Datenbank Spektrum
DOI 10.1007/s13222-013-0135-9

SCHWERPUNKTBEITRAG

On the Integration of Electrical/Electronic Product Data in the Automotive Domain

Challenges, Requirements, Solutions

Julian Tiedeken · Manfred Reichert · Joachim Herbst

Received: 10 June 2013 / Accepted: 24 August 2013

Abstract The recent innovation of modern cars has mainly been driven by the development of new as well as the continuous improvement of existing electrical and electronic (E/E) components, including sensors, actuators, and electronic control units. This trend has been accompanied by an increasing complexity of E/E components and their numerous interdependencies. In addition, external impact factors (e.g., changes of regulations, product innovations) demand for more sophisticated E/E product data management (E/E-PDM). Since E/E product data is usually scattered over a large number of distributed, heterogeneous IT systems, application-spanning use cases are difficult to realize (e.g., ensuring the consistency of artifacts corresponding to different development phases, plausibility of logical connections between electronic control units). To tackle this challenge, the partial integration of E/E product data as well as corresponding schemas becomes necessary. This paper presents the properties of a typical IT system landscape related to E/E-PDM, reveals challenges emerging in this context, and elicits requirements for E/E-PDM. Based on this, insights into our framework, which targets at the partial integration of E/E product data, are given. Such an integration will foster E/E product data integration and hence contribute to an improved E/E product quality.

Keywords Product data integration · Common integration ontology

1 Introduction

A modern car consists of up to 100 *electronic control units* (ECUs) [25, 26]. Continuous product innovations and growing customer requirements are further increasing the complexity as well as the number of electrical/electronic (E/E) products in cars. Due to country-, market-, and customer-specific requirements, in addition, a large number of variants of E/E product data and corresponding engineering processes must be maintained [17, 18]. In this context, simultaneous engineering, manufacturer-supplier-relationships, and development processes show complex interdependencies (cf. Fig. 1). In particular, E/E product data refers to numerous digital development artifacts, like requirement specifications, circuit diagrams, wiring diagrams, bootloader, flashware, and software. In turn, these artifacts are created and managed by autonomous, distributed, and heterogeneous information systems serving a variety of user requirements. In this context, numerous environmental regulations (RoHS,[1] WEEE,[2] EuP[3]) as well as legal requirements (e.g., product liability, ISO 26262 [19]) demand for an integrated traceability of all E/E product development steps. Accordingly,

This is an extended version of the paper *Managing Complex Data for Electrical/Electronic Components: Challenges and Requirements* selected for the special DASP issue Best Workshop Papers of BTW 2013.

J. Tiedeken (✉) · M. Reichert
Institute of Databases and Information Systems, Ulm University, Ulm, Germany
e-mail: julian.tiedeken@uni-ulm.de

M. Reichert
e-mail: manfred.reichert@uni-ulm.de

J. Herbst
ITM Group Research & Product Development MBC, Daimler AG, Böblingen, Germany
e-mail: joachim.j.herbst@daimler.com

[1]Restriction of Hazardous Substances Directive.

[2]Waste Electrical and Electronic Equipment Directive.

[3]Energy-using Products Directive.

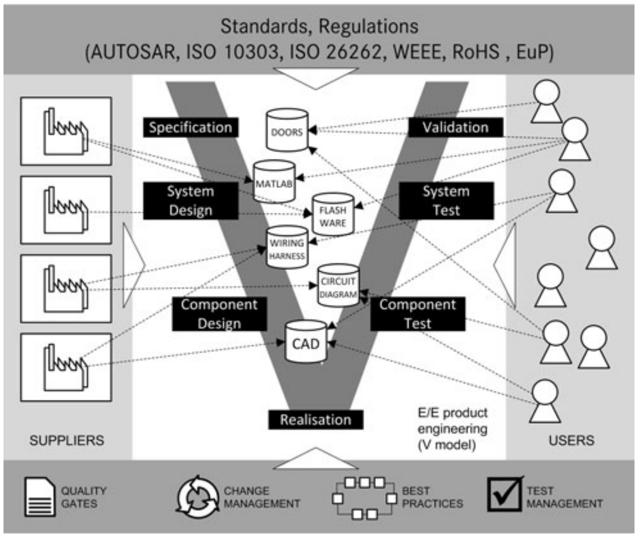

Fig. 1 Complexity of E/E product data

any change of E/E product data must be documented. For this purpose, E/E product data management (E/E-PDM) systems maintain configurations that bundle interrelated E/E products.

Each E/E product has a life cycle that comprises the steps required to create and distribute it, e.g., planning, development, production, and after sales [6, 25]. In particular, product development constitutes an integral part of the product life cycle. In this context, emerging market trends (i.e., e-mobility), increasing product liability, environmental regulations, tighter integration of suppliers, and shorter development cycles necessitate high data quality as well as integrated E/E product development processes. In particular, an IT landscape must adequately cope with product changes in the given context.

An application is based on a conceptual schema (henceforth referred to application schema), which describes its semantics based on concepts and their relationships. In current practice, a multitude of heterogeneous applications are used for accomplishing the different E/E product development tasks. In turn, this results in numerous artifacts with partially overlapping information. In particular, ensuring data consistency and data quality (e.g., accuracy and actuality) constitutes a challenging task in such an environment. Especially, use cases requiring data from different sources are difficult to handle due to missing mappings between interrelated artifacts. To realize these use cases, in turn, the partial integration of heterogeneous, distributed schemas from different IT applications become necessary. Usually, these schemas rely on different concepts for versioning, variability support, and aggregation of E/E product data. For example, certain applications have predefined release dates for new E/E product data versions, whereas other applications create new E/E product data versions for every performed change. To integrate E/E product data versions from different applications, manual mappings between these artifacts must be maintained, which is a cumbersome and error-prone task. Furthermore, semantic dependencies among the artifacts corresponding to different applications are usually not documented.

1.1 Problem Statement

The need for integrating E/E product data has been recognized as a major challenge for large companies that maintain a multitude of autonomous and heterogeneous data sources [16, 22]. The key principle of the approaches existing in this context is to create a unified view (i.e., *global schema*), which integrates data from local schemas. As an example consider Federated Database Systems (FDBSs) [33] and Data Warehouses (DWHs) [8]. While the former target at the integration of autonomous and heterogeneous data sources, the latter focus on intelligent decision support with respect to management decisions. In particular, FDBSs use view-based techniques for integrating data sources (*Local-as-View*, *Global-as-View*, and *Global-and-Local-as-View*). In turn, a DWH provides mechanisms for extracting, transforming, and loading data from different sources as well as for analysing these data. However, respective integration approaches focus on the technical level, solely. Still, there is a lack in respect to schema evolution.

Usually, an IT landscape supporting E/E product development has evolved over many years and is based on a comprehensive engineering expertise. In this context, a particular application domain involved in E/E product development does not fully disclose its application schema to other domains. In addition, tacit knowledge and schema workarounds are ubiquitous. Overall, FDBSs and DWHs are not sufficient for creating a common integration model in such an environment. Besides integrating data at the technical level, a comprehensive methodology is needed that explicitly considers standards fostering the exchange and interoperability of product data. Examples of such standards include MSR,[4] AUTOSAR[5] [3, 4], and ISO 10303 [30].

1.2 Contribution

This paper elaborates on the problems that emerge when partially integrating heterogeneous application data models, which contribute to the development of E/E product data. We identify fundamental requirements any IT support for E/E product data management must meet in such a setting and give insights into our integration framework that aims at satisfying these requirements. In particular, we discuss the

[4]Manufacturer Supplier Relationship.

[5]AUTomotive Open System ARchitecture.

limitations of existing matching approaches in the context of E/E product data and present solutions to overcome these limitations.

This paper provides a significant extension of the work we presented in [36]. First, the requirements we elaborated in [36] are presented in more detail. Second, we present fundamental aspects of our framework enabling the partial integration of E/E product data. Third, we present the results of a use case from the automotive industry during which we integrated two application schemas. Finally, we give detailed insights into our integration framework.

The remainder of this paper is organized as follows: Sect. 2 discusses the problems and challenges we identified in the context of a comprehensive system analysis. This includes a discussion of the requirements existing for an effective E/E-PDM support. Taking these requirements into account, in Sect. 3 we present our framework, which targets at the partial integration of E/E product data. Related work is discussed in Sect. 4. The paper concludes with a summary and an outlook in Sect. 5.

2 Findings, Challenges, Requirements

In the automotive domain, we analyzed a variety of applications involved in E/E development processes. In this context, we further analyzed specific use cases dealing with application-spanning consistency checks for E/E product data (e.g., checking the plausibility of logical connections between electronic control units and networks). This section summarizes the results of these analyses. It further identifies fundamental challenges emerging in the context of E/E development processes and their IT support and, finally, elicits requirements for a holistic and effective E/E-PDM. For the sake of readability, we grouped our results into four main challenges.

2.1 Challenge 1: Establishing a Common Ontology for E/E Product Data Integration

Findings A main characteristic of E/E development processes is imposed by the underlying manufacturer-supplier relationships; i.e., nowadays, systems and components (e.g., sensors, actuators, and electronic control units) are developed in close collaboration with suppliers. Thereby, systems and sub-systems describe parts of a car, which are characterized by the functions provided. Usually, systems aggregate features that can be perceived by the customer, e.g., heat, ventilation, or air conditioning. Furthermore, systems are technically based on interacting components. Due to the networked enterprises in a supply chain [31], for different development stages, deadlines need to be set to guarantee product delivery times. To cope with such a scenario, engineers work on development tasks concurrently (simultaneous engineering) using a variety of applications (cf. Fig. 2). In addition, these applications rely on different concepts for versioning, variability support, and aggregation of product data. Usually, engineers are working with numerous applications, hosted and maintained by different departments, to create and maintain E/E product data, e.g., applications for software function modeling, requirements engineering, and computer-aided design. In particular, these applications have been designed and developed independently from each other, and their schemas usually reflect the long-term experiences of engineers.

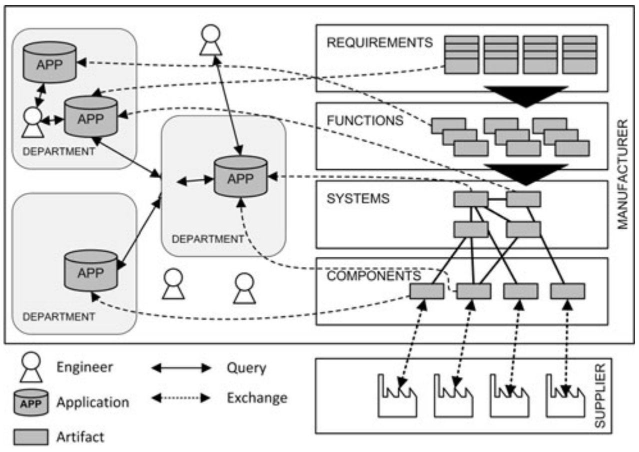

Fig. 2 Logical structure of E/E product data

In general, there exist fundamental use cases not covered by the schemas of these applications at all. For example, a car usually has ECUs to control the functions of installed seats (e.g., position, heat, and ventilation). If these ECUs do not differ in their functions, they are represented by a single object, which may cause problems when integrating this data model with other application data models. To support these use cases, a number of workarounds are applied in practice, which are based on tacit knowledge of the engineers and hence are usually not documented. Other use cases, which require data from heterogeneous data sources (e.g., checking plausibility of references between ECU pins and logical signals) are realized manually by responsible actors, i.e., the latter collect and aggregate development artifacts from different departments. In particular, this constitutes a cumbersome and time-consuming task. Moreover, it might happen that some of the relevant artifacts have to be replaced by new versions during this data collection process.

Challenges Usually, development tasks are separated into system- and component-specific parts. Both terms are ambiguously defined and used in application schemas. For example, while a component in a wiring diagram refers to a variant of an E/E component at a specific geometric position, in the context of a requirements document, the same term describes all variants of an E/E product in general. Due to the large amount of application schemas involved in E/E development processes, a full integration into one system, meeting the various user requirements, is neither possible nor feasible from a practical point of view. In fact, a partial integration of application schema concepts, which can be used to support application-spanning use cases, is needed.

Requirements For use cases that require read access to E/E data stored in different applications, a common integration ontology is needed (*Requirement 1*). Such an ontology constitutes the core of applications involved in E/E development processes. It aggregates semantically related artifacts and, therefore, provides a unified view on E/E product data. To integrate E/E product data, application-specific concepts for versioning, variability support, and aggregation of E/E product data as well as documentation methods must be considered (*Requirement 2*). Moreover, integrating the various development artifacts may be accomplished semi-automatically, which contributes to save time and to avoid errors as common in the context of manual integration. To prevent ambiguities, however, an integration ontology must allow modeling the semantics of application schema concepts (*Requirement 3*). Finally, business users should be able to query data of the common integration ontology (*Requirement 4*).

2.2 Challenge 2: Consistency of E/E Product Data

Findings E/E product data are created, maintained, and used in all stages of the automotive life cycle (i.e., development, production, and after sales). Usually, different stakeholders (e.g., E/E architects, system and component management, and engineers) participate in these life cycle phases using different applications [25]. In turn, this results in redundant artifacts and models (e.g., wiring harness, logical connection). Many of these are created in a particular life cycle phase and required in subsequent development phases. We denote such interdependent applications as tool chains. Generally, development artifacts are reused along such tool chains to reduce development times. For example, when developing a new car model, artifacts of the previous model series might serve as basis for product improvements. To coordinate the various development steps, E/E systems and E/E components are often developed using the V-Model [13]. The latter defines necessary steps and responsibilities of all involved parties considering the distributed development and testing of artifacts. To control and test required functionality of an entire system, integration tests for the used components become necessary. For this purpose, quality gates are introduced as integral parts of any development process [9]. In particular, they describe fixed development stages with predefined quality requirements to be fulfilled at specific points in time. Generally, development processes involve different stakeholders, departments, and applications. Although certain overviews of the global process are provided, these have not been realized based on workflow technology [32]. Also, note that changes of E/E products often become necessary and might impact other E/E products. For example, when changing the software interface of an ECU, logically connected ECUs must be identified and analyzed. For this purpose, global change management processes are required that coordinate and log the activities in the context of such change requests.

Challenges Although development of E/E products is accomplished simultaneously, artifacts from heterogeneous data sources must be integrated at specific points during development (e.g., design review or prototype analysis). In this context, complex transformations between heterogeneous application schemas must be defined and maintained. In particular, explicit mappings between artifacts from different application schemas become necessary. Currently, these mappings are defined manually. Hence, frequent communication between the different stakeholders and development departments is required, which is an error-prone and time-consuming task. Overall, ensuring the consistency of related artifacts from heterogeneous, distributed application schemas constitutes a costly task. As another problem, there is a lack of technical processes implementations for supporting E/E development. Consequently, the monitoring of E/E product data changes constitute a challenging and tedious task; i.e., requests by different stakeholders and departments must be handled manually in current practice.

Requirements The tight integration of manufacturer and suppliers necessitates data of high quality as well as standardized data collection processes (*Requirement 5*). For example, during the development of an ECU, functional models and communication information (software ports, frames, and signals) are concurrently created and maintained by different applications. Furthermore, ECU development processes are separated into manufacturer- and vendor-specific parts, i.e., vendors realize software for ECUs based on interfaces provided by the manufacturer. Hence, the consistency of the various artifacts must be guaranteed. Note that any error of a signal, frame, or bit might lead to a malfunctioning product. Consequently, it is crucial to be able to identify inconsistent E/E product data (*Requirement 6*).

2.3 Challenge 3: Schema Evolution

Findings In general, changes of application schemas are performed autonomously based on domain-specific schedules. Although these schedules are distributed across the departments involved in E/E development, the coordination of application schema changes remains a challenging task.

Challenges When evolving an application schema, other application schemas might be affected. To identify such dependent schemas is a difficult task, because the dependencies that exist between the artifacts from different schemas are not explicitly documented. In turn, if a change is implemented without notifying affected applications in advance, existing tool chains might become corrupted. In summary, before changing an application schema, dependent schemas of other applications must be determined and analyzed.

Requirements To handle application schema changes in a more systematic manner, bidirectional mappings between related artifacts from different schemas should be explicitly modeled and maintained (*Requirement 7*). This includes artifacts at the schema as well as the instance level. As an advantage of such a mapping, semantic inconsistencies between application schemas can be eliminated. Furthermore, application schemas usually comprise a multitude of artifacts, whose manual comparison across different schemas would be too costly. To reduce this complexity, matching algorithms based on well-defined criteria (e.g., name, data type) are needed. Additionally, application schema interdependencies must be analyzed to assess the impact of schema changes prior to their introduction (*Requirement 8*). In order to enable a flexible change management, therefore, transformations between application schemas should be derivable from the interdependencies maintained (*Requirement 9*). Based on this, concomitant adaptations are necessary in the context of schema change, which allows reducing development costs and error rates.

2.4 Challenge 4: Providing a Methodology for E/E Product Data Documentation

Findings In the automotive domain, AUTOSAR constitute an industry standard for developing embedded systems. In this context, the definition of standardized protocols for software development allows exchanging software running on ECUs from different vendors. In addition to the software architecture of ECUs, methodologies for describing and defining reusable and scalable software components are an integral part of AUTOSAR. Besides standardizing ECU software, processes for developing safety-related E/E products (e.g., airbag, electronic stability control) constitute an important component of contemporary E/E-PDM systems. First, *Failure Mode and Effects Analysis* (FMEA) [35] is used for failure prevention and security management. In particular, FMEA focuses on reducing the costs of changing artifacts, i.e., late changes and hence implementation costs shall be reduced. Second, as emerging standard for functional safety in the automotive domain, ISO 26262 needs to be considered. It defines a security life cycle covering all aspects of the development process. To classify different levels of security requirements, so-called *Automotive Safety Integrity Levels* (ASILs) must be assigned to safety-relevant E/E products.

Challenges Even though the methodologies inherent to AUTOSAR consider both software and hardware, other aspects like wiring harness and connectors of E/E product data are not taken into account. Additionally, tacit knowledge makes the integration of application schemas a difficult task to accomplish, because semantic inconsistencies of E/E product data might remain unresolved. Finally, an explicit monitoring as well as control of existing methodologies and guidelines for documentation of E/E products are missing. As a consequence, ambiguities in E/E product data can be frequently observed.

Requirements A comprehensive methodology for documenting E/E product data is needed, taking existing standards and methodologies (e.g., AUTOSAR, ISO 26262, Automotive SPICE [24]) into account (*Requirement 10*). To create a common integration ontology, allowing for the integration of existing application schemas, a comprehensive modeling methodology is needed. Note that applications that create and maintain E/E product data have been based on long-term user experiences. Hence, a modeling methodology must support the creation of a common integration ontology from scratch (top-down) as well as from existing schemas (bottom-up).

Table 1 summarizes the discussed findings, challenges, and requirements.

3 Integration Framework

In the following, fundamental concepts of our integration framework for E/E product data are discussed. First, we present a use case from the automotive domain for integrating two characteristic application schemas. Following this, we discuss the underlying data structure required for the integration of E/E product data. Finally, we elaborate the differences between local and common integration ontology and give insights into our matching approach.

3.1 Use Case

Figure 3 exemplarily depicts two applications from our system analysis. In particular, these applications maintain artifacts related to the same concepts and store, amongst other things, networks and related ECUs (indicated by two correspondences). However, the two applications are intended

Table 1 Summary of findings, challenges, and requirements for E/E product data management

Findings	Challenges	Requirements
– manufacturer-supplier relationships – system and component orientation – simultaneous engineering – different concepts for versioning, variability support, and aggregation of E/E product data – numerous applications hosted and maintained by different departments	– ambiguous use of terms *system* and *component* – large application schemas – partial integration of application schema concepts	– common integration ontology – consideration of existing methodologies for E/E product data documentation – explicit modeling of artifact semantics – query support for business users
– different stakeholders participating in E/E product life cycle – artifact reuse, redundancies – predefined quality gates, V-Model – frequent changes of E/E product data changes	– continuous integration of E/E product data – maintenance of complex application schema transformations	– standardized processes ensuring data collection and provisioning – data consistency management
– autonomous changes of application schemas – domain-specific schedules	– evolution of an application schema taking its interdependencies with other schemas into account	– explicit documentation and maintenance of artifact interdependencies – a-priori-analysis of application schema changes – generation of schema transformations
– numerous standards for product data exchange – domain-specific methods for documenting E/E product data	– tacit knowledge – monitoring and control of methodologies for E/E product data documentation	– comprehensive documentation methodology (support top-down and bottom-up integration scenarios)

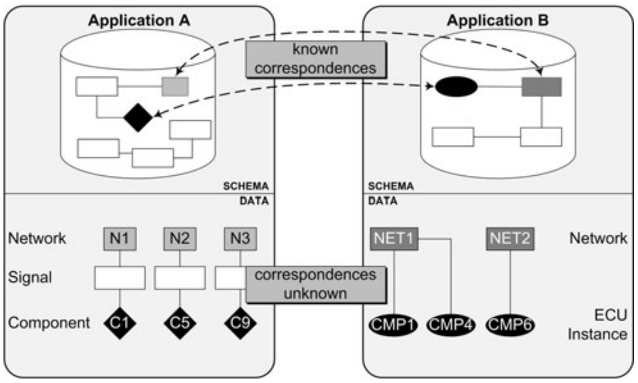

Fig. 3 Excerpt of two application data models

for and used by different stakeholders. The first application (denoted as Application A) deals with the documentation and release management of physical E/E components (e.g., ECU, sensor, actuator, battery). In turn, the second application (denoted as Application B) focuses on logical connections between ECUs (e.g., signals, frames).

As aforementioned, data from different applications must be integrated at different points in time. To check whether the data from the two applications are consistent, equivalent artifacts must be identified and their corresponding attributes be analyzed. We denote this as *matching problem*. Existing approaches distinguish different types of matching: schema, instance, and ontology matching. The first type focuses on finding semantically related artifacts in the schema or meta model layer, while the second one aims at matching artifacts at the instance level. Since ontologies comprise schema as well as instance data, both matching types are relevant here.

Note that there exists a multitude of approaches dealing with the matching problem [11, 12, 20]. The most prevalent techniques applied in this context are based on labels as well as structural information. In addition, there exist matching frameworks (e.g., AgreementMaker [10], COMA++ [2]) that provide different algorithms for matching schemas, instances, and ontologies.

We applied these existing frameworks to match application schema and instance data for the mentioned use case, but did not obtain satisfactory results. In detail, we analyzed the similarity between database tables of both applications (Application A consisted of 138 database tables, whereas application B comprised 20 tables).

To reduce the complexity of instance matching, we focused on data of two database tables related to a particular car model series (table `Component` with 160 entries and table `Network` with 44 entries of Application A; table `ECU Instance` with 187 and table `Network` with 28 entries of Application B). In particular, we used the labels of the artifacts to identify correspondences. While the matching of artifacts between entries of table `Network` from Application A and entries of table `Network` from Application B worked very well, only few artifact matchings between entries from table `Component` of Application A and table `ECU Instance` of Application B could be found.

As a result from this use case we learned that a systematic approach is required to integrate E/E product data from different applications.

3.2 Fundamentals of our Integration Framework

Figure 4 illustrates our integration framework for E/E product data, which meets the presented requirements presented

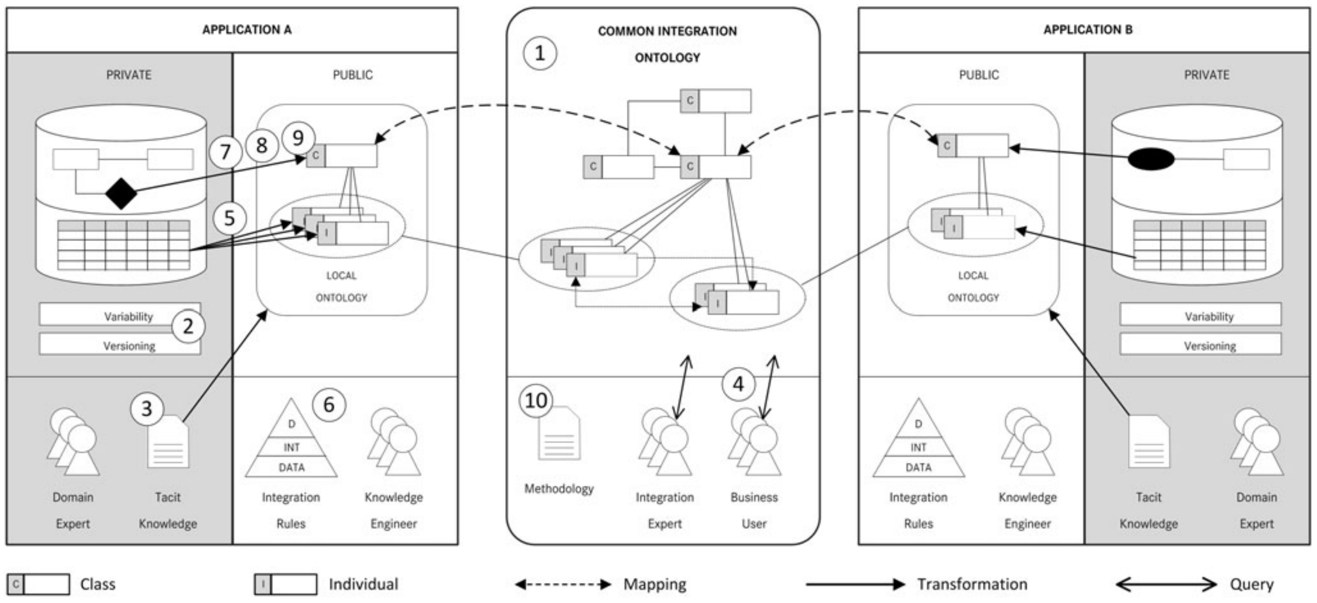

Fig. 4 Integration framework

in Sect. 2. Our main objective is the partial integration of application schema concepts from different applications based on a central ontology (denoted as *common integration ontology*) to support application-spanning use cases. Examples of the latter include consistency checks and comparisons of E/E product data. In turn, *local ontologies* encompass tacit knowledge as well as application schema concepts that are made publicly available by the application owner. In particular, corresponding concepts may be integrated into the common integration ontology. In the following, we detail the main aspects of this framework.

3.3 Data Integration Layers

An elaborated analysis of application schemas for E/E product development revealed the following common structure for the resulting product data: application schemas consist of different concepts of which each comprises a set of objects. In certain cases, these may be different variants of an object, which differ in respect to the features provided. Furthermore, each object may consist of different versions.

Figure 5 depicts the basic structure of E/E product data (ⓐ), as well as two corresponding examples (ⓑ and ⓒ). While the schema concept *Component* in ⓑ has one object (*Engine Control Module*) with two variants (*Gasoline*, *Diesel*) and corresponding versions, in ⓒ, objects of the schema concept *Network* only distinguish between different versions. Furthermore, our analysis revealed a few dozen artifacts at the schema concept level, while there are thousands of artifacts at the object level and even more at the variant and version levels.

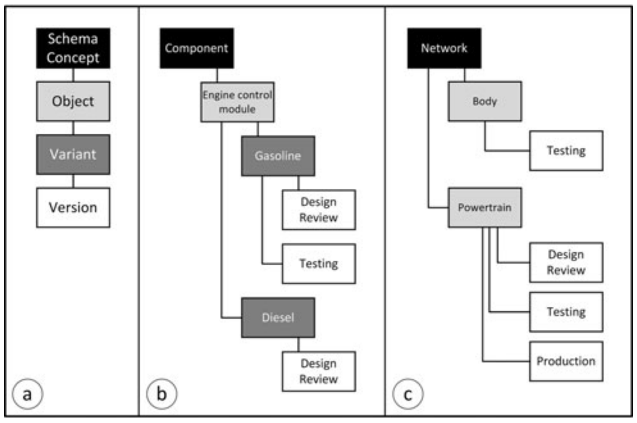

Fig. 5 Different layers for integrating data

3.4 Local Ontology

In general, an application schema has resulted from long term experience and hence provides a high business value. Therefore, application owners do not want to fully disclose their application schemas to others. Despite this fact, partial data integration of different applications can be realized with an additional layer on top of an application schema. This model, which we denote as *local ontology*, encompasses schema concepts that are made publicly available by an application owner (Requirements 2 and 3). The local ontology allows for a standardized interface for data access and guarantees independency from the private schemas. Usually, tacit knowledge related to application schemas is limited to few domain experts. In turn, this knowledge is important for integrating application schema concepts with other applications and should therefore be explicitly documented in the local ontology. As mentioned, essential E/E product data concepts (e.g., requirement, function, system, and component) share a common structure. Each concept comprises at least a set of objects with respective versions. While certain objects have different object variants, others only comprise a number of object versions. As a result, the artifacts of a local ontology are structured in the same way. Usually, local ontologies are created and maintained by knowledge engineers in cooperation with domain experts. This is accomplished through interviews and questionnaires. Domain experts and knowledge engineers are responsible for establishing data provisioning processes, which handle the transformation of data from a private schema into the local ontology (Requirement 5). As discussed in the context of our use case, database tables may comprise dozens of attributes, of which not all might be relevant for integration or queries of business users. As a result, domain and integration experts need to decide which artifact properties are relevant and hence must be transferred into the local ontology. This additional layer offers one key advantage: if a private application schema changes, the structure of the local ontology will remain stable. Consequently, schema changes are decoupled from local ontologies and, therefore, do not affect the common integration ontology.

3.5 Common Integration Ontology

A common integration ontology constitutes the smallest common knowledge base of essential E/E product schema concepts (Requirement 1). It is created by knowledge engineers in cooperation with domain and integration experts. More precisely, semantically related artifacts of different local ontologies are aggregated into new artifacts belonging to the common integration ontology. Furthermore, their interdependencies are determined (Requirement 7). Artifacts of this common integration ontology provide the basis for querying E/E product data, as required in context of consistency checks or summaries (Requirement 4).

Figure 6 illustrates a common integration ontology referring to the two local ontologies identified in our use case (cf. Sect. 3.1). For the sake of readability, we restricted ourselves to schema concepts *Component* and *ECU Instance*. Local ontology A consists of schema concept *Component*, which comprises an object *Engine Control Module*, one variant *Gasoline* and two versions *ECM Gas V1* and *ECM Gas V2*. In turn, local ontology B comprises schema concept *ECU Instance* with object *ECM* and the three object versions *ECM V1*, *ECM V2*, and *ECM V3*. To integrate the local ontologies into a common integration ontology, matchings between the artifacts are required. In particular, correspondences at the schema concept, object, variant, and version layer are required.

Based on the existing artifacts of the common integration ontology, different integration scenarios are possible: If

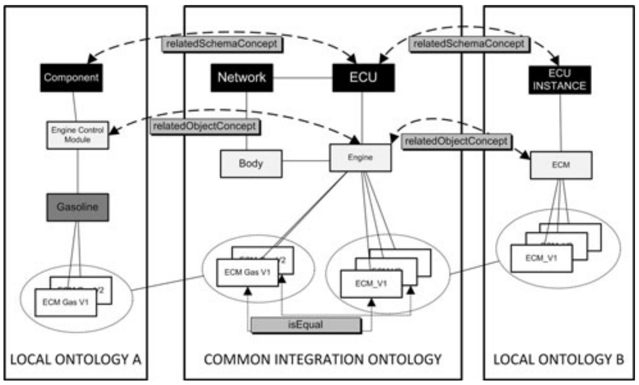

Fig. 6 Common integration ontology

there are no correspondences between the schema concepts of a local ontology and the ones of the common integration ontology, the schema concepts as well as their objects, variants, and versions will be copied into the common integration ontology. If necessary, artifacts will be labeled to match naming guidelines.

In turn, if there exist corresponding artifacts in the common integration ontology, the schema concept *ECU Instance* and its subsequent artifacts (*ECM*, *ECM_V1*, *ECM_V2*, *ECM_V3*) must be integrated with them. For example, if local ontology A is already integrated with the common integration ontology and string-based matching algorithms reveal that schema concept *ECU* from the common integration ontology and *ECU Instance* of local ontology B are related, objects, variants, and versions of *ECU Instance* will have to be integrated with the corresponding artifacts of the common integration ontology. In particular, semantically related objects, variants, and versions must be matched and particular properties (e.g., `relatedSchemaConcept`, `relatedObjectConcept`) be created (cf. Fig. 6). Note that details of the matching between the artifacts of the different layers (schema concept, object, variant, and version) are out of the scope of this paper.

Due to the small number of schema concepts, the matching of semantically related schema concepts will be realized manually by integration experts. Regarding the other artifact layers, however, thousands of objects and even larger numbers of variants and versions may have to be integrated, a task that can not be performed manually. Consequently, the matching process between objects, variants, and versions must be automated.

So far, we have focused on the integration of single schema concepts from local ontologies into the common integration ontology. Generally, local ontologies consist of many interrelated schema concepts. As a consequence, these connections between the artifacts must be added to the common integration ontology as well.

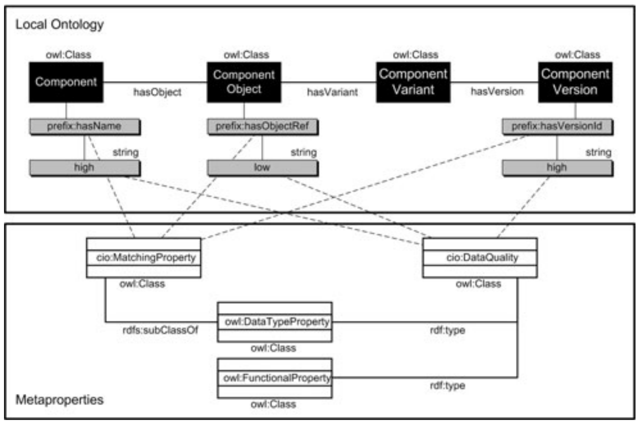

Fig. 7 Local ontology with matching properties

3.6 Matching Basics

This section presents basics of our matching approach. Our system analysis revealed that many application schemas contain cross references to artifacts of other applications. Usually, these references are string-based identifiers, which can be exploited to find correspondences between artifacts from different applications. In most cases, respective references are known to domain and integration experts. Therefore, they should be documented in the corresponding local ontologies. Note that the data quality of these cross references varies significantly. Figure 7 illustrates a local ontology including properties, which are relevant for the integration of E/E product data into the common integration ontology. Technically, local and common integration ontologies are modeled in OWL2 [23], which is an ontology language providing classes, properties, individuals, and data values. In particular, the latter are stored as Semantic Web documents. The different layer for integrating E/E product data are represented as `owl:Class`, e.g., schema concept layer `Component`, object layer `Component-Object`, variant layer `ComponentVariant`, and version layer `ComponentVersion`. Artifacts of the different layers represented as individuals of the respective classes. To annotate an artifact property for matching purpose, it is associated to meta property `cio:MatchingProperty`, which is a subclass of `owl:DataTypeProperty`. The data quality of the property values can be documented with the datatype property `cio:DataQuality`, which is a functional datatype property with predefined string values (‘`high`’, ‘`low`’). In order to automate the integration of a local ontology with the common integration ontology, integration experts must select matching properties from both ontologies and define a mapping function. Based on the data quality of matching properties, the correspondences resulting from the automated matching must be evaluated and adapted, if required.

4 Related Work

Federated database systems (FDBSs) [33] address a similar problem as presented in this paper. Their main goal is to integrate autonomous, heterogeneous data sources. To the best of our knowledge, however, there is no work on the documentation of application schema semantics and tacit knowledge. Furthermore, schema changes in FDBS have not been addressed in sufficient detail so far. Both aspects are essential for integrating application schemas in the context of E/E-PDM.

There exist a multitude of approaches focusing on the matching problem [11, 12, 20]. We applied some of the existing matching frameworks [2, 10] to the E/E product data from our use case, but did not obtain satisfactory matching results. Due to the large number of artifacts existing at the different layers of E/E product data (schema concept, object, variant, version), these generic matching frameworks are not sufficient for integrating E/E product data.

A common meta model for the integration of different tools in context of embedded system engineering is presented in [5]. This approach derives concepts for a common meta model from the HRC [34] and EAST-ADL2 [1] meta models. Although the authors consider concepts like component, part, and port, other relevant ones (e.g., requirement, system) are missing.

ToolNet [14] focuses on tool integration and data consistency. For this purpose, consistency relations between reference objects are modeled manually. The approach focuses on requirements specifications, function models and geometric product data. As a major drawback, changes of application schemas are not considered.

In [7], a model-based software (re-)engineering approach is presented. It integrates tools at the meta model level and proposes two design patterns for data integration. Furthermore, consistency rules and integration constraints are provided. For this purpose, simple graph grammar rules are defined restricting a meta model in order to enable interoperability. Note that this enables consistency checking as well. Finally, a triple graph grammar is used to model the semantic relationships between the different meta models. Other approaches using a triple graph grammar are presented in [21].

The evolution of database schemas is compared with the one of ontologies in [27]. In particular, it highlights the differences of both research areas and discusses challenges for ontology evolution. Different approaches for detecting changes between OWL ontologies are presented in [15]. Altogether, the results presented in [15, 27] constitute a starting point for our future work on the evolution of the required common integration ontology.

Problems concerning interoperability for software system are discussed in [28]. Based on identified mismatches in message exchange protocols, mediation patterns are introduced. In [29], different levels of software system interoperability are distinguished (syntactic, semantic, and pragmatic interoperability). Furthermore, requirements for assessing respective interoperabilities are elaborated and possible solutions are discussed. In particular, ontologies are used to represent individuals, classes, properties, result constraints, and causality constraints.

Overall we conclude that there exists no holistic approach covering the aforementioned challenges for E/E product data management in an integrated and comprehensive manner. Although many approaches focus on tool integration, change management of partly integrated application schemas has not been adequately considered so far. Besides this aspect, methodologies for E/E product data management are missing.

5 Summary and Outlook

This paper has identified fundamental challenges emerging in the context of E/E-PDM. Based on a comprehensive analysis of applications involved in E/E development processes as well as a characteristic use case for application-spanning consistency checks, we have elicited requirements for effective E/E-PDM. In order to enable application-spanning uses cases, in turn, a common integration ontology is required allowing for the integration of relevant concepts from existing application schemas. In this context, ensuring consistency among the distributed artifacts is a challenging task. Another challenge concerns the evolution of application schemas. Note that respective changes of application schemas are common and hence should be supported. As a prerequisite, schema interdependencies must be explicitly documented. Note that the latter is crucial for automatically deriving schema transformations. Finally, a methodology for the documentation of E/E product data is needed, which takes existing applications schemas, technologies, and relevant standards into account. Based on the elicited requirements, we have derived the fundamental concepts of an integration framework for E/E product data. In particular, we introduced the terms local and common integration ontologies and gave insights into a corresponding matching approach.

In future work, we will focus on consistency management (Requirement 6) and the evolution of application schemas (Requirements 8 and 9). In addition, we will detail our documentation methodology (Requirement 10). Finally, we will complete our prototype and evaluate its concepts.

Acknowledgements This work has been conducted within the PROCEED[6] project funded by Daimler AG.

[6]**PRO**active **C**onsistency for **EE** product **D**ata management.

References

1. ATESST (2008) EAST-ADL2 specification. Tech rep
2. Aumueller D, Do HH, Massmann S, Rahm E (2005) Schema and ontology matching with COMA++. In: Proc of the 2005 ACM SIGMOD international conference on management of data. ACM, New York, pp 906–908
3. AUTOSAR (2011) AUTOSAR methodology. Tech rep version 1.2.2, release 3.2, rev 0001
4. AUTOSAR (2011) Technical overview. Tech rep version 2.2.2, release 3.2, rev 0001
5. Baumgart A (2010) A common meta-model for the interoperation of tools with heterogeneous data models. In: MDTPI, Paris, France
6. Bestfleisch U, Herbst J, Reichert M (2005) Requirements for the workflow-based support of release management processes in the automotive sector. In: ECEC, pp 130–134
7. Burmester S, Giese H, Niere J, Tichy M, Wadsack J, Wagner R, Wendehals L, Zündorf A (2004) Tool integration at the meta-model level: the Fujaba approach. Softw Tools Technol Transf 6:203–218
8. Chaudhuri S, Dayal U (1997) An overview of data warehousing and OLAP technology. ACM Sigmod Rec 26(1):65–74
9. Cooper R (1990) Stage-gate systems: a new tool for managing new products. Bus Horiz 33(3):44–54
10. Cruz IF, Antonelli FP, Stroe C (2009) AgreementMaker: efficient matching for large real-world schemas and ontologies. Proc VLDB Endow 2(2):1586–1589
11. Ehrig M (2007) Ontology alignment: bridging the semantic gap. Springer, Berlin
12. Euzenat J, Shvaiko P (2007) Ontology matching. Springer, Heidelberg
13. Forsberg K, Mooz H, Cotterman H (2005) Visualizing project management: models and frameworks for mastering complex systems. Wiley, New York
14. Freude R, Königs A (2003) Tool integration with consistency relations and their visualization. In: Proc workshop on tool-integration in system development (TIS 2003), pp 6–10
15. Gonçalves RS, Parsia B, Sattler U (2011) Analysing multiple versions of an ontology: a study of the NCI thesaurus. In: Description logics
16. Halevy A, Rajaraman A, Ordille J (2006) Data integration: the teenage years. In: VLDB, pp 9–16
17. Hallerbach A, Bauer T, Reichert M (2008) Managing process variants in the process life cycle. In: ICEIS (3-2), pp 154–161
18. Hallerbach A, Bauer T, Reichert M (2010) Configuration and management of process variants. In: Rosemann M, von Brocke J (eds) Handbook on business process management, vol 1. Springer, Berlin, pp 237–255
19. International Organization for Standardization (2011) ISO/FDIS ISO 26262-2:2011: road vehicles—functional safety—part 2: management of functional safety
20. Kalfoglou Y, Schorlemmer M (2003) Ontology mapping: the state of the art. Knowl Eng Rev 18(1):1–31
21. Königs A, Schürr A (2006) Tool integration with triple graph grammars—a survey. Electron Notes Theor Comput Sci 148(1):113–150
22. Lenzerini M (2002) Data integration: a theoretical perspective. In: Proc of the twenty-first ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems. ACM, New York, pp 233–246
23. Motik B, Patel-Schneider PF, Parsia B, Bock C, Fokoue A, Haase P, Hoekstra R, Horrocks I, Ruttenberg A, Sattler U et al (2009) OWL 2 web ontology language: structural specification and functional-style syntax. W3C Recomm 27:17

24. Mueller M, Hoermann K, Dittmann L, Zimmer J (2012) Automotive SPICE in practice: surviving implementation and assessment. Rocky Nook, Santa Barbara
25. Müller D, Herbst J, Hammori M, Reichert M (2006) IT support for release management processes in the automotive industry. In: BPM, pp 368–377
26. Müller D, Reichert M, Herbst J (2008) A new paradigm for the enactment and dynamic adaptation of data-driven process structures. In: CAiSE, pp 48–63
27. Noy N, Klein M (2004) Ontology evolution: not the same as schema evolution. Knowl Inf Syst 6(4):428–440
28. Pokraev S, Reichert M (2006) Mediation patterns for message exchange protocols. In: EMOI-INTEROP
29. Pokraev S, Quartel D, Steen MW, Reichert M (2006) Semantic service modeling—enabling system interoperability. In: I-ESA
30. Pratt MJ (2001) Introduction to ISO 10303—the STEP standard for product data exchange. J Comput Inf Sci Eng 1(1):102–103
31. Reichert M (2013) Collaboration and interoperability support for agile enterprises in a networked world: emerging scenarios, research challenges, enabling technologies. In: IWEI, pp 4–5
32. Reichert M, Weber B (2012) Enabling flexibility in process-aware information systems: challenges, methods, technologies. Springer, Berlin
33. Sheth AP, Larson JA (1990) Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput Surv (CSUR) 22(3):183–236
34. SPEEDS Project (2009) SPEEDS L-1 meta-model: deliverable: rev 1.0.1. Tech rep, Information Society Technologies
35. Stamatis D (2003) Failure mode and effect analysis: FMEA from theory to execution. ASQ Press, Milwaukee
36. Tiedeken J, Herbst J, Reichert M (2013) Managing complex data for electrical/electronic components: challenges and requirements. In: BTW workshops, pp 141–150