Common Vehicle Interface Initiative AMM Working Session

Version 1.2, October 2020

Is it time *to discuss the implication of having a common language to describe* ***data*** *and* ***function*** *interaction between all automotive technology companies* ***?***

Is it time *to commit to selected technologies, such as* ***open W3C protocols****, to build* <u>*interoperable*</u> *solutions for vehicle data and service invocation* ***?***

Is it <u>finally time</u> *to define the industry-wide* ***standard vehicle data model***

...and then do the same for **services / APIs ?**

After recognizing industry trends across many conferences, and after two OEM Roundtable discussions and W3C project activities,
the prevailing answer to these questions is "YES".

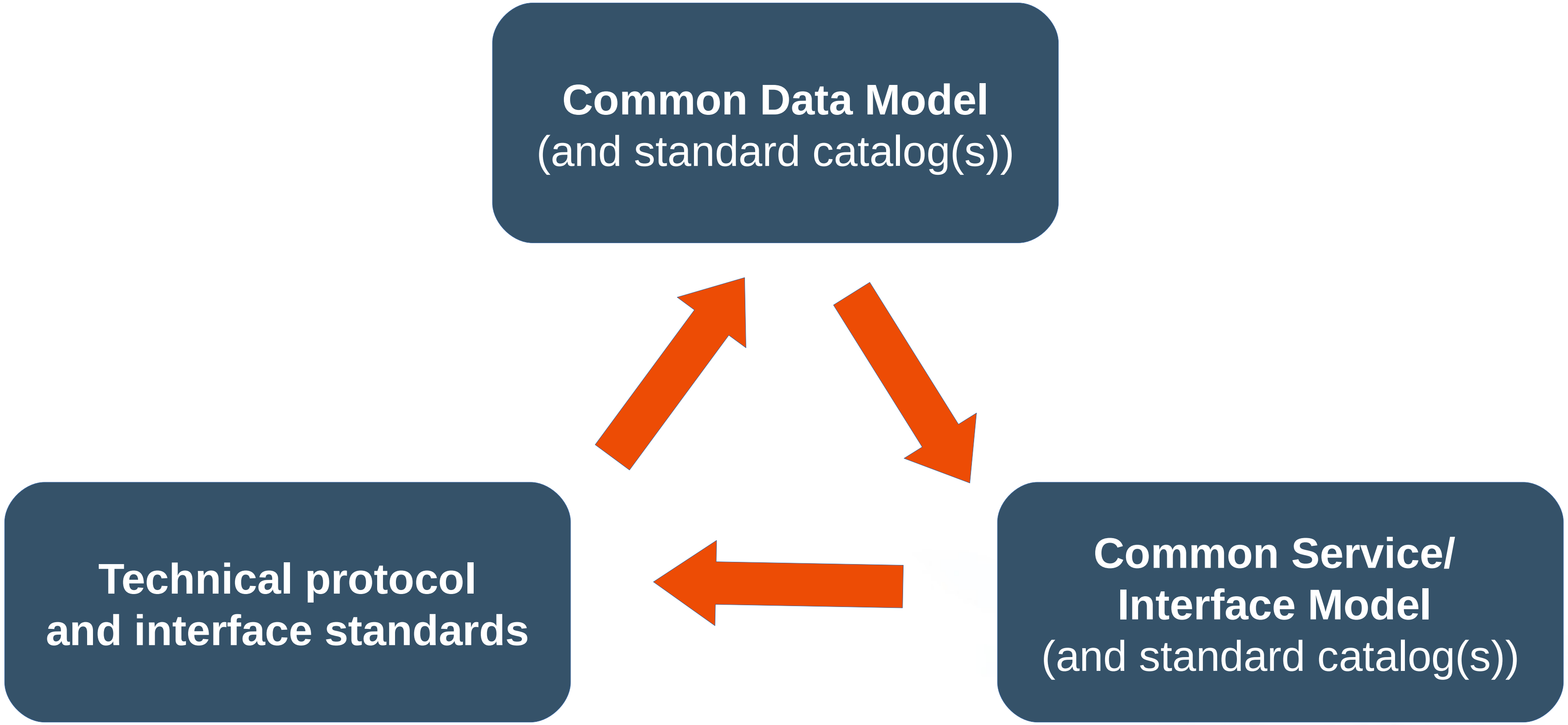
Recent History (abbrev.)

After the positive response a number of activities were set in place:

- We defined the initiative to be *starting a conversation* beyond the boundaries of individual industry consortia, to build a fully global conversation around **standard interfaces**, **data models** and other similar agreements.
- A renewed collaboration has been established between **W3C** and **GENIVI** around important parts of the CVII initiative. W3C and GENIVI are as a first step *initiating the conversation*, and collecting feedback on how it may be best organized **across the whole industry**.
- Two individual **OEM round-table** meetings established the initial needs among car manufacturers.
- Strategic first-level discussions have been had with industry organizations involved in definitions of vehicle data catalogs and data transfer technologies, including **SENSORiS, OmniAir, eSync alliance.** A request to **JASPAR** and others is pending.
- After the OEM-only discussions, outreach to the whole industry supplier ecosystem is now under way with information, workshops and individual meetings.

What is the Common Vehicle Interface Initiative?

Deliverables:


Common Data Model
(and standard catalog(s))
Common Service/ Interface Model
(and standard catalog(s))
Technical protocol and interface standards

Ongoing and future Industry Collaboration

CVII is not just another *competitive* technology proposal to either accept or reject.

CVII is different:

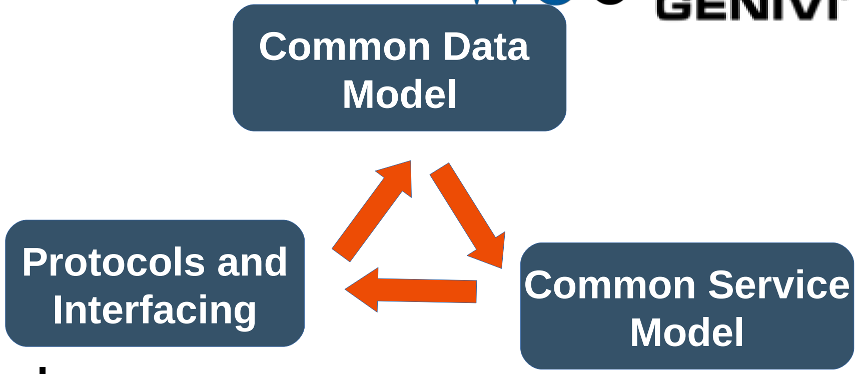
1. **The goal of CVII requires a *wide* industry collaboration**

2. **Several *projects*, *interest groups*, and *standards organizations* should be part of a shared movement.**

Ongoing and future Industry Collaboration (continued)

- **W3C and GENIVI** have a long history of producing industry-wide connectivity standards.
- GENIVI has produced technologies that connect to **AUTOSAR® Adaptive** platform, and has ongoing leadership-level discussion about future collaboration
- **W3C** has a strong liaison tradition, and the Automotive and Transportation groups are in contact with working groups including:
  ISO TC 22 – developing ExVe
  TC 204 Intelligent Transportation Systems and the working group for **ISO 20077 Extended Vehicle (20078 = Web Interface)**
- The existing **GENIVI Cloud and Connected Services** (CCS) project studies and coordinates with previous work in the area of data standards, including several government funded projects (**Automat/CVIM**, **NEVADA** from **VDA**) and industry-initiatives (**SENSORiS**, **ISO 27077**)
- Contacts established with **SENSORiS, eSync Alliance** and **OmniAir**
- GENIVI & W3C recognizes that the execution of CVII needs to further connected.
- **CVII** includes outreach for collaboration with **additional industry partners** to form a joint *initiative* to define a single data/service model & organize corresponding catalogs
- This week we hope to start up discussions with **JasPar** and **others**

What is the Common Vehicle Interface Initiative?



Common Data Model

Common Service Model

Protocols and Interfacing

- An *invitation* to the automotive industry to discuss fundamental standards-related issues that will accelerate development and business value
- A *continuation* of the existing movement towards "**A common data model**", where **Vehicle Signal Specification (VSS)** is a driving example
- An *extension* to define a standard model also for services & interfaces
- A *collaboration* to define associated protocols (e.g. **W3C VISS v2.0**) and interfacing technologies to *make use of* the data/service model in real systems
- A *discussion* on *where* **standard interfaces** are most useful for *vehicle and cloud*
- A *movement* towards unification of **fragmented ecosystems** that inadequately address only *part of* the vehicle-data and services problem, and not in concert.

Background: Automotive development trends

- The **software market place** = interoperable components are as important as ever!
- The **Service Oriented Architecture** trend enables very useful agility and flexibility, but does *not* eliminate integration work. Standard interfaces do.
- Consolidation trends (Virtualization / Central ECU / Compute ECU) might require diverse systems to work even more closely together
- Investment into **shared infrastructure** (Smart City, V2X) drives the need for standards
- We see the “**CVII**” desire clearly (multiple companies, multiple similar conference talks), but not coordinated yet
- Tech partners including *non-automotive* data and *cloud-oriented* companies (**Amazon**, **Microsoft** and **start-ups**) need interoperability standards for data and cloud solutions
- Business opportunities for vehicle data can only **reach their full potential** by leveraging common and scalable data standards.
- **We believe as a conclusion that CVII is a necessary conversation in our industry!**

Welcome to our workshop leading personalities! (Tuesday)

- John Schmotzer, **Ford Motor Company**
- Jeff Jaffe, CEO of **W3C**
- Ted Guild, **W3C** Automotive and Transportation
- Christian Heissenberger, **Bosch**
- Daniel Wilms, **BMW**
- Sebastian Schildt, **Bosch**
- … and all workshop participants
- + Benjamin Klotz, **Eurecom (Thursday)**

Agenda - Tuesday (Topics and leaders)

- Needs for in-vehicle standardization, and thoughts on CVII (John)
- W3C: The importance and success of Web Standards (Jeff)
- ...discussions

== Networking Break ==

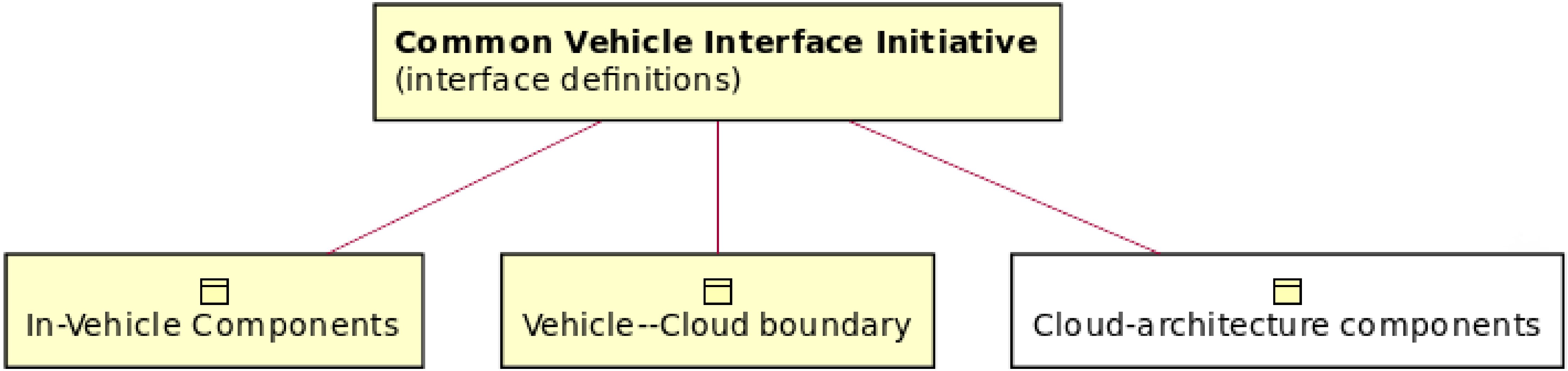
- Release v2.0 of VSS (Daniel)
- The potential of a common exchange language (Christian/Sebastian)
- Building the Technology stack, W3C Auto opportunities (Ted)
- Technology stack examples – Bosch working code demo (Christian/Sebastian)
- ...discussions
- Organizing the CVII – what’s next?
- (+ Backup topics, if time permits)

Detailed aspects and considerations

Recap of OEM Roundtables

- First GENIVI/W3C Joint OEM Roundtable on Common Vehicle Interface Initiative (CVII) held on 16 July w/ 10 OEMs present
  - Intent was to launch an industry-wide, OEM-led dialog on the benefits of joint development of common vehicle data models, access protocols and standard interfaces in the **entire inter-connected system of vehicle plus cloud.**
  - Supportive statements for the initiative provided by **Ford**, **BMW** and **JLR** Additional support voiced by **Volvo Cars**
- Second OEM Roundtable was held 24 September
  - The RT raised questions about scoping the initiative to each OEM
  - A full document that delves into the main ideas of the initiative has been created, using the outcome from the roundtables.
    (to be distributed to all interested parties)
  - This document combines illustrations with open questions for OEM feedback
- The initiative content does NOT define specific potential business value, only general opportunities

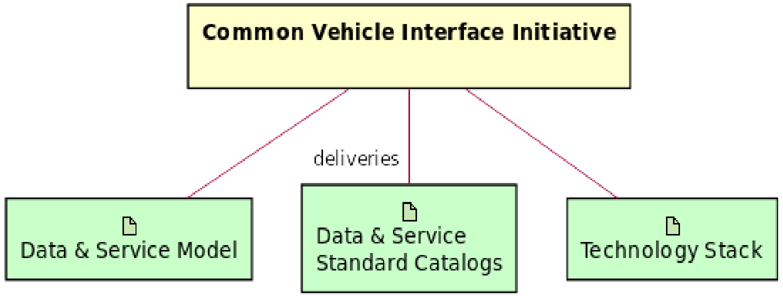
CVII Scopes



Common Vehicle Interface Initiative
(interface definitions)
In-Vehicle Components
Vehicle--Cloud boundary
Cloud-architecture components

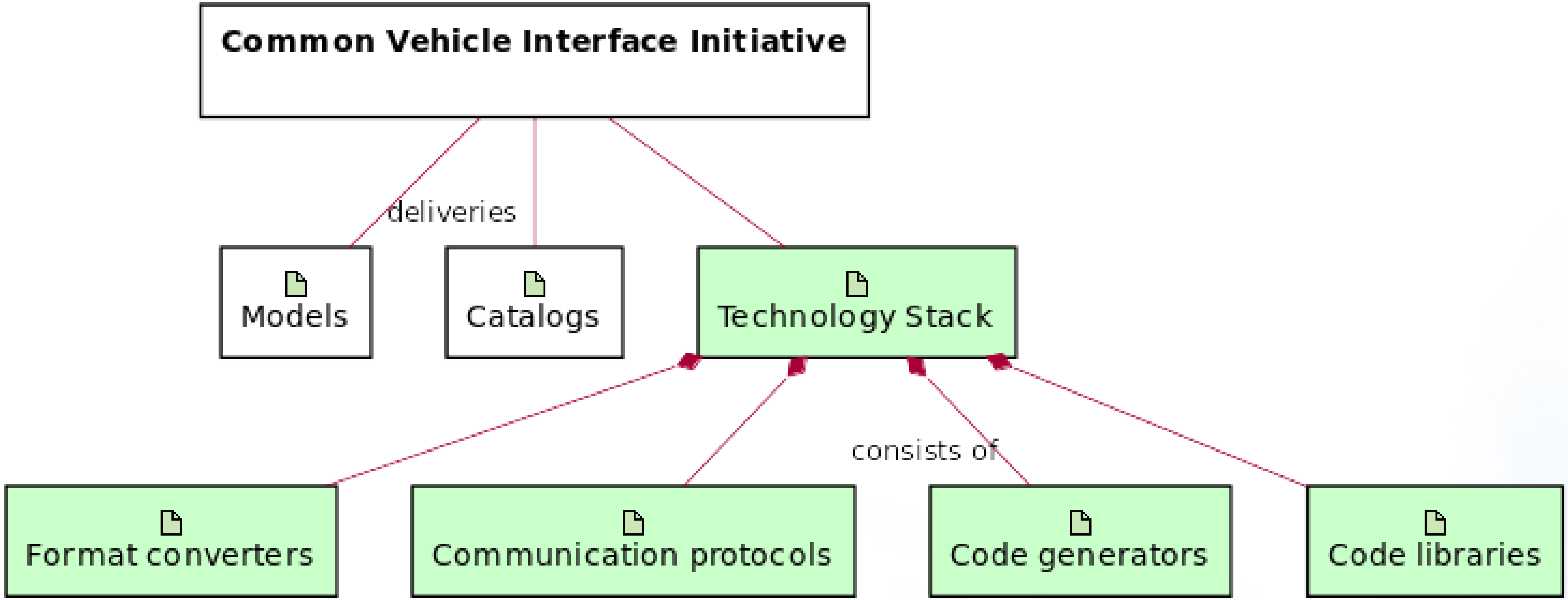
- Ongoing discussion points:
  - Are all these aspects are in scope?
  - Which is the primary interest (if any) for each company?
  - Should we explore the decomposition of cloud-architecture, including standard APIs and the technology stack of the cloud, as a separate topic?

Initial definition of terms (1)

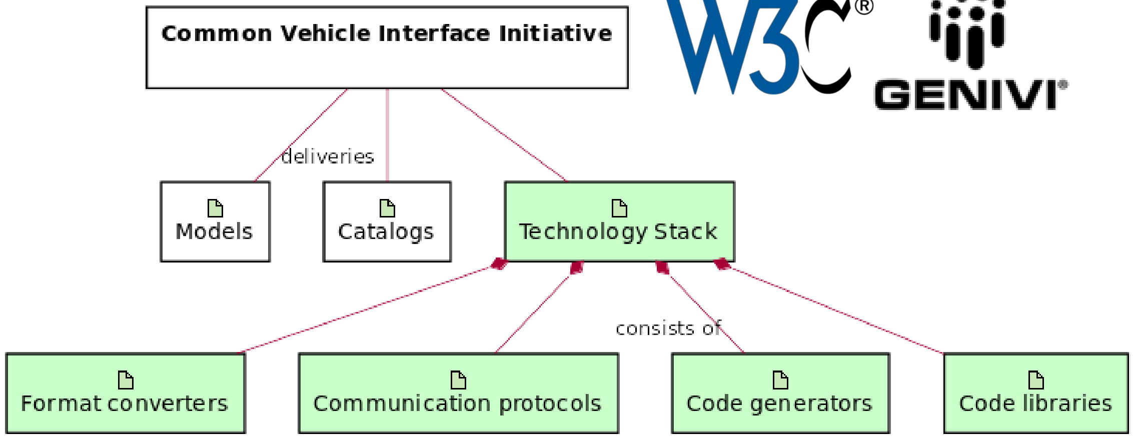

Common Vehicle Interface Initiative
deliveries
Data & Service Model
Data & Service Standard Catalogs
Technology Stack

- **Model (Rule Set)** = Rules for how to write a data or service definition. The model defines the syntax/format of expressions and their meaning (expected behavior). It specifies a set of metadata that is required to be included in any conformant list of data/interfaces. It defines the available datatypes to be used, and other similar rules.
- **Catalog** = An actual instance – a list of definitions. It is a collection of specified data items and Service APIs. The definition must follow the rules of the Model.
    - Sidenote: The word **Taxonomy** has been used when focusing more on the hierarchical nature and organization of data or services. The **Catalog** is however our name generic name for any (model-conformant) "list" of items.**
- ***Standard* Catalog** = A specific common and industry-shared catalog of items** expected to be provided by all implementations
- ****Items** may here refer to definitions of data, **or** definitions of services and interfaces (APIs). Both data and interfaces are *separately* in-scope for CVII activities.

Initial definition of terms (2) – Technology stack examples



Initial definition of terms (3)



By “**Technology Stack**”, we mean:

Any technology items involved in processing the agreed common data and interface models.

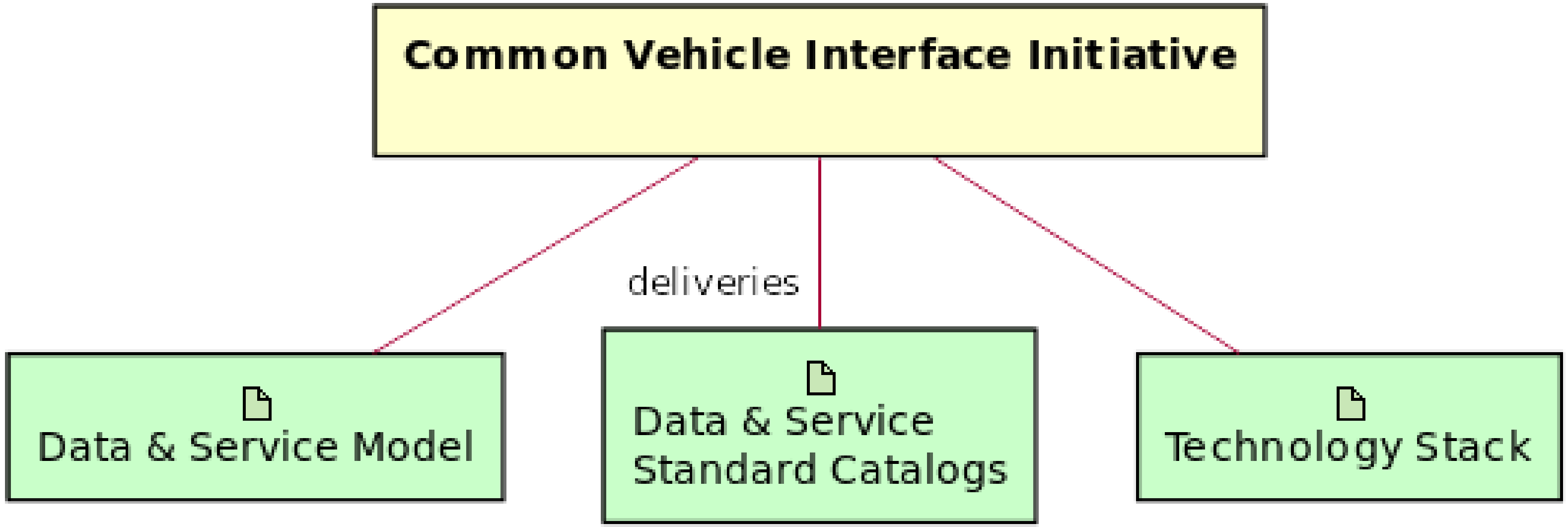
- This is primarily software definitions.
  Hardware is not explicitly excluded, but likely not the primary focus.
- It includes: **translators**, **bindings**, **tools**, **protocols**, **components**, **code-libraries** and other implementations.

By definition of the **CVII Technology Stack** we mean:

To agree on chosen technologies, and/or to develop those technologies.

The deliverables that define the Technology Stack are either **specifications** or **implementations** (to be decided within the project)

Outlook for CVII Deliverables


Common Vehicle Interface Initiative
deliveries
Data & Service Model
Data & Service Standard Catalogs
Technology Stack

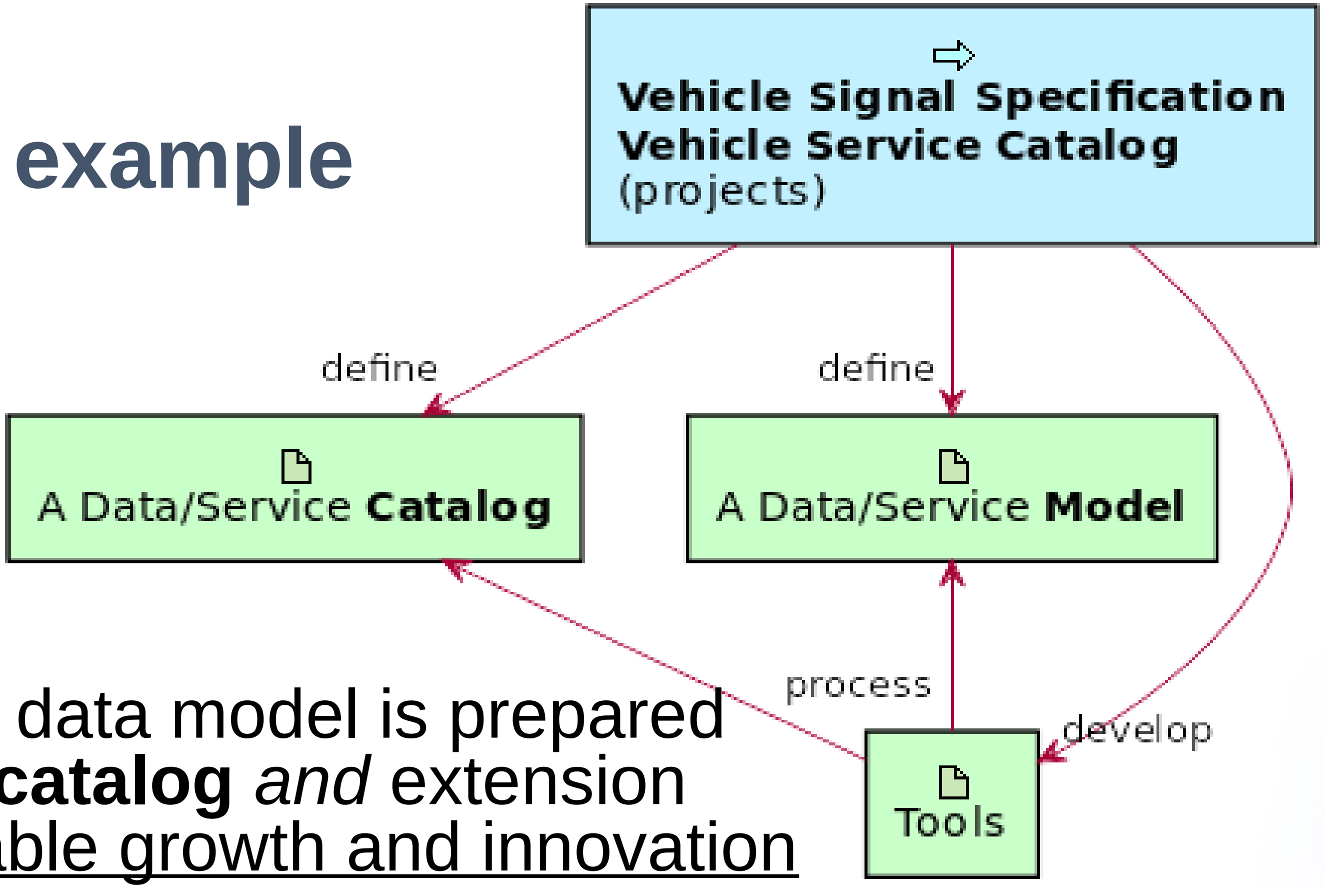
- The single **Data Model**
- The single **Services/Interface Model**
- *At least one* standard **Catalog** for **Data**
- *At least one* standard **Catalog** for **Services**
- A **Technology Stack** definition (collection of software & standards)

W3C development for the Technology Stack Standard protocols and implementations

- **The W3C Automotive Working Group (WG)** develops web-protocols for accessing vehicle data and is working on an more capable successor version to **VISS,** which already distributes VSS-defined data in production vehicles.
- The WG is starting to define the equivalent protocols for remotely-accessible services (as defined by the common service model)

- **W3C Automotive and Transportation Business Group** fosters development of transportation standards and conventions that provide additional guidance for the CVII activities, such as:
  - Development of **VSS ontology** (**VSSo**) enabling AI processing potential
  - Promotion of established solutions from other fields (Web-of-Things, Spatial Data on the Web, ...)
  - Best practices for automotive applications

Model vs. Catalog
with VSS (& VSC) as example

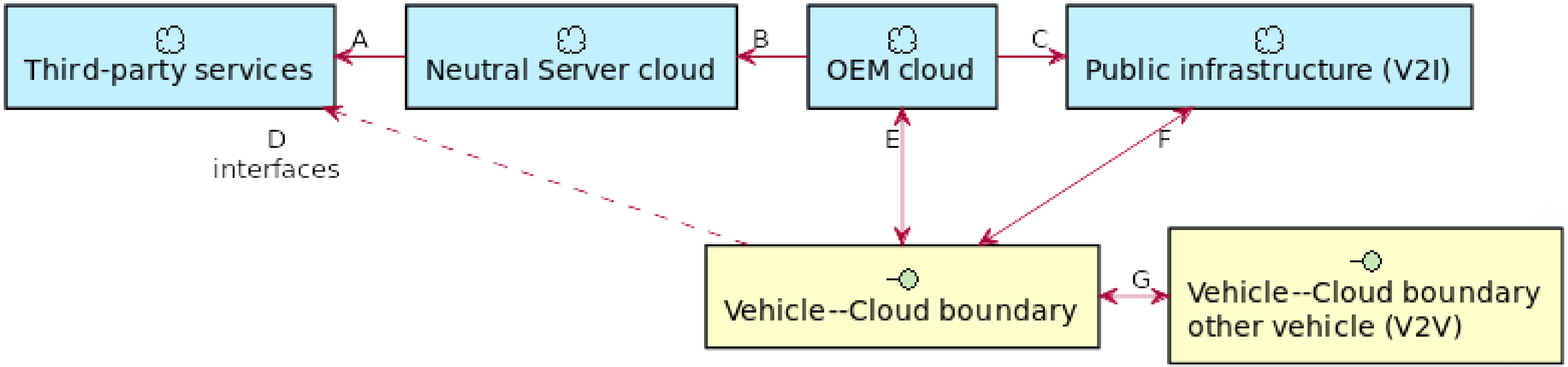


- VSS* as proposed common data model is prepared to support both a **standard catalog** *and* extension catalogs (proprietary) <u>to enable growth and innovation</u>
- But the agreed upon ***model (rule set)*** should always be common!

→ A common model is required for the *technology stack* to be shared, and for successful interconnection of parts across the whole system. The common model guarantees that there is a common understanding among industry partners.

*(*the exact same reasoning applies for VSC)*

CVII – usage and impact (1)



“Typical” cloud architecture

Which interfaces are most appropriate for industry standards?
Which interfaces does your company want to focus on?
How do we best organize the responsibility between *projects & consortia*?

What is VSS (Vehicle Signal Specification)?

- A mature (many years in development) data model & catalog which is used in vehicle production and W3C standards
- Fully open-licensed
- It is therefore a proposal for the industry-standard model deliverable from CVII
- VSS is a project that deals with **both**:
  - 1) **Model (Rule Set)** for vehicle data definition
  - 2) A proposed **Standard Catalog**
- It is important that the project separates model & catalog concepts, even if it has been driven under a single name (VSS) up to this point:
- These two aspects are individually usable for CVII, and of course open for adjustment
- The VSS data **model rule set** is *extensible* and *flexible* and has been shown to be able to model a very wide variety of data categories. Do not be fooled by the word “Signal” in its current name. VSS has evolved into a proposal for a widely applicable CVII common data model. In addition, where the expressiveness of VSS ends the VSC model can take over.
- (In the unlikely case you have never heard of VSS before, please seek further information beyond this single slide)

What is VSC (Vehicle Service Catalog)?

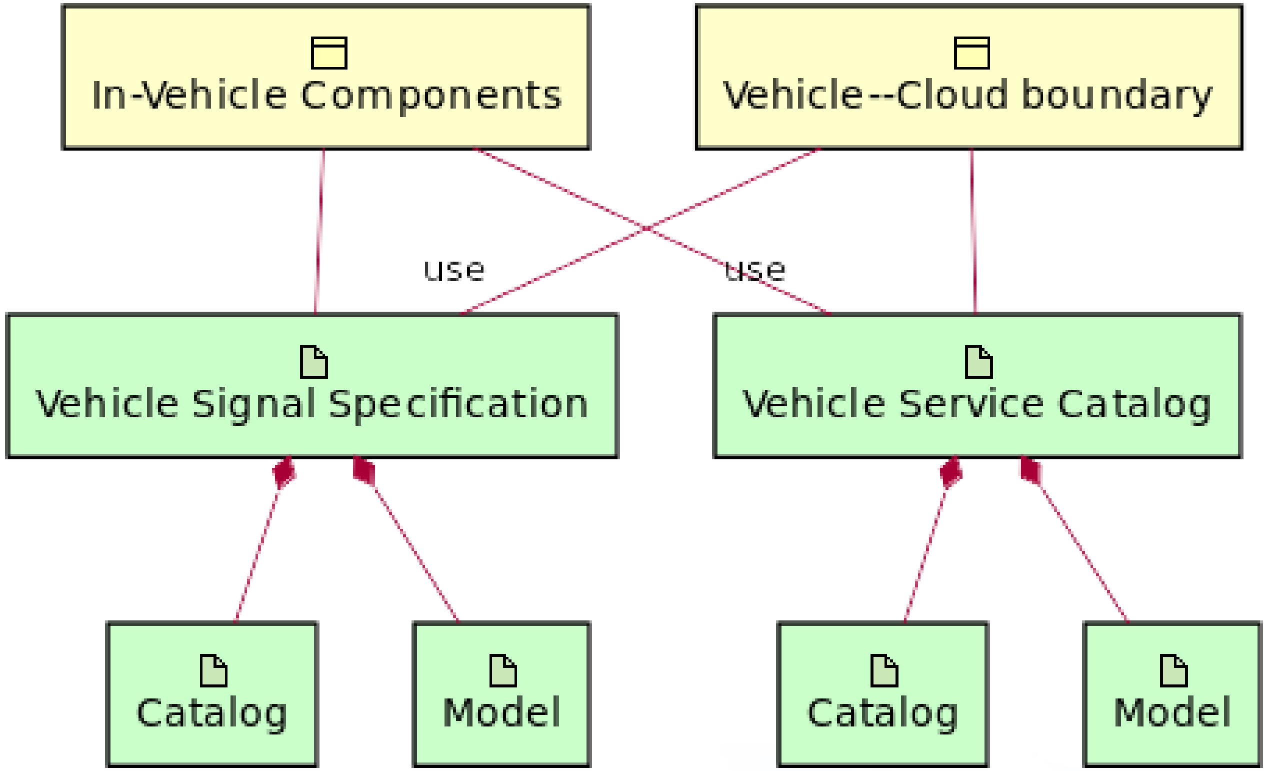
- A new definition in the same spirit of VSS
- Services are collections of Interfaces. Interfaces (**APIs**) are definitions of functions/procedures that can be activated.
- VSC started with the intention of defining remotely-callable functions, a.k.a. **Remote Procedure Call***, meaning services that vehicles expose to be activated from outside the vehicle (with the right security credentials of course) from a Cloud-connected infrastructure*.
- However, VSC uses a full featured interface language - in other words it is recommended to model *any* interfaces at *any* level that is desired, including between subsystems inside the vehicle, or even between software components.
- VSC is using a VSS-like YAML-based format but retains Franca IDL compatibility
- VSC is therefore *also* a proposal for *the* industry-standard model deliverable from CVII
- Like VSS, **VSC** is *also* a project that within it deals with **both**:

  1) **Model** for vehicle services definition

  2) A proposed **Standard Catalog**

*In traditional Computer Science literature the term RPC is often described as a function invocation between processes, inside of a single operating system, but there is no intention to be overly formal here.

CVII – usage and impact (1)



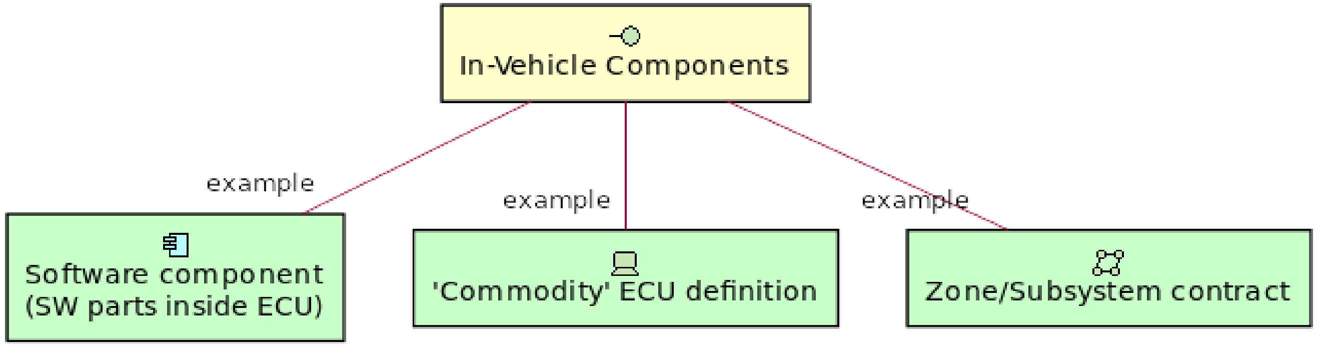
Common models will have the most impact if used both inside and outside the vehicle. <u>Minimize “translation code”!</u>

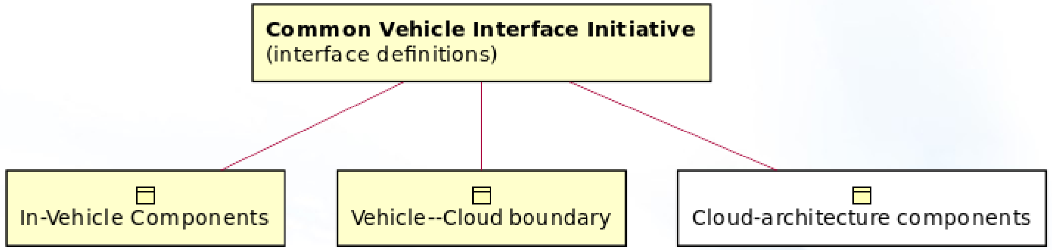
CVII – usage and impact – inside vehicle

Most likely the whole electrical architecture behavior is already today defined using primarily two aspects: (with different methods *in each company*)

1. Provided/consumed “data”
2. Provided/consumed functions/interfaces/services.

What if a **common data and service model** is used for all system behavior? How could this not create huge synergies across the development ecosystem (OEM/Tier-1/Software House/Innovator/Tester/Cost Estimator/Analyst)?



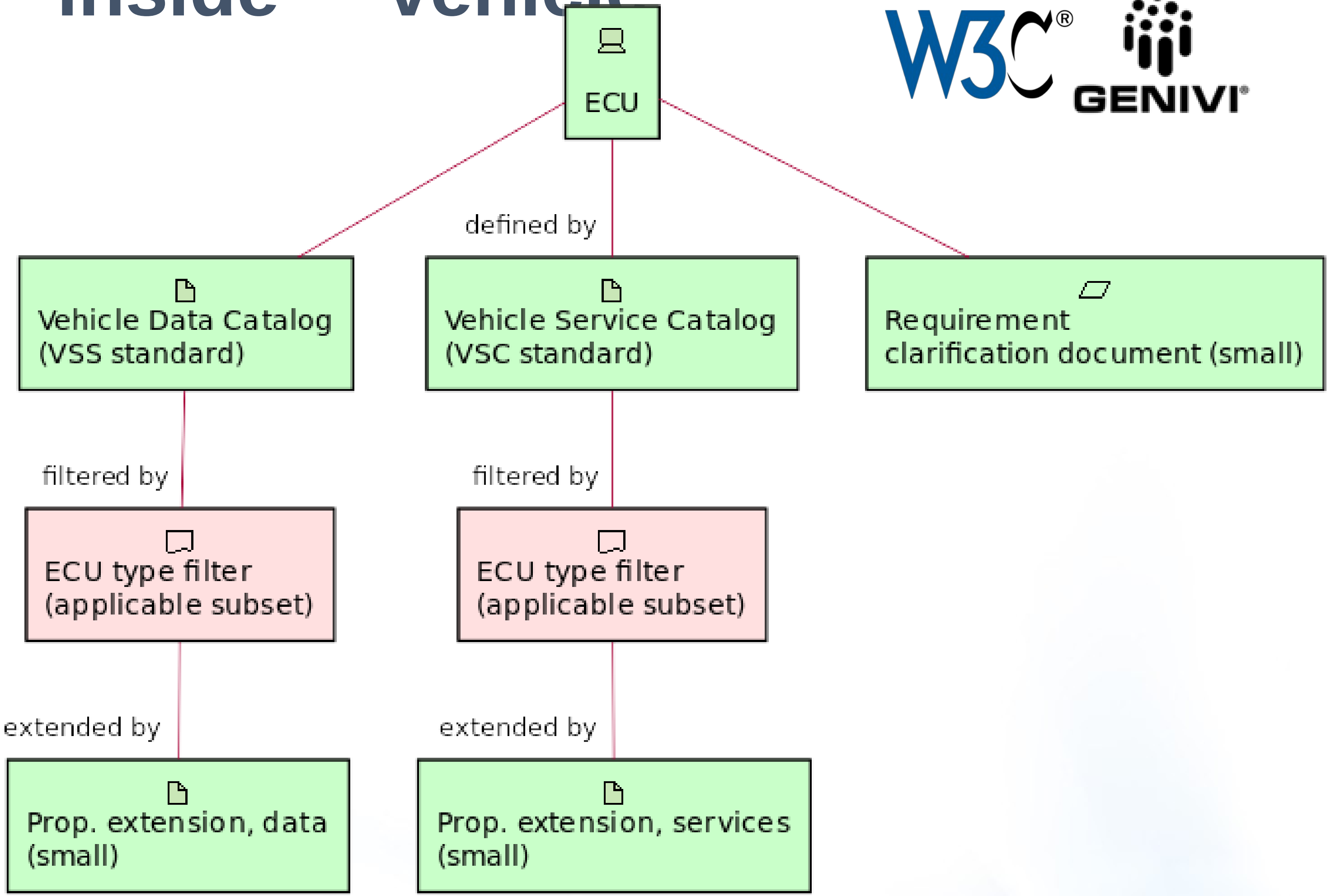


CVII – usage and impact – inside vehicle

Imagine a situation where ECUs were <u>not</u> specified using huge, unique, proprietary text documents.

…instead by simply selecting **Data** and **Services** from a **Shared catalog** – thereby defining the <u>major part</u> of the ECU function directly from standards that all partners know well.

… and remaining data and functions, while proprietary, are still specified using the <u>same common model</u> principles.



ECU
defined by
Vehicle Data Catalog (VSS standard)
Vehicle Service Catalog (VSC standard)
Requirement clarification document (small)
filtered by
filtered by
ECU type filter (applicable subset)
ECU type filter (applicable subset)
extended by
extended by
Prop. extension, data (small)
Prop. extension, services (small)

How might it facilitate both the **quotation** and **development process** for both OEMs and Tier-1 suppliers? How might it enable an active market of Commercial Off-The-Shelf standard ECUs (a.k.a. **Commodity ECUs**)?

Initiative organization – existing projects

- **Next major Seminar / workshop will be in November**

- **These projects are already active. They *could* be (re)used as part of of future CVII project organization.**
- **If the Time Zone does not fit, the meeting time can be adjusted, or additional meeting added, or a rotating schedule set up.**
  - **W3C “VISS” protocol** (Data) – contact: ted@w3.org, Tuesdays 2000 CET
  - **W3C “RPC”** (services) – contact: ted@w3.org, Biweekly Mondays 1900 CET / 1100 AM PST
  - **VSS data model & catalog project call**, Tuesdays 1930 CET – contact daniel.dw.wilms@bmw.de or gandersson@genivi.org
  - **GENIVI Cloud & Connected Services (CCS) project**, Mondays 1600 and Wednesdays 1700 CET

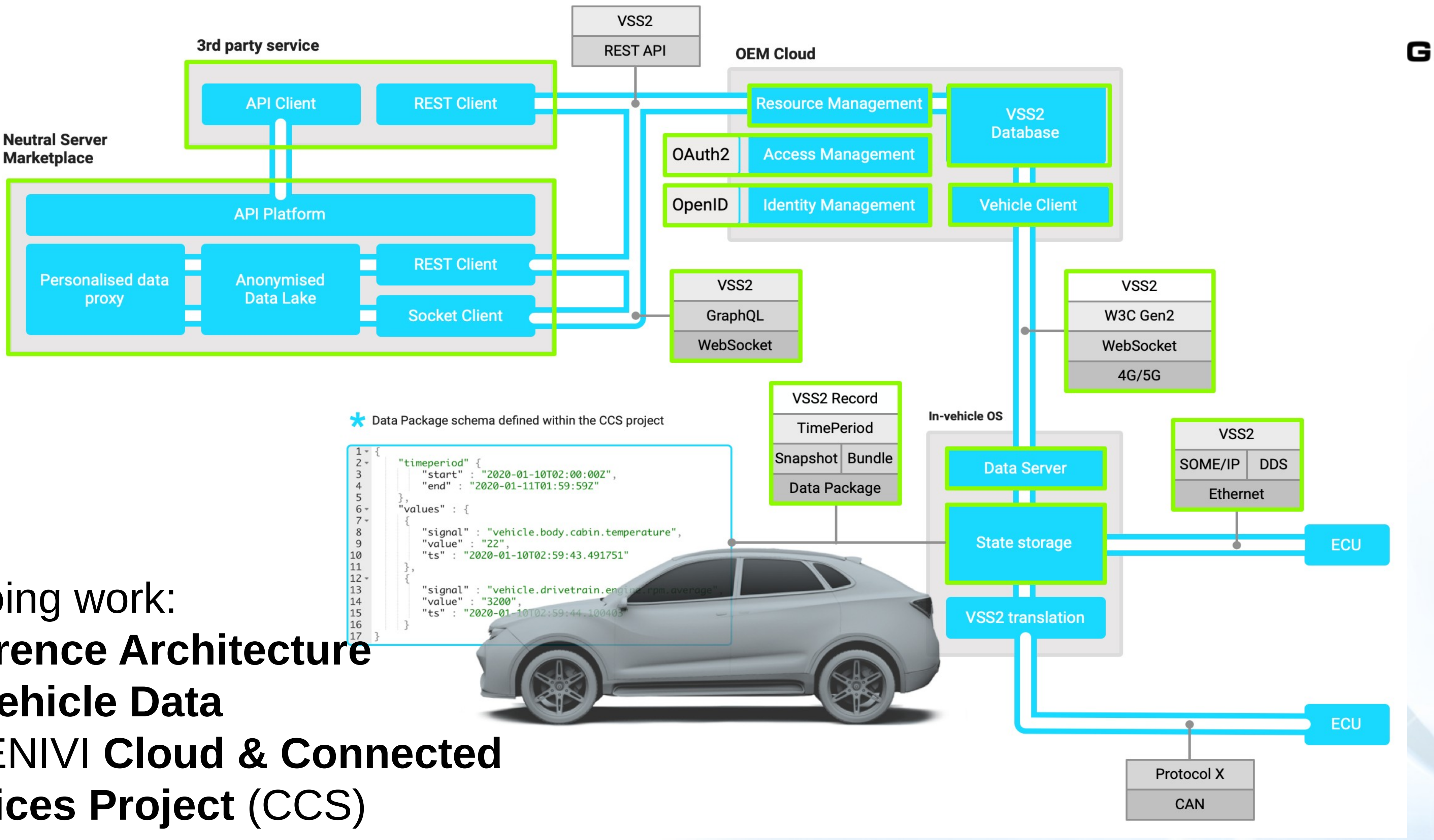
- New project-related calls (to be planned) – e.g. CVII all-hands, and specific topics

- General CVII questions: Email: ted@w3.org & gandersson@genivi.org, !
- GENIVI Membership questions: Email scrumb@genivi.org !

More details, associated projects, etc.

[Related projects]

Communication Framework draft v5



* Data Package schema defined within the CCS project

```
{
    "timeperiod" {
        "start" : "2020-01-10T02:00:00Z",
        "end" : "2020-01-11T01:59:59Z"
    },
    "values" : {
     {
        "signal" : "vehicle.body.cabin.temperature",
        "value" : "22",
        "ts" : "2020-01-10T02:59:43.491751"
     },
     {
        "signal" : "vehicle.drivetrain.engine.rpm.average",
        "value" : "3200",
        "ts" : "2020-01-10T02:59:44.100403"
     }
}
```

Ongoing work:

Reference Architecture for Vehicle Data

in GENIVI **Cloud & Connected Services Project** (CCS)

[Related projects]

Another related project:

Showing the usage potential of system-wide **common** data model (**VSS**) in Android Automotive operating system which is an example of one among many **specific** systems. The AASIG is promoting the value of applying the common data standard also to such specific environments.

- See GENIVI’s Android Automotive Special-Interest-Group (SIG) / Vehicle HAL topic, for more details.

Vehicle Signal Specification (VSS)

- Developed for around 4-5 years already → v2 release in October
- Defines **model** (description syntax, rules, datatypes, semantics)
- Proposes a **standard catalog** (std. signals, organization, taxonomy (hierarchy))
- VSS fully adopted for the official **W3C** automotive data access protocols (REST/WebSocket)
- Exists: **Partial Technology stack** (software, translations, protocols, bindings)
- **VSSo –** ontology extension, could support big-data / Artificial Intelligence usage
- **VSS best choice(?) for a common standard:**
  - See next slide for reasons
- What about alternatives?
  - 1) Remember, the common VSS standard is proposed only as the source format and to define the associated rules and behavior.
    Translations to many other formats are simply part of a technology stack setup!
  - 2) What other alternatives? Which standards may apply for creating the *common data model*? (not a rhetorical question – do you know any good alternative we ought to consider?)

Why is VSS (& VSC) the most appropriate “source format”?

- The selected format should be a plain-text, line-oriented format, like VSS:
  - Any text editor – **no special tooling** for editing! Uses well established “YAML” meta format
  - The line-oriented YAML format looks like any normal logical hierarchical document would look
  - Is **easy to read and write**, also by “non-programmers” ← **IMPORTANT FACTOR**!
  - A line-oriented format (YAML) promotes handling VSS like source code (version control in git repository, patches, change history, ...), better than syntax-oriented (JSON, XML, …) which are less convenient.
  - YAML still has formal meaning, machine-processable input, and it is easy to write and extend tools.
- **VSS** is plain-text YAML, has *several years of development*, and includes:
  - Easy, obvious hierarchical model
  - Simple and recognizable data types
  - Reasonable modeling power for signals/data (e.g. multiple instantiation of a single definition)
- **VSS is extensible using VSS-layers.** Adds unique deployment information for different target environments. Adds your own (proprietary) signals and/or potentially local modification to common catalog if required
- VSS is only the **source** and **model** of our information
  → Already proven conversion to other formats (including JSON, GraphQL, XML, Franca IDL,...)
- **W3C** develops official protocols for web-access to data with **VSS as the primary data model**

Vehicle Signal Specification (VSS) – Catalog

- The .id file has a flat list of signals – it now contains 1770 lines.
- But there is repetition in this signal tree where some subtrees are identical (same sensors on each wheel, each door, left/right pairs, etc.)
- The VSS definition is capable to create “instances”. In other words to define such repeated sections without explicitly repeating them.
- In addition to instances, the definition is in multiple files, and files can be “included” at multiple levels.
- As *yet another modeling feature*, the total data definition can *also* make use of *VSS-Layers* (described elsewhere).

Vehicle Signal Specification (VSS) – example signals

Our most important discussion topic is the model, but also the catalog.
Here are just a few examples from the proposed **standard catalog**:

Vehicle.VehicleIdentification.Brand
Vehicle.VehicleIdentification.Model
Vehicle.VehicleIdentification.Year
...
Vehicle.DriveTime
Vehicle.TravelledDistance
Vehicle.TripMeterReading
...
Vehicle.AmbientAirTemperature
...
Vehicle.AngularVelocity.Roll ,Pitch ,Yaw
...
Vehicle.RoofLoad , cargoVolume
Vehicle.Powertrain.PowerSource.CombustionEngine.Displacement
....Configuration
....MaxTorque
....FuelType
Vehicle.Powertrain.PowerSource.ElectricMotor.MaxPower
Vehicle.ADAS.CruiseControl.SpeedSet ,Error ...
...
Vehicle.ADAS.LaneDepartureDetection.Warning

Vehicle.Powertrain.Transmission.GearCount
Vehicle.Powertrain.Transmission.DriveType
...
Vehicle.Body.Horn.IsActive
...
Vehicle.Body.Raindetection
...Windshield.Heating.Status

Vehicle.Body.Lights.IsHighBeamOn
Vehicle.Body.Lights.IsLeftIndicatorOn
...
Vehicle.Cabin.RearShade.Switch
Vehicle.Cabin.HVAC.Station.FanSpeed
Vehicle.Cabin.Infotainment.Media.Volume
Vehicle.Cabin.Infotainment.Navigation.DestinationSet
Vehicle.Cabin.Sunroof.Position
Vehicle.Cabin.Door.Window.Position ,ChildLock, ...
...
Vehicle.Cabin.Door.Shade
Vehicle.Cabin.Seat.Recline
...
Vehicle.Chassis.AxleCount
Vehicle.Chassis.SteeringWheel.Angle
Vehicle.Chassis.Trailer.Connected
Vehicle.OBD.PidsA
Vehicle.OBD.Status.DTCCount
Vehicle.OBD.FreezeDTC

Vehicle Signal Specification (VSS) – Simple File Syntax

Snippet from **ElectricMotor.vspec:**

```
# This is a comment, anything can be written here, just like in program code.
# Here follows a signal definition:
- MaxRegenPower:
    datatype:  uint16
    type: attribute
    default: 0
    unit: kW
    description: Peak regen/brake power, in kilowatts, that motor(s) can generate.

- Motor.CoolantTemperature:
    datatype: int16
    type: sensor
    unit: celsius
    min: -50
    max: 200
    description: Motor coolant temperature (if applicable).
```

Sensor = readable, variable value
Attribute = “constant”
Actuator = writable

Note that due to the how the specification tree is created, the file can omit some part of the tree path. whereas the above snippet *actually* defines these two items written with their fully qualified name format:

```
Vehicle.Powertrain.PowerSource.ElectricMotor.MaxRegenPower
Vehicle.Powertrain.PowerSource.ElectricMotor.Motor.CoolantTemperature
```

Vehicle Signal Specification (VSS) – Layers

VSS tools can process and combine multiple definition files.

There is an explicit branch named **/private** where any tree can be placed.

However, it is also possible to use the VSS-Layer capability.

- **VSS Layers** can add metadata to the signal definitions
- **VSS Layers** are perfect to define a unique “deployment model” in which metadata that is only relevant for this particular usage environment can be added to the standard model.
- **VSS Layers** can add or remove signals, or even modify existing metadata.

As such, layers can be added and removed depending on situation, while keeping the main data model, and a main catalog definition intact.

While we promote the huge industry value of the standard data/service catalog, this feature explicitly prepares for the *inevitable* desire for some proprietary extension of available data and services.

By proactively allowing extension to the list of data (*services in the case of VSC), we can succeed in protecting the integrity (thus compatibility) of the model itself.

Vehicle Service Catalog (VSC)

- Early work so far.
- Reuses VSS principles, but a much extended model
- Same plain text, YAML* principle = low barrier for entry to program tools and transformations
- *However, semantic equivalence with **Franca IDL** is also expected
  → could hook into Franca ecosystem (code generators etc.)
- Same responsibility split GENIVI/W3C (data model vs. protocol)
- *"What about that VSS is so limited in its datatypes?"* (It is deliberately so)
  - => Solution: **VSC can define arbitrarily complex datatypes**
- → Proposal: Use VSC when complex data exchange is required
- → By pushing any need for complex data types to VSC instead,
  we can keep VSS more usable for simple "signal-like" data.
  Best of two worlds.
- (Up for discussion. A potential merge of VSS+VSC models in the future?)

Vehicle Service Catalog (VSC) – File Syntax

```
service:
  name: comfort
  description: A collection of interfaces for cabin comfort.
  datatypes:
    - namespace: movement
      description: |
        The units used to describe movement of the seats
      types:
        - name: movement_t
          type: uint16_t
          min: 0
          max: 1000
          description: |
            The movement of a seat component
        # Structs
        - name: position_t
          type: struct
          description: |
            The position of the entire seat
          members:
            - name: base
              type: movement.movement_t
              description: |
                The position of the base 0 front, 1000 back
            - name: cushion
              type: movement.movement_t
              description: |
                The position of the cushion 0 short, 1000 long
                ....
          type: enumeration
```

(continued)

```
interfaces:
  - include-interface: "xxx.yml"
  - name: seats
    description: Seats API
  commands:
      - name: move
        description: Set the desired seat position
        in_argument:
          - name: seat
            description: The desired seat position
              type: movement.seat_t
        out_arguments:
          ...
      - name: move_component
  methods:
      - name: current_position
        description: Get the current position of the seat
        in_arguments:
          - name: row
            description: The desired seat to row query
            type: uint8_t
        out_arguments: ...
  events:
      - name: seat_moving
        description: |
          The event of a seat beginning movement
        in_arguments:
```

Thank you!

Visit GENIVI:

http://www.genivi.org

http://projects.genivi.org

Contact W3C Transport and Automotive groups:

ted@w3.org