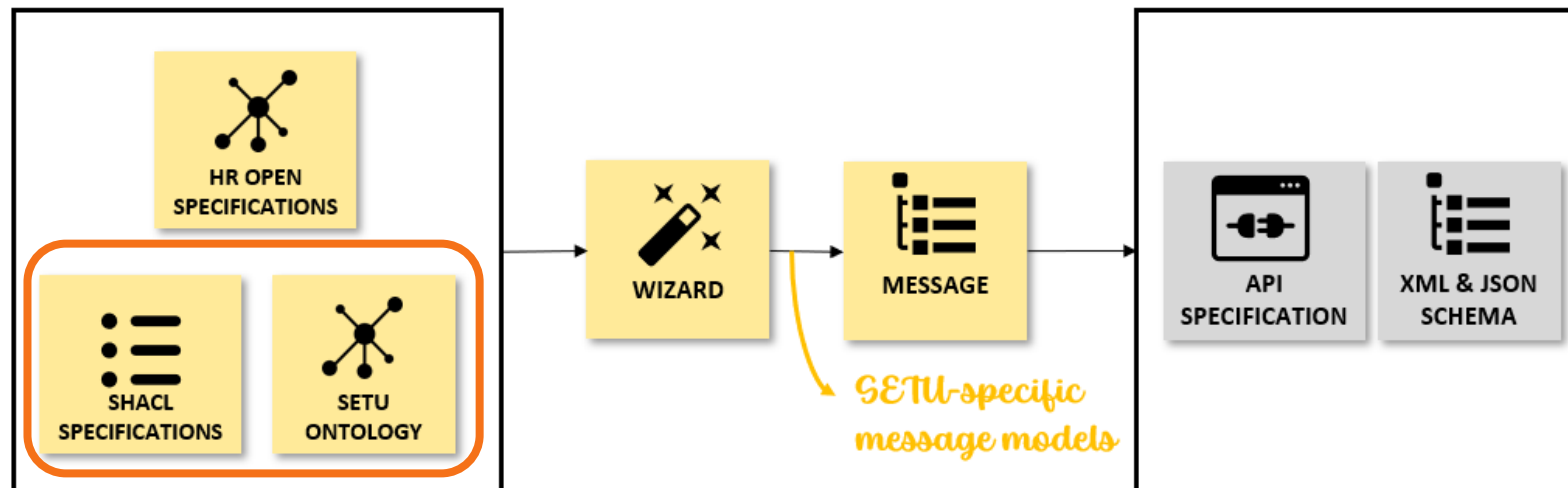
Use case 3: extending existing models with use case specific constraints

- Context: flexible staffing industry; human resource planning on a weekly basis.
- Required application profiles:
 - Planning constraints model
 - Planning request model
 - Planning assignment model
- Extending HR Open specifications + adding use case specific constraints.

The screenshot shows the setu application interface. On the left is a blue sidebar menu with the following items: Specifications, Codelists, Validator, Issues, Groups, People, Organizations, Accounts, Business rules, and Uploads. The main content area displays a 'Planning Request' element (version 1.0) with a search bar (CTRL+K) and a tree view of its structure:

- 1...1 PlanningRequest
 - 1...1 document id
 - 0...1 action code
 - 0...1 supplier
 - 0...2 id
 - 0...1 legal id
 - 0...1 tax id
 - 1...1 id ref...
 - 0...1 legal id ref...
 - 0...1 tax id ref...
 - 0...1 communication ref...
 - 0...1 person contacts ref...
 - 0...1 name...
 - 0...1 code...
 - ⊕ Add all
 - ⊕ Add next level descendants
 - 1...1 customer
 - 1...1 position profile
 - 0...n request comments
 - 0...n requested periodic planning line
 - 0...n requested single planning line

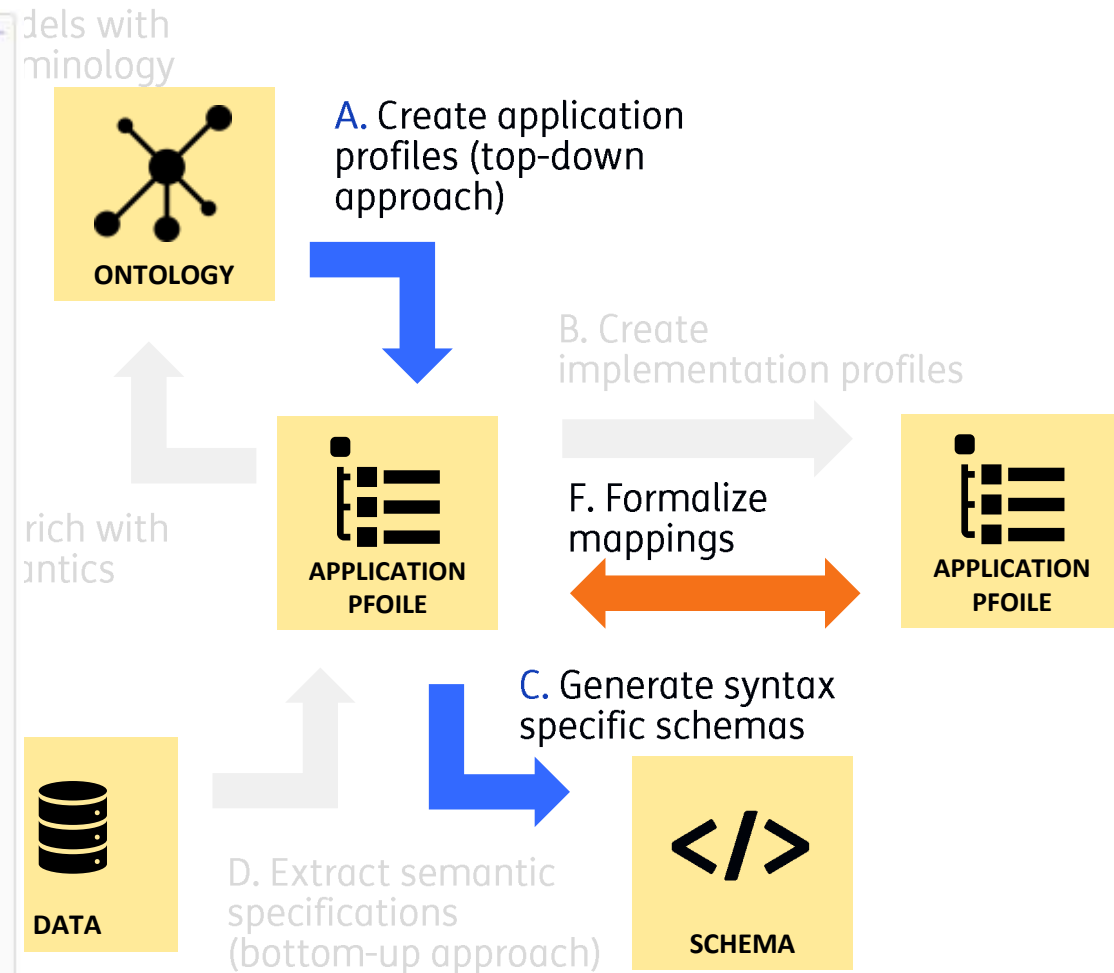
On the right, the 'Edit element' panel shows the 'supplier' element selected. It displays the label 'supplier', the element name 'supplier', the namespace 'https://ontology.se', and the definition 'The supplier rece'. It also includes input fields for 'Min multiplicity' (0) and 'Max multiplicity' (1), and a 'Ref element to' field with the value 'Add Message'.



One more thing: mapping to existing models

The screenshot displays the STPE interface with three panels:

- NLCIUS 1.0.3**: A list of invoice elements including BT-1 Invoice number, BT-2 Invoice issue date, BT-3 Invoice type code, BT-5 Invoice currency code, BT-6 VAT accounting currency code, BT-7 Value added tax point date, BT-8 Value added tax point date co., BT-9 Payment due date, BT-10 Buyer reference, BT-11 Project reference, BT-12 Contract reference, BT-13 Purchase order reference, BT-14 Sales order reference, BT-15 Receiving advice reference, BT-16 Despatch advice reference, BT-17 Tender or lot reference, BT-18 Invoiced object identifier, BT-19 Buyer accounting reference, BT-20 Payment terms, BG-1 INVOICE NOTE, BG-2 PROCESS CONTROL, BG-3 PRECEDING INVOICE REFERE..., BG-4 SELLER, BG-7 BUYER, BG-10 PAYEE, BG-11 SELLER TAX REPRESENTAT..., BG-13 DELIVERY INFORMATION, BG-16 PAYMENT INSTRUCTIONS, BG-20 DOCUMENT LEVEL ALLOW..., BG-21 DOCUMENT LEVEL CHARGES, BG-22 DOCUMENT TOTALS, BG-23 VAT BREAKDOWN, BG-24 ADDITIONAL SUPPORTING ..., and BG-25 INVOICE LINE.
- UBL Invoice 2.1**: A tree structure of invoice elements including UBLVersionID, CustomizationID, ProfileID, ID, IssueDate, DueDate, InvoiceTypeCode, Note, TaxPointDate, DocumentCurrencyCode, TaxCurrencyCode, AccountingCost, BuyerReference, InvoicePeriod, OrderReference, BillingReference, DespatchDocumentReference, ReceiptDocumentReference, OriginatorDocumentReference, ContractDocumentReference, AdditionalDocumentReference, ProjectReference, AccountingSupplierParty, AccountingCustomerParty, PayeeParty, TaxRepresentativeParty, Delivery, PaymentMeans, PaymentTerms, AllowanceCharge, TaxTotal, LegalMonetaryTotal, and InvoiceLine.
- UN/CEFACT Cross Industry Invoice 16.1**: A tree structure of invoice elements including CrossIndustryInvoice, ExchangedDocumentContext, ExchangedDocument, ID, TypeCode, IssueDateTime, IncludedNote, SupplyChainTradeTransaction, IncludedSupplyChainTradeLine..., ApplicableHeaderTradeAgree..., ApplicableHeaderTradeDelivery..., and ApplicableHeaderTradeSettle...



Want to know more about what Semantic Treehouse can bring for you?

- Documentation page: <https://www.semantic-treehouse.nl/>
- Testimonials by [communities using STH](#)
- Position paper + ENDORSE-conf recording on [Vocabulary hub for data spaces](#)
- Position paper on [Sharing Vocabularies in Federated Data Spaces](#) using DCAT
- Get in contact with the team via Discord → <https://discord.gg/kdrbm9RUu8>
- Last years [IDSA Tech Talk on Semantic Interoperability](#)



SEMANTIC
TREEHOUSE

Thank you

