www.cadanda.com
Computer-Aided Design AND APPLICATIONS

Streamlining Function-oriented Development by Consistent Integration of Automotive Function Architectures with CAD Models

Moritz Cohrs[1], Stefan Klimke[2] and Gabriel Zachmann[3]

[1]Volkswagen AG, Germany, moritz.cohrs@volkswagen.de
[2]Volkswagen AG, Germany, stefan.klimke@volkswagen.de
[3]University of Bremen, Germany, zach@cs.uni-bremen.de

ABSTRACT

A primary challenge in the automotive industry is the increasing complexity of modern cars caused by the high amount of vehicle electronics respectively vehicle functions which are implemented as mechatronic systems. A promising solution is the relatively new *function-oriented* development approach that focuses on the interdisciplinary development of such functions and which helps to handle the high complexity in automotive development. At this stage, however, a function-oriented development does not fully exploit the capabilities of *virtual technologies* which are fairly well-established technologies in the automotive product development. One reason in particular is that function-oriented data is not yet integrated with geometric CAD data. The authors' main contributions begin with an analysis of the data structures of *function architecture data* and *CAD data* and they provide a definition of the requirements for a consistent mapping of named data structures. Moreover, they also develop a meta-format that enables a system-independent description and exchange of function architectures. In addition, the authors carry out a prototypical implementation that shows the applicability of the proposed data integration approach and they derive new methods that can assist a function-oriented development. Finally, the authors evaluate these methods by means of actual use cases. Summarizing, their research focuses on the interdisciplinary integration of function architectures with CAD models to create synergies and to enable new, beneficial methods for the spatial visualization and utilization of such data.

Keywords: function-oriented development, virtual technologies, virtual prototyping, systems engineering, digital mock-up

1. INTRODUCTION

Today, the increasing complexity of modern cars is one of the primary challenges in the automotive industry [7],[12]. One significant driver of complexity is the high amount of vehicle electronics respectively vehicle functions, like *Park Assist*, *Dynamic Light Assist* or *Start-Stop Automatic*. Such functions are implemented as mechatronic systems, consisting of sensors, actuators and controllers. A function-oriented approach to development addresses the interdisciplinary implementation of such systems. This approach can complement component-driven development by extending the overall focus on functions rather than on single components and it provides a fundamental solution to handle the rising complexity in automotive development [7],[11],[17]. In addition, another significant complexity driver is the high amount of different variants due to different engines, transmissions, steering controls and markets and the possible configurations including multiple customizable options. For instance, in Germany, a VW Golf 7 is currently available with options for different engines, transmissions, assistance systems and more than 100 optional features resulting in a total of multiple thousand possible configurations only for the German market.

Virtual technologies are computer-based methods for the processing of virtual product prototypes [16] and are an important resource in the automotive product lifecycle management [10],[13],[18]. Moreover, virtual technologies help to master the increasing product complexity and they can be beneficial in many aspects like product quality, time-to-market and cost competiveness. At this stage, however, the capabilities of virtual technologies are not yet fully exploited for a function-oriented development

Computer-Aided Design & Applications, 11(4), 2014, 399–410, http://dx.doi.org/10.1080/16864360.2014.881182

Taylor & Francis
Taylor & Francis Group

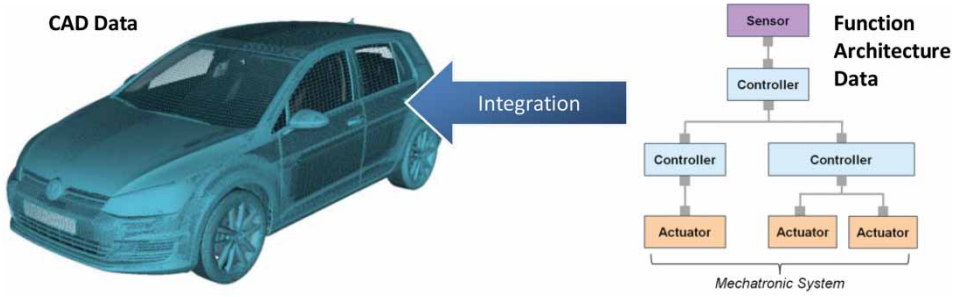

Fig. 1: Our approach integrates function architecture data with CAD data to create synergies and to enable new beneficial methods for the spatial visualization and utilization of function architectures.

because function-oriented data structures are not yet integrated with geometric CAD data. Consequently, our research focuses on the interdisciplinary integration of automotive function architectures with CAD models to create synergies and to enable new, beneficial methods for the spatial visualization and utilization of such data. Fig. 1 illustrates our approach of data integration.

2. RELATED WORK

In this section, we review related work in the fields of virtual prototyping and CAD data synthesis, exploring an increasing need for interdisciplinary data integration, standards and interfaces due to the increasing complexity and cross-linking of mechatronic systems in the automotive industry.

In the crafting industries, a typical and established application of virtual prototyping is a digital mock-up (DMU), which describes the utilization of CAD models for geometric analyses. Solutions like such proposed by Song et al. [15] show how challenges of heterogeneous, collaborative CAD assembly can be handled by DMU approaches. In addition, approaches like [8] and [13] assist in streamlining interfaces between CAD data and virtual reality applications to benefit in design review processes.

A *functional (digital) mock-up* (FMU/FDMU) enhances traditional DMU by integrating numerical simulation models, like such created with MATLAB/SIMULINK, with CAD data to enable functional simulation of product properties [5],[9]. For instance, an FMU framework has been proposed by Schneider et al. [14] which helps to shorten development times of multi-domain systems and which allows integration tests at early stages of development.

In addition, Farkas et al. [6] proposed an interdisciplinary approach to functional prototyping. The authors developed a framework that couples different simulators to address the necessity of timely simulation, review and debugging of multiple mechatronic components in complex automotive systems. The results indicate that the authors' work allows early functional design reviews and assists in the specification and evaluation of mechatronic systems.

Blochwitz et al. [3] reported a lack in independent standards for model exchange and co-simulation and therefore introduced the Functional Mockup Interface (FMI). This tool-independent standard focuses on the exchange of dynamic models and on streamlining co-simulation to improve collaboration between suppliers and OEMs. Having been started under the MODELISAR project, the FMI has recently arrived at the 2.0 standard (beta) [2] and is continued under the Modelica Association Project.

Differences, advantages and disadvantages of both concepts, FMU and FMI, have been explored by Enge-Rosenblatt et al. [4]. The authors highlight the FMU focus on interactive 3D visualization including functional simulation and the FMI focus on efficient co-simulation and model exchange. By the proposal of three options, the authors argue that these concepts can be complementary combined for comprehensive investigation of multi physical systems with promising results.

While virtual prototyping is beneficial in multiple areas, it concurrently generates many challenges due the diversity of heterogeneous systems, domains and collaborators involved in automotive design processes. Recent proceedings in related work, however, clearly indicate benefits for product life cycle management due to interdisciplinary approaches and novel data integration. The approach we propose in this paper contributes in this field by focusing the synthesis of automotive function architectures with CAD/DMU data.

3. INTEGRATING FUNCTION ARCHITECTURES WITH CAD DATA

In this section, we propose our approach of integrating automotive function architectures with CAD models. Our main contributions begin with an analysis of relevant data structures and a derivation of requirements for a consistent data mapping. Moreover, we develop an XML-based meta-format for the system-independent description of function-oriented data. In addition, we carry out a prototypical implementation by which we derive new methods for the

processing of such data. Finally, we evaluate these methods by means of actual use cases.

3.1. Definition of Relevant Data Structures

3.1.1. CAD data

In our work, the term *CAD data* is understood as a set of data that includes a geometric representation of a virtual product prototype [16] as well as some product properties. The data is originally created by CAD engineers in a source CAD system like CATIA and afterwards stored in a data management system (DMS) for further applications like a digital mock-up. We use the Siemens Teamcenter data management system because it is an established and leading system in the field of automotive product lifecycle management. Moreover, the system enables an export of CAD data with two complementary file formats: JT (Jupiter) and Siemens PLM XML. Hereby, the geometric data is kept in the JT file while structural information and product attributes of the CAD data are separately stored in the PLM XML file. The main advantage of the PLM XML format for our integration approach is that it enables an integration of custom metadata into a set of CAD data while maintaining compatibility with established system standards and processes.

In the Teamcenter data management system, the CAD data is stored in predefined sets whose content can be related to vehicle projects, assembly zones, car configurations and other extents. Such volumes can include car segments like *front end* or *cockpit*, complete car configurations (100% models) and configurations which include multiple variants and derivatives (150%+ models). A 100% model represents a specific car configuration as it can be assembled in the real world.

We use the term of *granularity* to describe the degree of segmentation of the geometric parts within a set of CAD data. A low granularity means that parts of the CAD data are merged as shown in Fig. 2 (left). Especially wires might be merged to simplify DMU applications which focus on geometric analyses and therefore do not require a high granularity.

3.1.2. Function data

Many vehicle functions are implemented as mechatronic systems which can be considered as sets of its related *elements* including *components* and *connections*. Components can be of the type *sensor*, *actuator* or *controller*. Connections between components may include but are not limited to *signal wires*, *CAN busses*, *ground cables* and *supply lines*. We use the system PREEvision for the development and visual modeling of such function architectures which we shortly call *function data* in this paper. Function architecture descriptions are used by various vehicle manufacturers in many areas of automotive product lifecycle management including development, quality assurance, production, research, customer service and the legal system. Fig. 3 illustrates an example of the function architecture of a vehicle *Headlamp Flasher*.

The actual system implementation of a vehicle function can be different, depending on the particular car configuration. For instance, the *Seat Heating* function could be implemented in the *Body Control Module* or within another dedicated controller, depending on the configuration or vehicle project. Moreover, function architectures may include different variants as illustrated in Fig. 3 where the *Headlamp Flasher* function either uses standard headlights or *Bi-Xenon* lights.

3.2. Requirements for a Consistent Data Mapping

The data integration that is approached in our work aims for a connection between two types of data: *function data* and *CAD data*. This connection requires an assignment of *elements* of the function data to corresponding *geometric parts* of the CAD data which is called *mapping*. In that regard, *consistent* mapping is

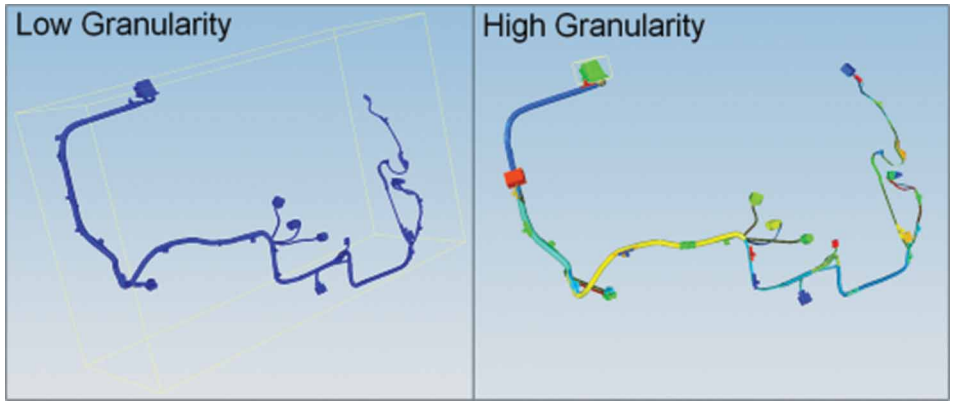

Fig. 2: Different granularity of CAD data based on the example of wiring harness.

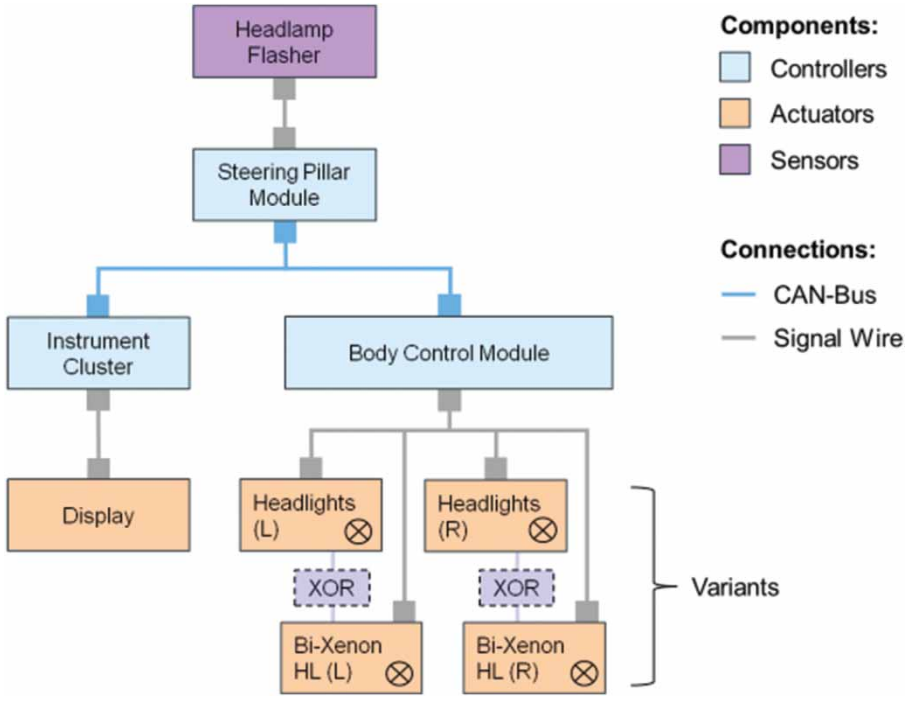

Fig. 3: Example of a proposed architecture diagram of a vehicle *Headlamp Flasher.*

understood as a complete and explicit data assignment. We briefly identify the following proposed requirements for a consistent data mapping:

- ***Identifiers.*** Identifiers are necessary to enable an explicit description and allocation of all data objects that are to be mapped.
- ***Completeness.*** The data volume of the CAD data needs to include all geometric parts which are required to completely assign all configuration-relevant function elements.
- ***Granularity.*** A sufficient level of CAD data granularity is required for a proper data assignment, especially for the mapping of connections/wires.
- ***Relations.***
  - Each function component is mapped to exactly one geometric part for a particular car configuration.
  - Each function connection is mapped to at least one geometric part if the connection is not a controller-internal connection.

In the following, we explore and discuss the above requirements and their derivation. A primary question of the data mapping is how to make function elements *know* to which geometric parts they are related to. In terms of data structures, the answer to this question is the use of clearly defined **identifiers** which are implemented as explicit key fields. In addition, the implementation and maintenance of such key fields needs to be considered in related process management. In that regard, it is reasonable to consider the use of identifiers that are already existing and utilized in established system standards and/or processes. For instance, in the development environment of our work, function identifiers are defined in a system containing a function catalog so we use these identifiers to avoid redundancy and additional complexity along the data integration.

An obvious yet important requirement for a consistent mapping is a sufficient integrity and level of **completeness** of the CAD data so that all configuration-relevant function elements can be successfully associated with geometric parts. We need to consider that particular functions may not be available in all car configurations and that the implementation of functions can be different along such configurations. If the function data contains variants as in our example in Fig. 3, a complete mapping theoretically requires a set of CAD data that includes these variants as well. However, it also is a common use case to map a single variant only with the CAD data. Therefore, we consider a mapping as complete if at least one function variant is completely mapped.

Another requirement for a consistent mapping is the **granularity** of the CAD data. In practice, such granularity is not necessarily available in all CAD data volumes because geometric parts might be merged as illustrated in section 2.1.1. A proper mapping of function data, however, requires a level of granularity that is sufficient to assign all function elements to related geometric parts. This requirement is particularly crucial for the mapping of function

connections which are usually mapped to multiple separate wire-segments.

Finally, a logical requirement of the mapping is the definition of **relations** between function elements and geometric parts of the CAD data. A single function component is always mapped to a single geometric part. However, it is also possible that multiple components refer to the same geometric part. For instance, an instrument cluster physically is a single vehicle assembly part and likewise it is represented in the CAD data as a single geometric part. However, in a functional view, it might be separately represented as two components, a controller as well as an actuator (see Fig. 3).

A single function *connection* is mapped to one or more geometric parts representing particular continuous wire segments. However, a connection might also be internal, which means that it is implemented within a controller. In this case, there is no geometric representation at all and thus no need for a data mapping.

3.3. An XML-based Format for the Interdisciplinary Utilization of Function Architectures

In this subsection, we describe the development of a new XML-based format for the description of automotive function architectures. Our format enables an integration of such data into PLM XML structures and, in general, it enables an interdisciplinary exchange and utilization of function data across different systems and domains. The format exploits many advantages of XML like language- and platform independence, human-readability and validation using schemas.

Based upon the mapping requirements and the properties of the function data, prerequisites for our XML schema can be identified as follows:

- Identifiers are necessary to enable an explicit data allocation.
- A function is a set of its elements, including components and connections.
- Components can be of different types (actuator, sensor or controller).
- Connections can be of different types (signal-wire, can-bus, etc.).
- Functions comprise 1…* components and 0…* connections.
- Mapping references should be optionally specifiable in XML files.
- Support for configurations/variants is required.
- Additional properties should be optionally specifiable for function elements.

To begin with, the schema is supposed to support the descriptions of single functions as well as multiple functions in a single XML file to provide a high grade of flexibility. Therefore, a data element called *Function* is defined to include the function data and in addition, a parental object called *FunctionCatalog* is defined which can include any number of Function elements. Both elements are declared as global elements so that they both can be used as root elements in valid XML documents.

The data mapping requires functions and all of its elements to be referable by explicit identifiers. In addition, to increase the human-readability of related XML-documents, such elements should be given a descriptive name. Finally, functions and its elements should be optionally relatable to particular car configurations. We have implemented these three requirements as suggested data fields in a global attribute group called *Header* so this data structure can be reutilized by all data elements of the schema.

The schema uses a complex type called *ComponentType* that defines the data type for the description of function components. This data type uses a simple type *MechatronicType* to define the type of the component. Enumerations are used to restrict values to either *actuator*, *sensor* or *controller*. An optional element *Annotations* is added for additional descriptive information. Finally, we define an element called *MappingLink* which is used to refer to a geometric part to which the component is mapped to. This element is defined as *optional* because we also want to allow XML documents as valid which do not include mapping references.

A complex type *ConnectionType* is defined for the description of function connections which also uses the *Header*. Annotations are omitted in favor of simplicity because they were not needed in our use cases. *ConnectionType* includes a simple type *WireTyp* to describe the type of connection and uses enumerations to restrict its values to either *can-bus*, *signal, ground, supply* or *other*. In addition, we define an element called *Connector.* This element is needed to create the logical connection to the components to which the connection is attached to using a single identifier. We set up the minimal occurrences of connectors to 2 because a connection needs to be consisting of at least two connectors. Analogously to components, we add an optional *MappingLink* with unbounded maximum occurrences to allow a connection object to be assigned to multiple wire segments.

The complex data types *ComponentType* and *ConnectionType* are implemented in the XML schema as sub elements *Component* and *Connection* of a complex type *FunctionType* that defines the primary data structure for functions. The *Connection* element is considered optional (0…*) because a function theoretically can be consisting of one controller only without any further connections. In addition, the *FunctionType* element also uses the *Header* to provide a unique identifier, name and configuration key.

Another requirement for the schema is the optional extensibility of function elements with

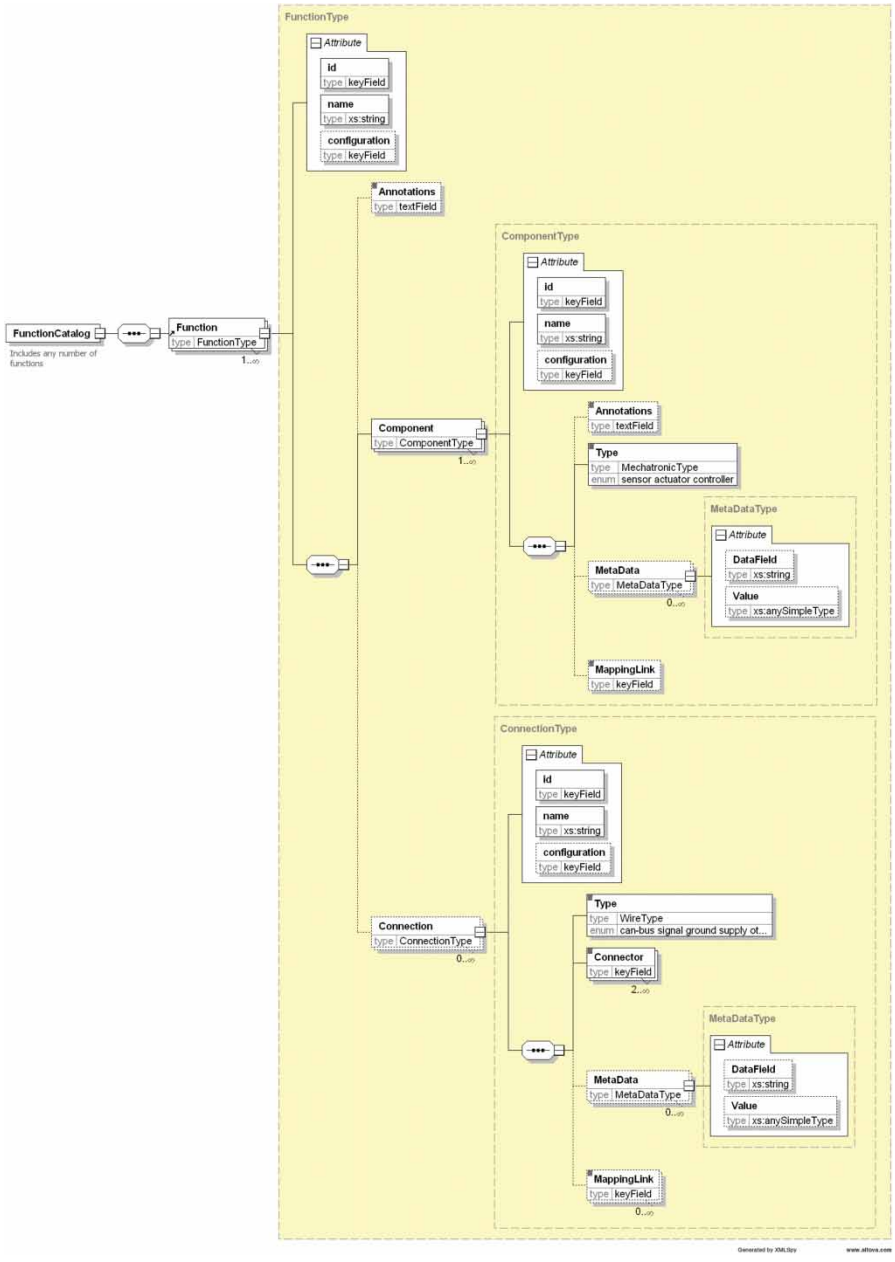

Fig. 4: Our proposed XML-schema enables an interdisciplinary and system-independent utilization of function architectures across different domains of automotive development.

additional metadata. For example, such metadata could include additional information like the software version or test maturity of controllers. Since the type of required metadata is significantly depending on particular functions and applications, the schema uses a generic data structure to provide a sufficient grade of flexibility. Therefore, a complex type *MetaDataType* is defined which uses the attributes *DataField* and *Value* to describe the name and the value of the metadata element. A function element can utilize any number of *MetaData* elements to enable the inclusion of additional information.

Finally, a namespace can be assigned to the schema to avoid interferences with other XML documents. In this paper, however, we have left out the namespace in favor of a better readability of the illustrated documents and diagrams.

The proposed schema implementation focuses on simplicity and on the avoidance of redundancy so that it can be easily understood by end users. If necessary, it might become a subject of future work to extend our implementation by additional elements to adopt new and/or changing requirements. Fig. 4 shows the complete structure of the final XML schema using

a visual diagram created with the Altova XMLSpy® schema editor. The actual XML code is to be found in the **APPENDIX**.

The following code shows a schema-valid XML document that describes the function of a hypothetical *Headlamp Flasher* which has been proposed

```
<?xml version="1.0"encoding="UTF-8"?>
<FunctionCatalog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FunctionSchemaFinal.xsd">
  <Function id="f0134"name="Headlamp Flasher">
    <Component id="s023"name="Headlamp Flasher (Button)">
      <Type>sensor</Type>
    </Component>
    <Component id="ctr012"name="Steering Pillar Module">
      <Type>controller</Type>
    </Component>
    <Component id="ctr003"name="Instrument Cluster">
      <Type>controller</Type>
    </Component>
    <Component id="ctr002"name="Body Control Module">
      <Type>controller</Type>
      <MetaData DataField="Test Maturity"Value="40%"/> <!-- Metadata examples ->
      <MetaData DataField="Version"Value="1.00"/>
    </Component>
    <Component id="ac005"name="Display">
      <Type>actuator</Type>
      <MetaData DataField="Size"Value="8"/>
    </Component>
    <Component id="ac006L"name="Headlight (L)"configuration="00001">
      <Type>actuator</Type>
    </Component>
    <Component id="ac006R"name="Headlight (R)"configuration="00001">
      <Type>actuator</Type>
    </Component>
    <Component id="ac007L"name="Bi-Xenon Light (L)"configuration="00002"> <!-- Headlight
variants ->
      <Type>actuator</Type>
    </Component>
    <Component id="ac007R"name="Bi-Xenon Light (R)"configuration="00002">
      <Type>actuator</Type>
    </Component>
    <Connection id="cn01"name="Connection1">
      <Type>other</Type>
      <Connector>s023</Connector>
      <Connector>ctr012</Connector>
    </Connection>
    <Connection id="cn02"name="Connection2">
      <Type>other</Type>
      <Connector>ctr003</Connector>
      <Connector>ac005</Connector>
    </Connection>
    <Connection id="cn03"name="Connection3">
      <Type>can-bus</Type>
      <Connector>ctr012</Connector>
      <Connector>ctr003</Connector>
      <Connector>ctr002</Connector>
    </Connection>
    <Connection id="cn04"name="Connection4"configuration="00001">
      <Type>signal</Type>
      <Connector>ctr002</Connector>
      <Connector>ac006L</Connector>
```

```
      <Connector>ac006R</Connector>
    </Connection>
    <Connection id="cn05"name="Connection5"configuration="00002">
      <Type>signal</Type>
      <Connector>ctr002</Connector>
      <Connector>ac007L</Connector>
      <Connector>ac007R</Connector>
    </Connection>
  </Function>
</FunctionCatalog>
```

in section 2.1.2. The code also utilizes hypothetical metadata by way of example.

3.4. Prototypical Implementation / New Methods of Application

We carried out a prototypical implementation of our integration approach to derive new methods for the utilization of function data and to evaluate the benefits of these proposed methods in the field. Therefore, we integrated a few function architectures of different vehicle functions with CAD data for an exemplar vehicle. In particular, we exported the function architecture data from the system PREEvision using a prototypic interface that creates XML files that are valid to our schema. In addition, we exported the CAD data as JT and PLM XML from a data management system. With the function data available in XML, we were able to integrate this data into the PLM XML file so that we could load the merged data in a 3D visualization system. We used the system VisMockup, which is an established application for digital-mock-ups.

Our prototype enables different kinds of novel methods for the visualization and analysis of the function data. A primary application is the highlight of function architectures in the full car assembly as shown in Fig. 5. Function components are visualized in colors depending on their type and function-related wires are marked red within the blue-colored electrical system.

Our approach also allows for integrating multiple functions so that it enables a comparison of different function architectures in a specific car project and/or configuration. Fig. 6 shows an example that highlights function components of three different functions in the car front end from a top down view.

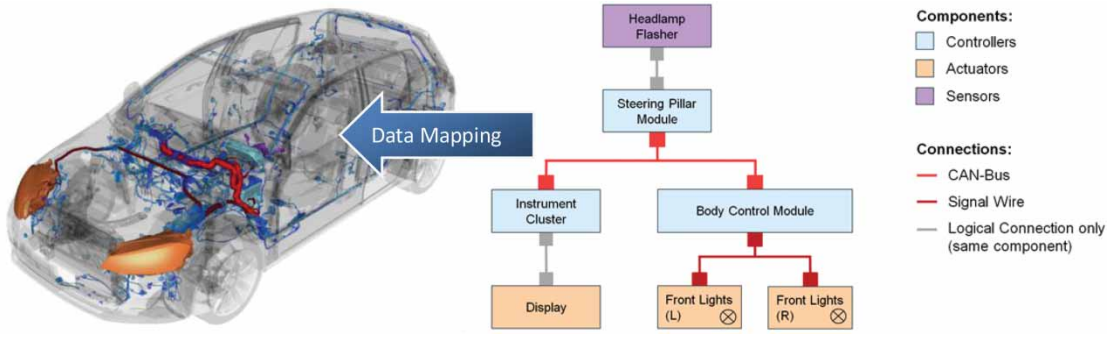

Fig. 5: Our data integration enables a spatial visualization of function architectures. This example shows the architecture of a vehicle *Headlamp Flasher*.

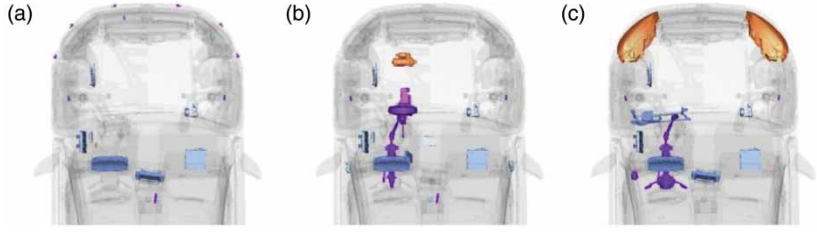

Fig. 6: Our work enables a spatial comparison of function architectures in particular vehicle projects. The example shows related components of the following functions (from left to right): (a) *Park Assist*, (b) *Start-Stop Automatic* and (c) *Dynamic Light Assist*.

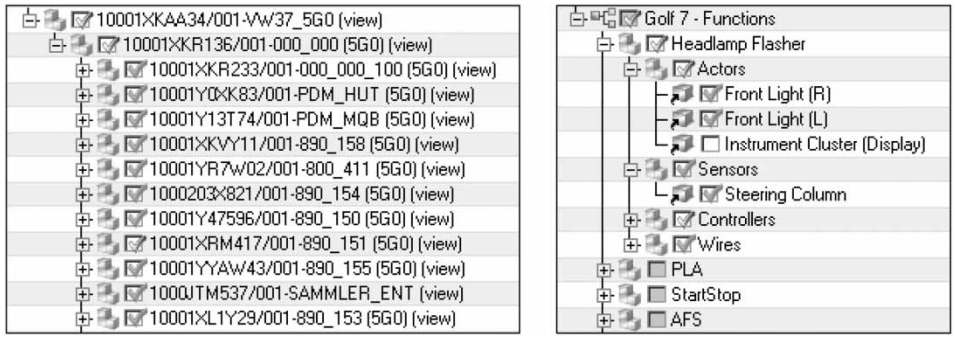

Fig. 7: Our proposal for data integration enables a function-oriented data hierarchy navigation (right) in addition to the hypothetical DMS/assembly-based hierarchy (left).

In addition, data integration enables us to create visual reports that incorporate any attribute values of the function metadata that can be based on predefined rules and conditions. For instance, all controllers of a function could have a custom attribute called *test maturity* with values ranging from 0% to 100%. Based on this example, a simple use case for a visual report could be a red marking of all controllers, whose test maturity is below 50%.

Finally, the integrated data adds the possibility of using a function-oriented hierarchy for data navigation within the CAD data structure. Usually, the CAD data hierarchy is based on assembly zones and/or structures that are derived from the data management system which can be considerably cryptic (see Fig. 7). The integration of function data provides the option to use an alternative data hierarchy that enables a function-oriented data exploration (see Fig. 7).

3.5. Use Cases and Practical Applications

Our novel approach to consistent data integration allows engineers to have a simultaneous and consistent view on function-oriented and geometric data throughout development processes and the product lifecycle. In this sub-section, we briefly explore a few use cases to demonstrate the applicability of our approach in the field.

A key requirement for industrial collaboration is a proper communication which especially applies for a function-oriented development due to its high degree of interdisciplinarity. Our approach enables a spatial visualization and communication of function architectures that increases the perceptibility and understandability of such data as visual experience is known as being the most dominant learning mode [1]. In automotive application, our work is helpful for all users of function architecture descriptions like function supervisors, project managers and many other engineers whose work is related to such information.

A manufacturer's module strategy can also significantly influence electric/electronic development and a specific challenge is the high diversity of derivatives and vehicle configurations. As noted in section 3.1.2, car configurations may also change the actual implementation of a function. As a result, individual function components can be differently located across such configurations. Our approach of spatial visualization allows for making such differences notable and comparable.

Another benefit of our approach is to be found in the development and validation of the electrical system. While two components may be adjacent and directly linked in an architecture diagram, it does neither necessarily mean that those components are implemented next to each other in the actual car nor that there is a direct connection between them. Our approach of spatial visualization enables precise statements on such wiring distribution as well as related cause-effect relationships so that potential issues can be detected and enclosed earlier in the design process.

The increasing electric/electronic complexity of modern cars also poses a challenge for the customer service. When a malfunction is detected, it is a typical task for the service to locate and repair the concerned part(s). A spatial visualization can assist technicians in the tracking of components and wires that are related to particular functions. Our approach can help to decrease such tracking times, especially under consideration that assembly positions of components can be different, depending on the car model and/or derivation, or, considering that technicians are not necessarily familiar with a particular car model. Moreover, our data integration enables reclusions about other functions that may be potentially malfunctioning because of their relationship to a defect component. Finally, our methods can be used in the education of new service technicians to illustrate function architectures and for the creation of service documents.

Finally, our work assists a functional mock-up approach because it integrates fundamental function-oriented data structures that can serve as groundwork for FMU and other advanced applications that are related to vehicle functions.

4. CONCLUSIONS AND FUTURE WORK

A holistic development process that integrates function-oriented development and geometric design is still a comparatively young concept. In our work we have shown that it is possible to implement the integration of function-oriented data with CAD data in existing development processes. Our research assists a function-oriented development by the provision of novel, visual methods for the analysis and communication of automotive function architectures and supports an interdisciplinary collaboration between different domains in automotive development. We have demonstrated the importance and benefits of our approach with a few use cases.

Our prototypical implementation has revealed some challenges that remain for further work. The utilized visualization system is not able to exploit all potentials of the data integration because it lacks optimal interfaces and data processing capabilities. Such limitations could be overcome, for example, by a custom plugin that focuses on the utilization of the function-oriented data. In addition, the implementation of our approach into daily work processes may require changes to established work processes.

In conclusion, the concept of holistic and interdisciplinary development is beneficial in many aspects but also raises challenges at interfaces as well as at the mindsets of developers and our approach shows much potential for further exploitation.

REFERENCES

[1] Barry, A.: Perception Theory, Handbook of Visual Communication, Erlbaum, New Jersey, 2005, 45–62.

[2] Blochwitz, T.; Otter, M.; Akesson, J.; Arnold M.; Clauß C.; Elmqvist, H.: Functional Mockup Interface 2.0, 9th International Modelica Conference, Munich, 2012.

[3] Blochwitz, T.; Otter, M.; Arnold, M.; Bausch, C.; Clauß, C.; Elmqvist, H.: The Functional Mockup Interface for Tool independent Exchange of Simulation Models, 8th International Modelica Conference, Dresden, 2011, 20–22.

[4] Enge-Rosenblatt, O.; Clauß, C.; Schneider, A.; Schneider P.: Functional digital mock-up and the functional mock-up interface, Proceedings of the 8th international Modelica Conference, Dresden, 2011, 748–755.

[5] Enge-Rosenblatt, O.; Schneider, P.; Clauß C.; Schneider, A.: Functional Digital Mock-up – Coupling of Advanced Visualization and Functional Simulation for Mechatronic System Design, ASIM-Treffen STS/GMMS Proceedings, Ulm, 2010, 41–48.

[6] Farkas, T.; Hinnerichs, A.; Neumann, C.: An Interdisciplinary Approach to Functional Prototyping for Mechatronic Systems using Multi-Domain Simulation with Model-Based Debugging, Proceedings of the International Multi-Conference on Engineering and Technological Innovation, Orlando, Florida, 2008.

[7] Gottschalk, B.; Kalmbach, R. (Eds.): Mastering Automotive Challenges, Kogan Page, London, 2007.

[8] Kim, S.; Weissmann, D.: Middleware-based integration of multiple CAD and PDM systems into virtual reality environment, Computer-Aided Design & Applications 3(5), 2006, 547–556.

[9] Krause, F.; Franke, H.; Gausemeier, J.: Innovationspotenziale in der Produktentwicklung, Carl-Hanser Verlag, München, 2007.

[10] Rao, N. R.: Innovation through Virtual Technologies, Virtual Technologies for Business and Industrial Applications, Business Science Reference, Hershey, New York, 2011, 1–13.

[11] Revermann, K.: Function-oriented Development, edaWorkshop 09 Proceedings, VDE, Berlin, 2009. https://www.edacentrum.de/content/function-oriented-development

[12] SAE-China; FISITA (Eds.): Proceedings of the FISITA 2012 World Automotive Congress, 6, Springer, Berlin, 2013.

[13] Schilling, A; Kim, S.; Weissmann, D.; Tang, Z.; Choi, S.: CAD-VR geometry and meta data synchronization for design review applications, Journal of Zhejiang Univ. Sci. A, 7(9), 2006, 1482–1491

[14] Schneider, P.; Clauß, C.; Schneider, A.; Stork, A.; Bruder, T.; Farkas, T.: Towards more insight with functional digital mockup, European Automotive Simulation Conference, 2009.

[15] Song, I.; Chung, S.: Synthesis of the digital mock-up system for heterogeneous CAD assembly, Computers in Industry 60(5), 2009, 285–295.

[16] Wang, G.: Definition and Review of Virtual Prototyping, Journal of Computing and Information Science in Engineering, 2(3), 2002, 232–237.

[17] Weber, J.: Automotive Development Processes, Springer, New York, 2009. DOI: 10.1007/978-3-642-01253-2

[18] Zachmann, G.: VR-Techniques for Industrial Applications, Virtual Reality for Industrial Applications, Springer, Berlin, 1998, 13-38. DOI: 10.1007/978-3-642-46847-6_2

APPENDIX – COMPLETE CODE OF OUR FUNCTION-ORIENTED XML SCHEMA

```
<?xml version="1.0"encoding="UTF-8"?>
<xs:schemaxmlns:func="VWFunctionSchema"xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"attributeFormDefault="unqualified">
<xs:element name="FunctionCatalog">
<xs:annotation>
<xs:documentation>Includes any number of functions</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:sequence>
<xs:element ref="Function"maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Function"type="FunctionType"/> 4 <xs:complexTypename="FunctionType">
<xs:sequence>
<xs:element name="Annotations"type="textField"minOccurs="0"/>
<xs:element name="Component"type="ComponentType"maxOccurs="unbounded"/>
<xs:element name="Connection"type="ConnectionType"minOccurs="0"maxOccurs="unbounded"/>
</xs:sequence>
<xs:attributeGroup ref="Header"/>
</xs:complexType>
<xs:complexType name="ComponentType">
<xs:sequence>
<xs:element name="Annotations"type="textField"minOccurs="0"/>
<xs:element name="Type"type="MechatronicType"/>
<xs:element name="MetaData"type="MetaDataType"minOccurs="0"maxOccurs="unbounded"/>
<xs:element name="MappingLink"type="keyField"minOccurs="0"/>
</xs:sequence>
<xs:attributeGroup ref="Header"/>
</xs:complexType>
<xs:complexType name="ConnectionType">
<xs:sequence>
<xs:element name="Type"type="WireType"/>
<xs:element name="Connector"type="keyField"minOccurs="2"maxOccurs="unbounded"/>
<xs:element name="MetaData"type="MetaDataType"minOccurs="0"maxOccurs="unbounded"/>
<xs:element name="MappingLink"type="keyField"minOccurs="0"maxOccurs="unbounded"/>
</xs:sequence>
<xs:attributeGroup ref="Header"/>
</xs:complexType>
<xs:attributeGroup name="Header">
<xs:attribute name="id"type="keyField"use="required"/>
<xs:attribute name="name"type="xs:string"use="required"/>
<xs:attribute name="configuration"type="keyField"use="optional"/>
</xs:attributeGroup>
<xs:simpleType name="MechatronicType">
<xs:restriction base="xs:string">
<xs:enumeration value="sensor"/>
<xs:enumeration value="actuator"/>
<xs:enumeration value="controller"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="WireType">
<xs:restriction base="xs:string">
<xs:enumeration value="can-bus"/>
<xs:enumeration value="signal"/>
<xs:enumeration value="ground"/>
<xs:enumeration value="supply"/>
<xs:enumeration value="other"/>
</xs:restriction>
```

```
</xs:simpleType>
<xs:simpleType name="keyField">
<xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="textField">
<xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:complexType name="MetaDataType">
<xs:attribute name="DataField"type="xs:string"/>
<xs:attribute name="Value"type="xs:anySimpleType"/>
</xs:complexType>
</xs:schema>
```