Artificial Intelligence, Social Computing and Wearable Technologies, Vol. 113, 2023, 279–289
https://doi.org/10.54941/ahfe1004200

AI-Enabled Semantic Modeling for Enhanced Boardnet Integration in Automotive Design

Frank Wawrzik, Johannes Koch, Sebastian Post, and Christoph Grimm

WG Design of Cyber-Physical Systems, Department of Computer Science, Kaiserslautern, Germany

ABSTRACT

This paper gives an outline of current knowledge-based design principles and processes for automotive and electronic design. We illustrate our design methods with a boardnet use case in automotive. Firstly, we introduce the modern challenges and architectures, and then we show how we can explore these different architectures in an interactive way to calculate the overall cable length of the architectures. Based on the example the contribution of the paper is twofold: First we elaborate on the differences and inconsistencies of the system modelers model and its knowledge-based counterpart. Secondly, we discuss the application of reasoning to improve the model and the speed of its development. Our tool for the exploration is both interoperable with SysML v2 and OWL.

Keywords: Boardnet design, Best practices, Knowledge engineering, OWL, Systems engineering, Systems modeling language

INTRODUCTION

AI and semantic modeling play a crucial role in the design and integration of modern boardnet architectures. AI-driven algorithms can optimize the placement and connectivity of electronic components, leading to reduced wiring complexity, improved energy efficiency, and enhanced data flow.

Semantic modeling allows for a higher-level understanding of the interactions between components, enabling a more comprehensive and efficient boardnet design. By applying AI to analyze large datasets and simulate different scenarios, designers can identify optimal configurations and make data-driven decisions.

AI-driven boardnet design offers numerous advantages over traditional methods. Firstly, it enables a more systematic and efficient design process, reducing development time and costs. AI algorithms can analyze vast amounts of data and simulate complex scenarios, leading to optimized architectures that meet performance, safety, and cost targets. Secondly, the integration of AI and semantic modeling allows for a holistic approach to boardnet design, considering various factors simultaneously. This results in better coordination between different vehicle systems and improved compatibility

with future technologies. Thirdly, AI-driven boardnet design enhances functional safety and reliability. AI algorithms can identify potential failure points and suggest redundant pathways or fail-operational strategies, making the boardnet more robust. Finally, AI-based design methodologies enable faster adaptation to changing regulatory requirements and customer preferences. By using digital twins, engineers can test different configurations and assess their compliance with regulations before implementing them in physical vehicles.

STATE-OF-THE-ART

Complex systems have long been subject to agile development, which prioritizes flexibility over fixed initial designs (Beck et al., 2001). This is extended to operations by the DevOps mindset, allowing for quick upgrades (Bass, Weber, and Zhu, 2015). Robust deployment, fail-safe operations, and automated testing are all components of a more stringent application known as iDevOps. However, there is no effective automated testing approach for sectors like the automotive. It can be difficult to close this gap, especially for safety-critical, complicated, and heterogeneous systems (Zimmerer, 2018). Adding hierarchy and composing components to handle complexity is one such method. Fundamental knowledge is provided through research in the area of hybrid system runtime verification (Frehse, 2006).

Reasoning in the context of the semantic web refers to the process to make inferences based on logical axioms. Hereby the axioms are restrictions on concepts, triples, object properties, datatype properties or others. Based on these restrictions explicit and implicit conclusions can be drawn. For example, by defining the "part of" property as the inverse of the "has part" object property, we now can traverse the part structure not only from bottom up but now also semantically from top to down. The missing class relationships get inferred by the reasoner. E.g. "system has part engine" now creates the additional triple "engine part of system" which was not stated explicitly before the reasoning process, but is now made explicit with an additionally entry in the knowledge base.

Reasoning in semantic models is still rare, because most ontologies are built as domain models and thus usually just describing classes and their relationships. Here, this works contributes by integrating various reasoning types actively in a use case boardnet model. In the caligraph ontology (Heist and Paulheim, 2022) mostly fill new instances with relationships with "hasValue" restrictions. The GENIAL! Basic Ontology (Wawrzik and Lober, 2021) has a tight ontological commitment to only allow concepts which are the exact words by differentiating it from related concepts. The human disease ontology (Shriml et al., 2021) is mostly a taxonomy with labels. It has relatively to its size few object properties with no complex expressions. In (Lu et al., 2023) the authors, like this work, use both OWL and SysML/OCL in conjunction to detect inconsistencies in their models, whereas we use it here to amend the knowledge construction process. They also translate SysML to OWL and use consistency check over state machines. (Riboni and Bettini, 2011) create an activity ontology and use it for reasoning in distributed environments

and for sensor data aggregation. (Batsakis et al., 2016) implement a temporal reasoning approach. For this they combine OWL with SWRL and implement complete and tractable reasoning with time instants and intervals and natural language expressions like "before" and "after." In the work of (Weser et al., 2020) a metamodel based on an ontology is presented, which is also rather a domain ontology, but which also demonstrates some reasoning to deduce implicit capability facts. In comparison to this work, they focus more on complex class expressions to assess the feasibility by checking if certain resources match the required manufacturing task.

BOARDNET DESIGN OVERVIEW AND GOALS

The boardnet in the automotive industry refers to the electrical system responsible for energy supply, communication, and data exchange between various electronic components within the vehicle. As automotive technology advances and the demand for smart, connected vehicles grows, the role of boardnet design becomes increasingly critical. The effective integration of electronic systems is vital for achieving optimal vehicle performance, safety, and efficiency. This section introduces the importance of boardnet design in modern vehicles and the challenges it faces.

In the past, vehicles had relatively simple electrical systems, consisting of basic lighting and ignition systems. However, with the evolution of automotive technology, the boardnet has become significantly more complex. It now includes power distribution, electronic control units (ECUs), sensors, actuators, infotainment systems, and advanced driver-assistance systems (ADAS). The boardnet serves as the central nervous system of the vehicle, facilitating communication between different subsystems and enabling seamless vehicle operation.

With the introduction of more electronic components, the length and weight of wiring harnesses increased substantially, leading to cost and space constraints. This issue prompted the development of multiplexed wiring solutions to reduce the number of wires required. As vehicles transitioned to electric powertrains and advanced driver assistance systems (ADAS), the demand for higher data rates and faster communication protocols emerged, further influencing boardnet design.

Requirements and Challenges

Boardnet design in modern vehicles is subject to stringent requirements and challenges. The increasing number of electronic components, varying voltage levels, data communication demands, functional safety, and electromagnetic compatibility are some of the key challenges faced by designers and engineers.

One of the primary challenges is to strike a balance between the growing number of electronic components and the weight, complexity, and cost of the wiring harness. As new vehicle features, such as electric powertrains and advanced driver-assistance systems, are integrated, the demand for power and data transmission increases, putting additional strain on the boardnet.

Ensuring functional safety and reliability is of utmost importance, especially in the context of automated driving. The boardnet must be designed to

handle fault tolerance, fail-operational capability, and redundant communication pathways to ensure continued functionality in case of failures.

Moreover, with the increasing digitalization and connectivity of vehicles, the boardnet faces cybersecurity threats. Preventing unauthorized access and ensuring data integrity are vital to protect drivers and passengers from potential risks.

Architecture

Before the advent of advanced technologies, traditional boardnet architectures relied on separate wiring harnesses and numerous control units for specific functions. Each control unit would manage a particular feature, such as engine control, air conditioning, infotainment, and ADAS, resulting in a high number of ECUs scattered throughout the vehicle.

While this architecture served its purpose for many years, it had several limitations. The proliferation of ECUs led to increased complexity, weight, and costs. Additionally, the lack of communication between ECUs and the extensive wiring harness posed challenges for further integration of new functionalities.

The automotive industry is currently undergoing a disruptive paradigm shift, triggered by new mobility and usage concepts. The emergence of automated and connected driving, coupled with the push towards electrification, has fundamentally altered the requirements for boardnet design.

Automated driving demands higher technical capabilities from the boardnet. It must handle the increasing volume of data generated by sensors and cameras while providing real-time responses to control actuators. At the same time, connectivity requirements demand faster and more reliable communication protocols to support V2X (vehicle-to-everything) communication.

To address the limitations of traditional architectures and support the requirements of automated driving, the concept of intelligent zonal and centralized boardnet architectures emerges. These new concepts aim to consolidate functionalities and reduce the number of ECUs, resulting in a more integrated and efficient system.

In a zonal architecture, multiple ECUs are integrated into larger modules, known as zones, each serving specific functions. These zones collaborate closely, leading to reduced communication overhead and better control of power distribution and data flow.

On the other hand, centralized architectures concentrate critical functions into a few high-performance controllers. By centralizing processing power, centralized architectures facilitate efficient data exchange, enable more sophisticated data fusion, and offer greater flexibility for future upgrades. Figure 1 exemplifies the architecture models.

SEMANTIC MODEL OF THE BOARDNET AND SYSTEM MODEL VS. ONTOLOGY MODEL

This section will first introduce our general conceptualisation of electronic and system engineering design in the next subsection. Based on that understanding we show a short semantic model of the boardnet. The motivation

of the work presented in this section is the following: In using our web-based tool we are both interoperable with SysML v2 standard as well as an ontology-based knowledge graph in a graph database. In the context of this work, we investigate the results of a system modeller who has been roughly trained in semantics and constructed the boardnet model of future car systems. This investigation takes place from the viewpoint of the knowledge engineer in order to assess the suitability to the graph database world. We will address various investigation points and discuss them in this chapter.

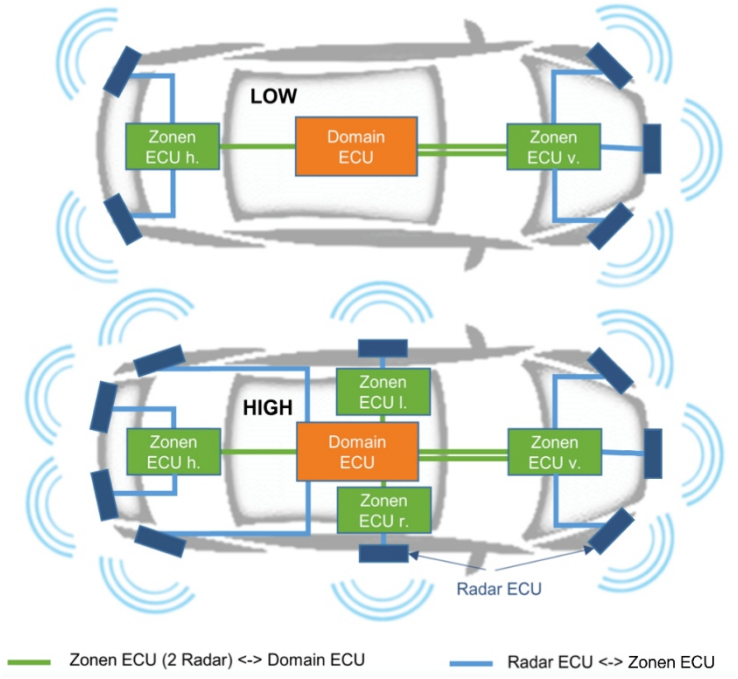

Figure 1: Centralized vs. zonal architecture of the boardnet for wire length calculation. Here showing 2 zones vs. 4 zones alternative.

GENIAL! Basic Ontology

The GENIAL! Basic Ontology (in short GBO) is the standard reference model for our knowledge base. It is a system engineering and electronic description in the form of an OWL file. It describes terms such as hardware, software, properties, functions. systems and its parts and their dependencies. It is a minimal ontology which still can describe most of what is relevant for the domain and is based on the upper ontology Basic Formal Ontology.

Boardnet Design

In previous works, we built the boardnet model and built our own constraint solver, which can calculate dependencies of the connected parts and properties in a forward as well as backward directed, meaning bi-directional way. In this way, we address the main challenge of exploring different architectures and estimating the overall cable length, cost and other parameters such

as data rates and the overall automotive context and their other properties. Figure 2 illustrates the challenge.

The boardnet model currently consists of

- Wires, Sensors (Radar, Lidar, Ultrasound, Cameras), Controllers (ECU's)
- Transmission Technologies (Ethernet, I2C, CAN), BUSes, Data Rates, Various Properties like Cost, Length, Latencies, Positions, Locations and others

Our overall automotive models currently consist of various libraries, such as

- Car Model, Mechanical Parts, Hierarchical Decomposition
- Detailed Sensor Models with Properties, Functions, Radar Model, Hardware Accelerator Model with Neural Net

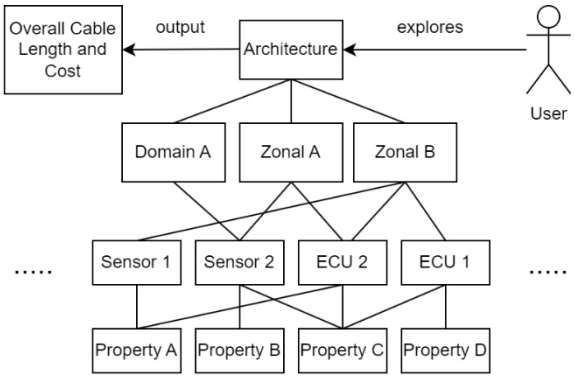

Figure 2: Agile exploration of the boardnet architectures with its properties and parts via the tool and user interface.

In the following rather than describing the model we focus on the differences between that a knowledge engineer sees to improve the consistency for the computer assisted use as well as the possibilities for reasoning with the model.

Naming and Terms

System modellers often name things intuitively, based on their background, experience and usage. In system models we sometimes find abbreviations, numbers and combination of words. The knowledge engineer on the other hand, carefully investigates known terms and their definitions at the beginning before putting the word/term/concept/class into the ontology. Often doing state of the art research on given ontology models.

In Figure 3 we see an example how connecting different components is done with wires. Here, we see every wire gets its own "semantic" name by indicating how it is connected as part of its name. Because we work at an interface and every class is an element it seems plausible to assign a unique name. Otherwise, it would not show up in the knowledge base. On the bottom part of the image, we can see how the wire between the zonal controller

and the front central controller is connected. They have a start and end point at the component.

```
WireZonalControllerFrontCentralController isA Wire.
WireZonalControllerRearCentralController isA Wire.
WireMidRangeRadarZonalControllerFront isA Wire.
WireLongRangeRadarZonalControllerFront isA Wire.
WireLidarZonalControllerFront isA Wire.
```

```
WireZonalControllerFrontCentralController hasA
    Component Start: CarComponents::ZonalControllerFront,
    Component End: CarComponents::CentralController,
    Component Wiretyp: WireTypes::Ethernet.

```

Figure 3: Connecting controllers and sensors with wires.

Resolution: In that case for the knowledge engineer it makes most sense to reside in the modeling paradigm and to assign the wires unique ids in combination with a name, but semantically to keep it a "wire." With the corresponding SPARQL query, the actual meaning of the wire is maintained by its connection, not requiring an entry in the ontology as a class implicitly stating the location, but an instance in the knowledge base as an assertion. This way the purpose of the ontology is retained, and we have no unnecessary "clutter." Here, the distinguishment comes from what an ontology class is supposed to represent. At the same time in our textual language, we must still refer to the modelling construct we want to modify.

Alignment With Vocabulary

The vocabulary of GBO serves as a basis for correct construction of the knowledge base as well as a general reference across domains and applications. As a modelling guideline for the developer also of our systems models, we discuss some aspects of the modelling result in Figure 4. Two basic classes of the meta model (of our tool SysMD) are "element" and "component." They are connection points to construct the model. Sinilar to SysML v2, we import the corresponding packages (e.g. ISO26262) and modules (e.g. CarComponents). "Car" is our package which then defines the elements, like the BUSes package on the right.

From the modeling perspective on the left of the figure we see the architecture defined as an element. The architecture in the semantic sense represents a type of boardnet. However, in the model we do not refer to is as a boardnet, but just a general architecture. Further the architecture/boardnet type is more accurately classified as a system in the ontology since it consists of controllers. The model element here functions as a maximum abstraction to simplify classification at the disadvantage of computational correctness. A similar occurrence can be found on the right side, where BUS is a component rather than a hardware part (meaning it is one hierarchy level down). The construction of correct taxonomies which are at the same time useful is a challenge.

```
Document imports SI, ISO26262.

Global hasA Package Car.
Car imports SI.
Car hasA Package CarComponents.
Car hasA Package WireTypes.
Car defines
    Architecture isA Element;
    ZonalArchitecture isA Architecture;
    DomainArchitecture isA Architecture;
    Wire isA Element.
```

```
Global hasA Package BUSes.
Document imports SI, ISO26262.

BUSes defines
    BUS isA Component;

    CAN isA BUS;
    CANFD isA CAN;
    CANXL isA CAN;

    Ethernet isA BUS;
```

Figure 4: Architecture and its semantics and imports.

Correct Semantics

Figure 5 shows how the property "wireLength" gets summed up over each connected component and forms a "TotalLength" at the system level. From the knowledge engineers' point of view the new property is allocated to the right hierarchy level, though it is not defined that it is a property of wires. Thus "Total Length" could be the length of anything and would not be well findable in the overall model.

```
Architecture hasA
    Value TotalLength: Length [m] = bySubclasses(TotalLength),
    Value TotalWeight: Mass [kg]  = bySubclasses(TotalWeight),
    Value TotalCosts: Currency [€] = bySubclasses(TotalCosts).
DomainArchitecture hasA
    Value TotalLength: Length [m] = sumOverParts(wireLength),
    Value TotalWeight: Mass(0..0.5) [kg] = sumOverParts(Wiretyp::specificWeight * wireLength),
    Value TotalCosts: Currency [€] = sumOverParts(Wiretyp::CostsPerMeter * wireLength)
                                   + sumOverParts(Start::plugCosts + End::plugCosts),
```

Figure 5: Total wire length on the system level.

REASONING WITH SEMANTIC BOARDNET DESIGN

We already implemented a variety of algorithms and functions in our language. As seen in Figure 5 "sumOverParts" propagates the values to the next higher hierarchies. Further we have basic arithmetic functions that form a constraint, see "TotalWeight" or "TotalCosts" in the figure. This subsection shows work on complementing our model with the expressiveness of the Ontology Web Language OWL-DL 2. We also show how the reasoner will fill missing information to aid the modeler and continue to build the knowledge base.

Investigating Application and Capabilities of SysMD

As can be seen in the example base model as well as in the previous section in Figure 4, basic taxonomic structures can be created with the SysMD language, they can be put into packages and documents, their properties, parts and functions can be described. Mathematical constraints can be solved quicker than with the Semantic Web Rule Language (SWRL) and are thus to be used preferably in case of mostly arithmetic. Our solver additionally supports backpropagation to allow for calculations in the opposite direction

simultaneously. In case of dominant semantics, rather than arithmetic, rules may be preferable. Since reasoning time is an issue, it makes sense to withdraw load from the reasoner as much as possible. Existential restrictions are partly mimicked as well. For example, temperature ranges from sensors can be inherited downward and further restrict certain sensor types and their allowed ranges.

Object Property Axioms

One of the first part of the work is to make the content of our knowledge base more searchable and spare modeling time. We connect sensors, controllers, cameras, wires and other components via the "is_connected_to_directly" object property. The wires are extra instances, since their properties are important for the architecture considerations, especially regarding the properties "length", "cost per meter", "data rate", "overhead per frame" and others. This object property has itself as an inverse and is non-transitive. It is further a subclass of "is_connected_to" which is transitive. This allows to ask the competency question CQ1: "What component is connected to which component?" According to the axioms, this includes the wires as well as the components.

The second competency question is CQ2: "What type of connection (e.g. ethernet, I2C, etc.) connects which components?"

Property Chains

Superproperty Of (Chains) in short property chains are used to couple object properties. In this way a characteristic can be transferred along certain relationships. We coupled the current environment temperature to the car instance as well as the current acceleration as part of the DevOps cycle. And chained the property_of axiom with the has_parts_directly axiom. In this way all the boardnet components have both properties explicitly stated as their system context.

Disjoint Classes

Disjoint classes are 1) inherited down from the upper ontology to separate and distinguish parts from properties and functions. And 2) from the GBO ontology to identify the exact hardware or software elements for unambiguous use. In the current scenario, using disjoint classes in combination with complex class expressions is not feasible for performance reasons and calculated by an external solver.

Complex Existential Restrictions

As being part of the TBox (Terminology Box, meaning concept part of the ontology), these expressions make most sense in the realm of definitions and controlling these definitions. The modeler in our use case neither has known the constructs nor has used definitions as is typical in system modelling. With complex definitions we can for example formalize qualitative and quantitative properties of properties and parts. Additional building control is for

example introduced with similar axioms like "domain_architecture Subclass Of: has_part_directly only (domain_controller or not zonal_controller)".

Rules

Here two rules are exemplified. Rule 1 is similar to the sumOverParts function in Figure 5. Rule 2 simplifies and automates the has_part hierarchy construction in the model and uses property chains.

Table 1. Sample rules of boardnet implementation.

SWRL Rule / OWL Axiom	Explanation
ns:mass(?m) ^ om-2:hasNumericalValue(?m, ?val) -> om-2:hasNumericalValue(ns:totalmass, ?val)	Adds all masses to the totalmass
has_part SuperProperty Of (has_part o is_connected_to)	Electronics components that are connected to each other are all part of its superpart

CONCLUSION

In this paper we showed how the system modeler can improve his design rationale to be well interoperable with computer understandable knowledge bases. We also showed how to utilize basic and advanced reasoning constructs to speed up the design process of modern boardnet architecture design. The boardnet design is a critical aspect of modern automotive development, especially in the context of automated and connected driving. By embracing AI-driven semantic modeling and intelligent architectures, automotive manufacturers can revolutionize boardnet design, leading to more efficient, safer, and adaptable vehicles. The integration of AI technology in boardnet design is expected to have a significant impact on the automotive industry's future, driving innovation and revolutionizing vehicle electronics. As this field continues to advance, further research and collaboration between academia and industry will be essential to harness the full potential of AI in boardnet design and beyond. With continuous advancements in AI algorithms, boardnet design will become more sophisticated and dynamic, ensuring that vehicles of the future are at the forefront of automotive innovation and safety.

ACKNOWLEDGMENT

This work has been supported by the GENIAL! project with funding from the BMBF under grant agreement No 16ES0865K.

REFERENCES

Bass, Len, Ingo Weber, and Liming Zhu (2015). DevOps: A software architect's perspective. Addison Wesley Professional.

Batsakis, Sotiris et al. (Nov. 2016). "Temporal representation and reasoning in OWL 2". In: Semantic Web 8, pp. 1–20. doi: 10.3233/SW-160248.

Beck, Kent et al. (2001). The agile manifesto.

Frehse, Goran (2006). "On timed simulation relations for hybrid systems and compositionality". In: International Conference on Formal Modeling and Analysis of Timed Systems. Springer, pp. 200–214.

Heist, Nicolas and Heiko Paulheim (2022). The CaLiGraph Ontology as a Challenge for OWL Reasoners. arXiv: 2110.05028 [cs. AI].

Lu, Shan et al. (Jan. 2023). "Detection of Inconsistencies in SysML/OCL Models Using OWL Reasoning". In: SN Computer Science 4. doi: 10.1007/s42979-022-01577-0.

Riboni, Daniele and Claudio Bettini (2011). "OWL 2 modeling and reasoning with complex human activities". In: Pervasive and Mobile Computing 7.3. Knowledge- Driven Activity Recognition in Intelligent Environments, pp. 379–395. issn: 1574–1192. doi: https ://doi.org/10.1016/j.pmcj.2011.02.001. url: https://www.sciencedirect.com/ science / article/pii/S1574119211000265.

Schriml, Lynn M et al. (Nov. 2021). "The Human Disease Ontology 2022 update". In: Nucleic Acids Research 50. D1, pp. D1255–D1261. issn: 0305-1048. doi: 10.1093/nar/gkab1063. url: https://doi.org/10.1093/nar/gkab1063.

Wawrzik, Frank and Andreas Lober (2021). "A Reasoner-Challenging Ontology from the Microelectronics Domain". In: Proceedings of the Semantic Reasoning Evaluation Challenge (SemREC 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Event, October 27th, 2021. Ed. by Gunjan Singh, Raghava Mutharaju, and Pavan Kapanipathi. Vol. 3123. CEUR Workshop Proceedings. CEUR-WS.org, pp. 1–12. url: http://ceur-ws.org/Vol-3123/paper1.pdf.

Weser, Michael et al. (2020). "An Ontology-based Metamodel for Capability Descriptions". In: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). Vol. 1, pp. 1679–1686. doi: 10.1109/ETFA46521.2020.9212104. 7

Zimmerer, Peter (2018). "Strategy for continuous testing in iDevOps". In: Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, pp. 532–533.9