Defining the COVESA data modeling strategy and its associated artifacts

- Current setup (summary)
- Current setup (description)
- Proposal overview (summary)
- Proposal overview (description)
  - [1] - Generalise the current data modeling approach used in VSS to make it re usable when needed
    - [1a] - Generic YAML-based tree modeling approach (one hierarchy)
    - [1b] - Rule-set for a new tree model
  - [2] - Extend the tools to use standard data exchange format and to allow the construction of custom schemas
    - [2a] - A standard data exchange format
    - [2b] - Allow the construction of custom schemas
  - [3] - Publish the COVESA tree(s) with standards and future-proof name spaces
  - [4] - Use a COVESA tree to automatically feed one hierarchy of a domain ontology
    - [4a] - Domain ontology
    - [4b] - COVESA core ontology
- Implications (summary)
- How to move forward?
  - Use a methodology suitable to design a solution

Disclaimer

This proposal is just that, a proposal. So, contributors are very welcome. If any ideas or assertions presented in this draft are incorrect, do not hesitate to share your own view by commenting on the page or contacting the author (Daniel Alvarez). The principal intention of sharing these ideas is to find synergies and joint agreements on the best possible next steps for the data modeling group. In this sense, I tried to point out aspects where I believe there is room for improvement with as much constructive criticism as possible.

This page describes the proposal to restructure the data modeling activities within COVESA. First, the current state is presented to provide the necessary context. Then, the proposal is introduced.

Current setup (summary)

- As of today (05.2023), COVESA's data modelling activities focus mainly on VSS.
- VSS is the conceptual model for the description of vehicle properties in a high-level of

Current setup (description)

So far, the data modeling activities in COVESA have been primarily centered on the continuous development and maintenance of the Vehicle Signal Specification (VSS) and the tools that parse VSS into different formats. In the current setup, there is no clear description of what requirements (i.e., functional and non-functional) are driving the design of the data model. It seems that the primary purpose of VSS is to serve as a naming convention for the properties of the vehicle. Nevertheless, there is little attention given to the separation of concerns:

abstraction (i.e., friendly for non-experts).
- VSS-tools complement VSS by providing the scripts that parse the specification into a few standard formats.
- Pros
    - Easy to understand and contribute to.
    - Simple YAML files with custom constructs.
    - VSS model constantly maintained
    - Naming convention.
- Cons
    - The specification of the data model is not represented in a standard data exchange format (only after parsing it using the tools).
    - One tree covers only one hierarchy.
    - No clear definition of the scope. That is, where does VSS end and something else start?
    - No clear distinction between conceptual and application areas.
    - Most of the attention is given to the leaves of the tree, while no rule-set is in place to control how the information is classified in different branches.
    - A few misleading terms are used. For example:
        - Using "Signal" to refer to dynamic data properties of the vehicle (i.e., a signal is the information carrier that can have one or multiple properties, and it occurs at a lower layer of abstraction than what VSS defines).
        - Saying that VSS is a "Taxonomy" when the implicit relationship of the concepts in the hierarchy is

- On the one hand, there is the ***"conceptual area,"*** where the controlled vocabulary has to be described and adequately documented. Data models belong to this area because they define the entities of interest in a particular domain and the possible relationships between them.
- On the other hand, we have the ***"application area,"*** where a data model is used in specific implementations (e.g., databases, applications, etc.).

The figure above shows how VSS modeling belongs to the conceptual area. To use the specification described in VSS (i.e., a "vspec" file), one has to parse it into a specific format (e.g., JSON) by using the VSS tools. The tools are the mechanism that makes the VSS data model usable in the application area. From the practical point of view, the application area needs a specific schema that determines the structure in which the data is to be stored. In this context, we mean long-term storage (e.g., a database) or short-term storage (RAM and variables' allocation during application execution).

In the current setup, the whole data model is taken one-to-one and parsed as the schema for the application area. Then, it is up to the specific implementation to use custom mechanisms to ignore the overhead when only some concepts defined in the data model are required or used. Although this aspect has shown no significant limitation until now, it becomes relevant when multiple domains are involved. Therefore, with the increasing interest in adding other domains apart from vehicle-specific data, it is crucial to define a data modeling strategy that can scale beyond tree hierarchies and vehicle-specific data.

One may argue that, with the current setup, it is also possible to describe a customised shorter spec file or use functions (e.g., overlays) to define the specification that matches the needs of the application area. Although possible, that implies creating disparate data models. Ideally, the concepts should be modelled only once in the conceptual area to serve as a controlled vocabulary. Then, an arbitrary selection of the concepts and some modifications to the constraints (e.g., min, max, etc.) can deliver the schema needed in specific use-cases, without affecting the standard definition of the concepts. Hence, the following section propose a few specific tasks to improve the data modeling workflow.

not a sub classification .

Proposal overview (summary)

- Unique and standard definitions in the conceptual area, and arbitrary selection in the application area
- Suggested tasks to address:
  - (1) Generalize the VSS approach to model one tree-like hierarchy, so that another modelling group can reuse it when convenient (inside or outside COVESA).
    - Inside COVESA, the definition of a new tree only would take place when an specific rule set is fully satisfied.
  - (2) Keep using the YAML format for the further specification of the domain hierarchy, but extend the tools to first translate the YAML into the RDF standard data exchange format (i.e., publishing the specification in RDF + SKOS). Some advantages include:
    - Unique identification of resources
    - Interoperability
    - Machine readable format
    - RDF libraries available in multiple programming languages
    - RDF data can be queried with SPARQL
    - Facilitate the integration of multiple hierarchies because OWL ontologies are built on RDF data, so the tree hierarchy can

Proposal overview (description)

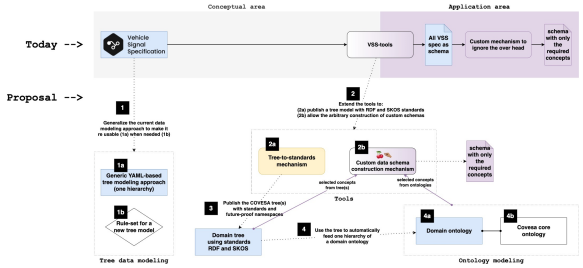
The idea is to define the data modelling workflow for COVESA in terms of the conceptual area (i.e., development and maintenance of the controlled vocabulary) and the application area (i.e., tools that construct a usable schema out of the data models for a particular use case).

- In the **conceptual area**, there should be a clear step-by-step guide on how to work with two different levels of expressiveness. This consideration is needed because a tree hierarchical model alone (what VSS has been so far) is not the best model type to handle some upcoming needs, such as data integration. Hence, the model might be selected depending on the user needs.
  - (less expressive) Tree hierarchical model, good for:
    - Information classification according to a given criteria (e.g., taxonomy, meronomy, custom tree.)
    - Naming convention following a dotted notation (i.e., concatenation of the branches)
  - (more expressive) Ontology, good for:
    - Data integration
    - Concepts re usability
    - Reasoning
    - Multiple hierarchies
    - Knowledge representation

> Regardless of the level of expressiveness selected by the end user, the data models developed and maintained by COVESA should use a common standard data exchange format. This standard representation of the data models will facilitate the transition from and inter operability between trees and ontologies.
>
> As reported by experienced data modellers [R1], the current preferred data exchange format for tree-like models is RDF and SKOS standards. However, the current VSS tree is published with a custom YAML `vspec` file that requires the tooling to parse it.
>
> Luckily, the RDF data model is also the foundation for the Web Ontology Language (OWL). Hence, there is a good opportunity here to harmonise the activities by using a common data model and vocabulary.

- In the **application area**, the tools have to provide a mechanism to "cherry pick" (i.e., to arbitrary select) concepts of interest from one or multiple domains, including the context and pointers to the uniquely identified definition of the concepts.

In other words, the proposal is not to enforce the use of ontologies. It is rather providing a common agreement on how and when to use what data model. The idea is better explained with the following figure, which is explained below:



[1] - Generalise the current data modeling approach used in VSS to make it re usable when needed

contribute to it.
    - Taxonomy editors and other tools can import RDF data with SKOS concepts in it directly.
- (3) Publish the COVESA tree(s) with standards and future-proof name spaces.
- (4) Use the COVESA tree(s) as a continuous feed for one hierarchy of a domain ontology.
    - Model in the COVESA ontology the concepts that are of common interest and that involve the integration of multiple domains:
        - What is an event? and what types of events there are, for example:
            - EV charging session
            - Car crash
            - Driver commutes from private car to public transportation using a parking spot at a park & ride site.
            - etc.

The simplicity of VSS has proven to be a successful approach for the continuous contribution of Subject Matter Experts (SMEs); just by modifying a text file, discussing the changes, and creating a pull request. The approach itself should be generalized to serve as a guideline to describe and maintain one hierarchy. The idea here is to abstract the modelling approach used in VSS and describe it with generic terms that might be re used with other domains. This implies doing some minor adjustments to geralize the YAML-based modeling approach, and also defining a rule set to support COVESA participants to identify the need of a new tree model.

[1a] - Generic YAML-based tree modeling approach (one hierarchy)

People often refer to a hierarchical tree data model as a taxonomy. However, this is not always the case. Depending of the meaning of the implicit relationship between branches of a tree, the hierarchy can be:

Tree hierarchy type	Implicit relationship	Example

Taxonomy	ChildBranch **-- Sub class of -->** ParentBranch	• Vehicle
		  ◦ Car
		  ▪ Private car
		  ▪ Company car
		  ◦ Plane
		  ▪ Commercial plane
		  ◦ Train

meronomy	ChildBranch **-- Part of -->** ParentBranch	• Vehicle
		  ◦ Wheel
		  ▪ Tire
		  ▪ Brake

Custom	ChildBranch **-- Custom -->** ParentBranch	VSS

To handle these semantic differences, the following tasks are proposed:

- Add a field at the beginning of the tree specification to explicitly state the tree type

```
# To include at the top of the spec YAML file (the root branch)
tree type as one of ['TAXONOMY','meronomy','CUSTOM']
```

- Extend the tools to interpret the tree type as follows

Tree type	Tools will consider...	If custom relationship is needed...
TAXONOMY	the "SubClassOf" as the default implicit relationship between branches	the user can define it within the branch definition.
meronomy	the "PartOf" as the default implicit relationship between branches	the user can define it within the branch definition.
CUSTOM	that no default implicit relationship exist.	in this case, it will be mandatory for all branches in the specification

- Add a field in the branch definition to explicitly state the implicit relationship to the parent branch

```
# Example of a branch that has a custom relationship
PowerTrain.Charging:
  type: branch
  description: Properties related to battery charging.
  relationToParentBranch: 'functionOfVehicleComponent'
```

[1b] - Rule-set for a new tree model

Having an stablished approach to model a tree hierarchy does not mean that COVESA should motivate the arbitrary creation of multiple trees. Data modeling is a continuous design task that requires several iterations and a huge maintenance effort. Therefore, it is essential to define a simple set of rules that are to be satisfied before starting a new data tree that is to be developed and maintained by COVESA. Such a rule set can include, for example:

- There must exist the proven need for at least 3 branches.
- Each branch must have at least 2 leaves.
- The implicit relationship between consecutive branches must be documented.
- There must be no cross references to other existing data models.
- Branch names must use this XYZ format
- There should be at least 2 people responsible for the organization and maintenance of the model
- etc.

[2] - Extend the tools to use standard data exchange format and to allow the construction of custom schemas

Tools should be extended to use standards for the specification itself, and also to be able to arbitrarily construct the desired schema.

[2a] - A standard data exchange format

Current VSS-tools allow parsing the specification into multiple formats. The specification itself is done in the custom YAML file (i.e., with the vspec format). This is not an standard data exchange format. It is a good practice to specify taxonomies (and similar hierarchies) using standard formats [R1]. Two important reasons for that are interoperability, and re use of existing tools for editing and visualising the data models.

Here, the most prominent standards that are preferred today for these taxonomy-like data models are the Resource Description Framework (RDF), and the Simple Knowledge Organization System (SKOS).

[2b] - Allow the construction of custom schemas

The idea here is that a data model can contain much more concepts that the application area needs. This is because the application area might consist of multiple use cases. In principle, the idea here is to keep to the number of data models to maintain by COVESA to the minimum. Hence, a mechanism to construct custom schemas will decouple the need for the modification of the model.

[3] - Publish the COVESA tree(s) with standards and future-proof name spaces

Right now, the specification has a practical use after applying the vss-tools. The main identifier of a concept of interest (i.e., a tree leaf) is either the path defined with a dotted notation, or the UUID introduced recently. None of them is fully solving the identification of the resource.

The paths require a custom script that interpret them in other to do the mapping, whereas the current UUID approach does not guarantee that the identifier will remain unchanged when the path changes (e. g., a branch name change will lead to a different UUID).

The idea here is to take advantage of the principles of the RDF, which is the identification of resource in the web. For that, one must define a so called name space that will act as the address where the model is stored (hopefully forever). For example:

```
@prefix vss: http://covesa.global/datamodels/vss# .
```

[4] - Use a COVESA tree to automatically feed one hierarchy of a domain ontology

This is the entry point for ontologies. The main limitation of a tree is that it can only handle one hierarchy. In the real world, however, the knowledge looks more like a semantic network than a simple tree.

[4a] - Domain ontology

We can take the advantage of a well-maintained tree to further enrich the vehicle data model. For example, a domain ontology can be useful for:

- Say something about the branches of the tree, and how they relate to other domains
  - Production information of a particular vehicle component
    - Material
    - Place
    - Manufacture data
- Connect vehicle data to other areas
  - Smart city
  - Traffic management
- Connect the VSS high-level view to the low-level data of the vehicle electronic system.
- etc.

[4b] - COVESA core ontology

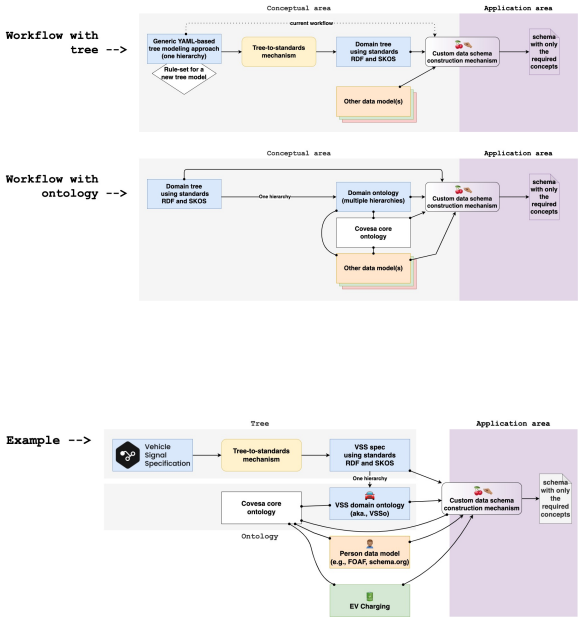
Once multiple models are involved, we must keep them inter operable. That means that we need to also have a controlled vocabulary to categorise new concept of interest into abstract and generic categories. For example

- A ChargingSession is an Event
  - Then the concept of an Event is modelled in the COVESA core ontology
- The same applies for other common concepts like time, colours, etc.

The systematic use of these controlled vocabularies will lead to an easy data integration.

Implications (summary)

- VSS approach of modeling things is maintained and reinforced with minor changes
- Tree models (the specification) are exposed with standards that allow machines to read it directly
- Trees and ontologies share the same foundation, RDF.
- COVESA has to decide the name space to store the data models forever.
- Use-case needs can be satisfied by cherry picking existing concepts from multiple domains.



How to move forward?

- Comprehensively describe the aspects that are driving the design of each of the artifacts that are (or will be) part of COVESA.

Use a methodology suitable to design a solution

Designing an artifact that solves a problem can lead to multiple valid solutions. Therefore, a simple and clear methodology to guarantee the validity of the proposed solution must contain at least these 4 elements (adapted from [R3]):

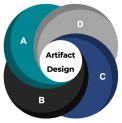
A well-defined scope leads to a valid and usable design.

A. **Problem context** → What is the current problem / challenge / limitation?

B. **Artifact** → What is the shape of the solution you want? (e.g., method, language notation, etc.)

C. **Requirements** → What is its function, and under which conditions it should work?

D. **Goal** → What are the desires of the stakeholders?



Answer → Follow the design cycle:

Design problem

Improve `<a problem context>`
by `<(re)designing an artifact>`
that satisfies `<some requirements>`
in order to `<help stakeholders achieve some goals>`

*Essential elements for designing an artifact, adapted from:
R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. doi: 10.1007/978-3-662-43839-8.

COVESA

Further references and recommended reads

- [R1] H. Hedden, The accidental taxonomist, Third edition. Medford, New Jersey: Information Today, Inc, 2022.
- [R2] P. Alexopoulos, Semantic modeling for data: avoiding pitfalls and breaking dilemmas. O' Reilly Media, Inc., 2020.
- [R3] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. doi: 10.1007/978-3-662-43839-8.