# Flexible Specification of STEP Application Protocol Extensions and Automatic Derivation of Tool Capabilities

Thorsten Koch[1], Jörg Holtmann[1] and Timo Lindemann[2]
[1]*Software Engineering, Fraunhofer IEM, Paderborn, Germany*
[2]*Emmet Software Labs GmbH & Co. KG, Bad Salzuflen, Germany*

Keywords: STEP, Model-driven Software Development, Meta-modeling, Model Transformation.

Abstract: Original equipment manufacturers (OEMs) build mechatronic systems using components from several suppliers in industry sectors like automation. The suppliers provide geometrical information via the standardized exchange format STEP, such that the OEM is able to virtually layout the overall system. Beyond the geometrical information, the OEM needs additional technical information for his development tasks. For that reason, STEP provides an extension mechanism for extending and tailoring STEP to project-specific needs. However, extending STEP moreover requires extending several capabilities of all involved tools, causing high development effort. This effort prevents the project-specific utilization of the STEP extension mechanism and forces the organizations to use awkward workarounds. In order to cope with this problem, we present a model-driven approach enabling the flexible specification of STEP extensions and particularly the automatic derivation of the required further capabilities for two involved tools. We illustrate and evaluate the approach with an automation production system example.



## 1 INTRODUCTION

The development of mechatronic systems in industry sectors like automation is characterized by complex supply chains, where original equipment manufacturers (OEMs) build an overall system using physical components from several suppliers. An example of such a system is depicted in Figure 1. The OEM integrates this overall automation production system, a so-called *Pick & Place Unit (PPU)* (Vogel-Heuser et al., 2014). The PPU encompasses the four components Stack, Ramp, Crane, and Stamp, which are delivered by suppliers. The Stack works as workpiece input storage and the Ramp acts as workpiece output storage. The Stamp is responsible for labeling the workpieces, and the Crane is responsible for transporting the workpieces by picking and placing them between the different working positions. The Crane transports workpieces from the Stack to the Stamp. After the Stamp has processed a workpiece, the Crane transports the workpiece finally to the Ramp.

Figure 2 sketches the exchange of the product information between the OEM and different suppliers in the development process of a mechatronic system like the PPU. In the course of integrating the overall system, one of the most important development tasks

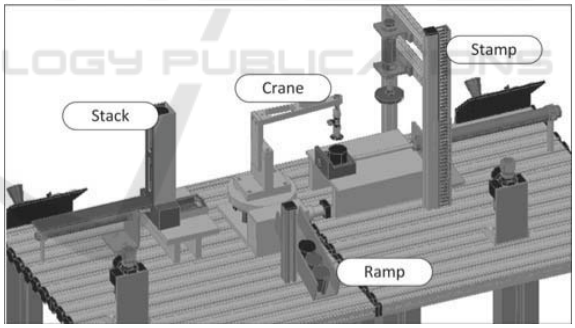


Figure 1: Pick & Place Unit as an example for a simple automation production system (Vogel-Heuser et al., 2014).

of the OEM is to geometrically assemble the overall system based on the particular supplier components. Prior to the actual production of the overall system, this task is performed by means of a virtual geometric layout within computer-aided design (CAD) tools. The suppliers geometrically design their particular components within CAD tools, too. Based on these designs, they provide geometrical information about their components via the standardized data exchange format *STandard for the Exchange of Product data (STEP)* (ISO, 1994), such that the OEM is able to virtually layout the overall system. This is indicated through the arrows labeled STEP-based exchange of



Koch T., Holtmann J. and Lindemann T.
Flexible Specification of STEP Application Protocol Extensions and Automatic Derivation of Tool Capabilities.
DOI: 10.5220/0006137400530064
In *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2017)*, pages 53-64
ISBN: 978-989-758-210-3

geometrical information in Figure 2, which sketches a typical tool chain in a Supplier-OEM relationship.

Beyond the geometrical information, the OEM needs additional technical information (e.g., the permissible payload of the Crane manufactured and delivered by Supplier A and the power consumption of the Stamp from Supplier B in Figure 2) to perform his development tasks. For that reason, STEP provides an extension mechanism for extending and tailoring STEP to project-specific needs. Typical applications of the STEP extension mechanism have been reported in (Usher, 1996; Zha and Du, 2002), for example.

However, extending STEP moreover requires extending the capabilities of all involved tools for the specification, the data exchange, and the interpretation of the additional technical information. That is, for one thing, all affected suppliers have to extend their CAD tools such that they are able to specify and export the additional information. For another thing, the OEM has to extend his CAD tool such that he is able to import and interpret the additional information. These tool extensions have to be implemented through plugins and application programming interfaces on the side of all involved organizations, which causes a high implementation effort. Thus, the application of the STEP extension mechanism is restricted to static, one-off, and long-term tool chains, which do not fulfill the needs of today's and future dynamic business processes (cf. the recommendations for implementing the feature "digital end-to-end engineering" for dynamic value chains in the context of Industry 4.0 (Industrie 4.0 Working Group, 2013)).

The fixedness of the STEP extension mechanism leads to a tool chain as exemplary sketched in Figure 2. Beyond the specification of geometrical information in CAD tools and the corresponding standardized data exchange via unextended STEP, the suppliers specify the respective additional technical information outside their CAD tools. This additional information is awkwardly exported to the OEM via different communication channels (e.g., phone, office documents via mail, or electronic data interchange—EDI—formats (Min, 2000)). In the example in Figure 2, Supplier A specifies the additional information like the power consumption and the admissible payload of his component in Excel sheets and exchanges this information via telephone as indicated trough the arrow Manual exchange via Telephone in Figure 2. Supplier B documents the power consumption in a Word document and exports the information via mail as indicated trough the arrow Manual exchange via Mail in Figure 2. Furthermore, the OEM faces the challenge of component-wisely storing and grouping the geometrical as well as additional information within a product data management (PDM) tool.

In order to cope with this problem, we present in this paper a complex application of existing metamodeling and model transformation techniques that enables the flexible specification of STEP extensions. This particularly includes the automatic derivation of the required capabilities of two involved tools for the specification, the data exchange, and the interpretation of the additional technical information. The two tools comprise a commercial-off-the-shelf CAD tool on the supplier side and a self-developed central data model on the OEM side. The central data model acts as an alternative to a PDM tool, which only has the capability to component-wisely store arbitrary artifacts (like CAD models and documents) but not to interpret model-based information from different sources. We illustrate the approach and conduct a case study with the PPU example.

The remainder of this paper is structured as follows. In the next section, we introduce fundamentals about STEP. Afterwards, we present our model-driven approach in Section 3 and conduct a case study in Section 4. Section 5 covers related work. Finally, Section 6 concludes this paper with a summary and an outlook on open future work.

## 2 ISO 10303 - STANDARD FOR THE EXCHANGE OF PRODUCT DATA (STEP)

The International Organization for Standardization has published the ISO 10303 - STandard for the Exchange of Product data (STEP) (ISO, 1994) to address the problem of exchanging product data between different systems. The overall objective of STEP is to provide a mechanism that describes a complete and unambiguous product definition throughout the entire life-cycle of a product. Furthermore, STEP provides a system independent and computer interpretable file format for the exchange of product data between different software tools, like computer-aided design (CAD) or simulation tools (Kramer and Xu, 2009). However, STEP especially focuses on the representation of geometrical information.

To realize the objective of a complete and unambiguous product definition, STEP defines so-called application protocols (ISO, 1994). An application protocol is a data model tailored to the specific needs of an application area. In the scope of this paper, we use the application protocol STEP AP214. Although the STEP AP214 is originally designed for the auto-

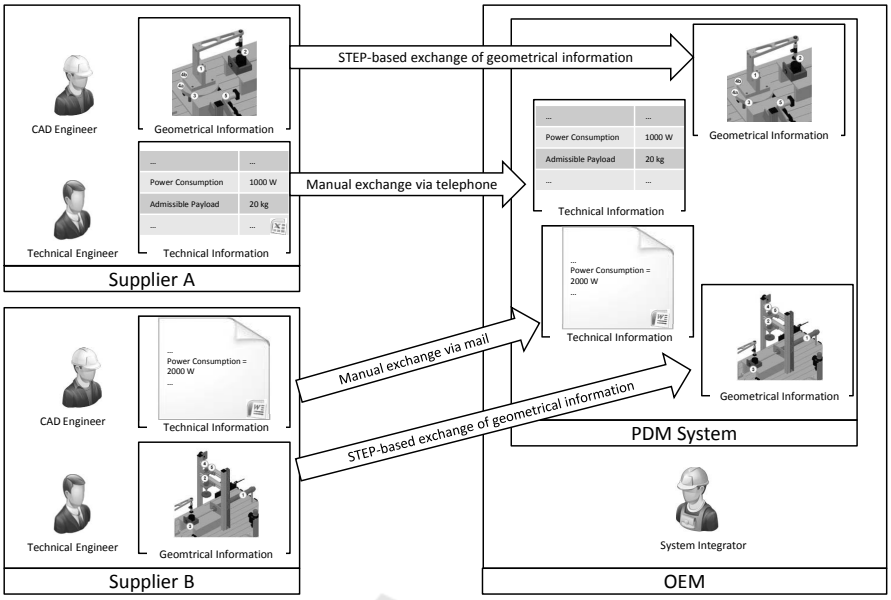


Figure 2: Overview of the exchange of product information between the OEM and different suppliers.

motive domain, it is broadly used in practice, since it describes product information like sheet-metal parts of the car body, mechanical parts of the engine, and glass components. Thereby, the STEP AP214 is also suitable for the exchange of product information in the application of automation production systems.

In an application protocol, the description of product data is defined in the *EXPRESS information modeling language* (ISO, 2004). EXPRESS is part of the ISO 10303 and has been defined to model geometry information. EXPRESS consists of language elements that allow unambiguous data definition and specification of constraints on the defined data. The most important EXPRESS element is the *entity* data type, which defines the objects of interest in the domain being modeled. The *entity* is characterized by its attributes and constraints. The EXPRESS information modeling language also supports various kinds of data types, including *simple types*, *aggregations types*, and *constructed types* (ISO, 2004).

STEP defines two different file formats for the exchange of product data: physical file (ISO, 2002) and XML file (ISO, 2007). Whereas the XML file is an XML encoding for the product data defined by an application protocol, the physical file is a purely ASCII encoding for product data. In the scope of this paper, we use the physical file format, since it is mostly used by exchange systems today to read and write STEP data (Kramer and Xu, 2009).

Figure 3 depicts an overview of the relationship between the EXPRESS information modeling language, a STEP application protocol, and the actual product information contained in a STEP file. The EXPRESS information modeling language has been developed prior to the Meta Object Facility (MOF) (OMG, 2015a) standard of the OMG. However, in terms of the MOF standard, the EXPRESS information modeling language is the meta-meta-model used to specify STEP application protocols by means of a grammar. The STEP application protocol is the meta-model used to specify the structure of the product information. The STEP file is the model containing the actual product information following the structure in the application protocol.

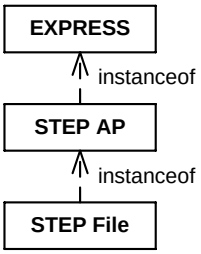


Figure 3: Overview of the relationship between EXPRESS, STEP application protocols and STEP files.

In the remainder of this section, we use the running example of the PPU to illustrate the different parts of the STEP standard. Therefore, Listing 1 depicts an excerpt of the STEP AP214 defined by means of the EXPRESS information modeling language showing the four entities product_context, product, line, and cartesian_point. The product_context contains the single attribute discipline_type of the type la-

bel. The type label represents a STRING. The product contains the attribute id, name and description; all of type STRING. Furthermore, it contains a list of references of product_contexts.

Listing 1: Exemplary excerpt of the STEP AP214 defined in EXPRESS.

```
1  TYPE label = STRING;
2  END_TYPE;
3
4  ENTITY product_context;
5   discipline_type : label;
6  END_ENTITY;
7
8  ENTITY product;
9   id:STRING;
10  name:STRING;
11  description:OPTIONAL STRING;
12  frame_of_reference:SET [1:?] OF
        product_context;
13 END_ENTITY;
```

Listing 2 depicts an excerpt of the physical file of Crane component of Pick&Place Unit. As mentioned before, a physical file is a pure ASCII encoded file with a simple structure. Each line of a physical file encompasses an identifier encoded "'#id"' and a key-value par encoding the actual product information. For example, in Line 1 of Listing 2, the product is defined. The entity has the identifier #86, the id and name HT_L1600. The identifier is also used to encode cross-references between different entities. For example, the entity product contains a reference to the identifier #91.

Listing 2: Exemplary excerpt of a STEP AP214 file.

```
1  #86=PRODUCT('HT_L1600','HT_L1600','',(#91));
2  #91=PRODUCT_CONTEXT(' ',#93,'mechanical');
```

# 3 FLEXIBLE SPECIFICATION OF STEP EXTENSIONS

In this section, we present our model-driven approach for the flexible specification of STEP extensions. Figure 4 depicts an overview of the approach encompassing three main contributions for the OEM and his suppliers to improve the problematic situation described in Section 1. First, the OEM as well as his suppliers are enabled to specify additional technical information directly in their tools (cf. 1 Specification of additional technical information in Figure 4). For this purpose, we enable the OEM to specify a central data model that can be tailored to the specific needs of a particular development project. This central data model contains all geometrical and technical information and is also the main artifact of our approach from which we derive the other parts using model-driven techniques. Furthermore, we provide an extension to the CAD tools of the suppliers based on the STEP extensions specified for the central data model. Second, we are able to derive an automatic data exchange for the involved tools (cf. Automatic exchange of product information in Figure 4). Finally, the specification of additional technical information as well as the automatic data exchange result in a machine-readable and processable representation of the product information (cf. 3 Interpretation of the additional technical information in Figure 4).

In the following section, we describe a systematic model-driven process to support the creation of the central data model. Furthermore, we present the technical details of the two process steps Automatic Derivation of the Central Data Model and Data Import and Generate CAD Extensions in the subsequent sections 3.2 and 3.3, respectively.

## 3.1 Process for the Creation of the Central Data Model

Figure 5 depicts our model-driven process to support the creation of the central data model. The process is specified by means of the Business Process Model and Notation (BPMN) (OMG, 2011). The main contributions of this paper are emphasized in Figure 5 with gray tasks and artifacts. We visualize manual steps by means of BPMN manual tasks (hand in the upper left corner of the task). Steps that we could automate are visualized by means of BPMN service tasks (cogwheel in the upper left corner of the task). Work results are specified by means of BPMN data objects (document icons), and persistent models that are subject to update and retrieval operations are specified by means of BPMN data stores (database icons).

In the following, we exemplarily perform and explain each process step depicted in Figure 5 referring to the PPU as a running example. We design the model-driven process in such a way that the OEM has to perform it, but may need to discuss several aspects with his suppliers.

In the first process step Analyze Requirements, the OEM decides which information is necessary for the current development project and should be stored in the central data model. For the development of the PPU, the OEM decides that the power consumption of all used components and the admissible payload of the Crane must be stored in the central data model besides the regular geometrical information.

In the second process step Select Base Applica-

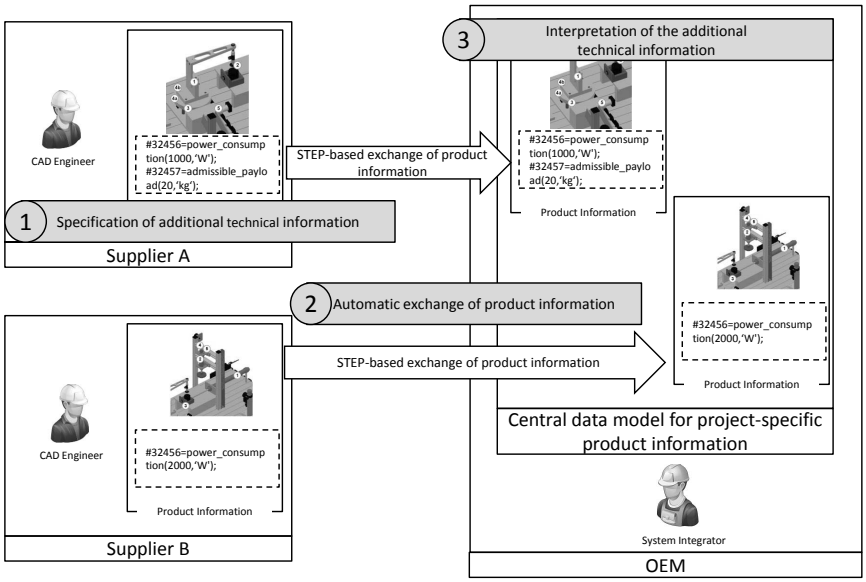


Figure 4: Overview of our model-driven approach for the exchange of product information.

tion Protocol, the OEM selects an application protocol from the STEP Application Protocol library that fulfills most of the analyzed requirements and that acts as a basis for the central data model. The library only contains application protocols that are officially defined in the ISO 10303. In our running example, the OEM decides to use the STEP AP214 as the Base STEP AP.

As mentioned in Section 1, STEP usually does not cover all product information that is needed for the development of the overall system. Hence, the OEM uses the next two process steps Create STEP Extensions and Select STEP Extensions to enrich the selected Base STEP AP with further descriptions of product information. For this purpose, we enable the OEM to create new STEP extensions in an EXPRESS-based textual editor and to store these extensions in a STEP Extension library. Furthermore, we enable him to select existing STEP extensions from the library that satisfy his specific needs.

In our running example, the STEP Extension library already contains several STEP extensions. While reading through the descriptions of these STEP extensions, the OEM noticed that the STEP_EXTENSION POWER_CONSUMPTION depicted in Listing 3 already satisfies the requirements on the specification of a component's power consumption.

Listing 3: STEP extension for the specification of a power consumption.

```
1 SCHEMA STEP_EXTENSION POWER_CONSUMPTION;
2   ENTITY power_consumption;
3     component: product;
4     value: NUMBER;
5     unit: Unit;
6   END_ENTITY;
7 END_SCHEMA;
```

The STEP_EXTENSION POWER_CONSUMPTION only contains the entity power_consumption. This entity refers to the entity product (cf. Section 2) defined in the STEP AP214. Furthermore, the entity power_consumption contains an attribute value of the type NUMBER and a reference to a unit. As mentioned before, this application protocol is sufficient to specify the description of a component's power consumption in a machine-readable manner. Thus, the OEM decides to reuse this STEP extension. Since the STEP Extension library does not contain a suitable STEP extension for the specification of the admissible payload of a Crane component, the OEM defines a new STEP extension STEP_EXTENSION ADMISSIBLE_PAYLOAD as depicted in Listing 4. The structure is analogous to the previous STEP extension. After the OEM has specified the STEP extension, he stores it in the STEP Extensions library to enable its reuse in further development projects.

Listing 4: STEP extension for the specification of an admissible payload.

```
1 SCHEMA STEP_EXTENSION ADMISSIBLE_PAYLOAD
2  ENTITY admissible_payload;
3    component: product;
4    value: NUMBER;
5    unit: Unit;
6  END_ENTITY;
7 END_SCHEMA;
```

After the selection of the required STEP extensions, the automatic derivation process of the

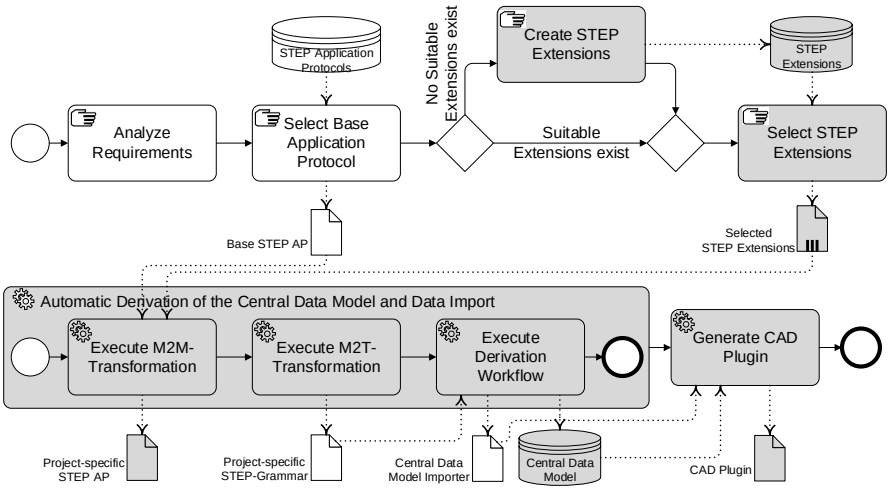


Figure 5: Overview of the model-driven process to support the creation of the central data model.

central data model (cf. Automatic Derivation of the Central Data Model and Data Import in Figure 5) is executed. The automatic derivation process encompasses three subprocesses: Execute M2M-Transformation, Execute M2T-Transformation, and Execute Derivation-Workflow. In the first subprocess, Execute M2M-Transformation, the conceived model-to-model transformation merges the Selected STEP Extensions into the selected Base STEP AP to derive a Project-specific STEP AP. This Project-specific STEP AP contains the description of all product information that should be contained in the central data model. In the subsequent subprocess Execute M2T-Transformation, the OEM executes our developed model-to-text transformation to derive a Project-specific STEP Grammar. This grammar enables the automatic derivation of the central data model and the corresponding import capabilities as described in the subsequent section (cf. Execute Derivation-Workflow in Figure 5).

Finally, the extensions to the CAD tools of the supplier are generated in the process step Generate CAD Plugin. These extensions enable the specification of entities of the central data model within the user interface of the CAD tool. Furthermore, it provides a mechanism to store the product information and to export it to a physical file (cf. Section 2).

Concluding the introduction of the model-driven process, we obtain the specification capabilities of geometrical and additional technical product information by defining flexible STEP extension. The OEM is enabled to describe a central data model by selecting an existing STEP application protocol as basis and by defining and/or selecting STEP extensions to enrich this STEP application protocol. The resulting project-specific STEP application protocol is further used to automatically derive the required capabilities for the data exchange between the OEM and his suppliers. Furthermore, it is used to derive extensions for existing CAD systems to enable the specification, storage, and exchange of additional technical product data needed in the development of mechatronic systems like the PPU.

## 3.2 Automatic Derivation of the Central Data Model and the Data Import

In this section, we describe the realization of the process step Automatic Derivation of the Central Data Model and Data Import depicted in Figure 5. For this purpose, we recreated and developed different meta-models, models, and grammars as depicted in Figure 6. Meta-models are depicted by means of UML classes. Grammars are depicted by means of UML classes with a small rectangle in the upper right corner. Finally, we depicted text files as UML classes with a document icon in the upper right corner and parser as UML classes with a circle in the upper right corner. As the technology icons indicate, we use the Eclipse Modeling Framework (Steinberg et al., 2008) to specify meta-models by means of Ecore models, and the Xtext framework (Eysholdt and Behrens, 2010) to define grammars. While using the Xtext framework, we are able to automatically derive a parser for a particular grammar. Besides the mentioned technologies, we use QVT-O (OMG, 2015a) and Xtend [1] to realize model-to-model and model-to-text transformation, respectively.

As mentioned in Section 2, the EXPRESS information modeling language has been developed in

[1] http://www.eclipse.org/xtend/

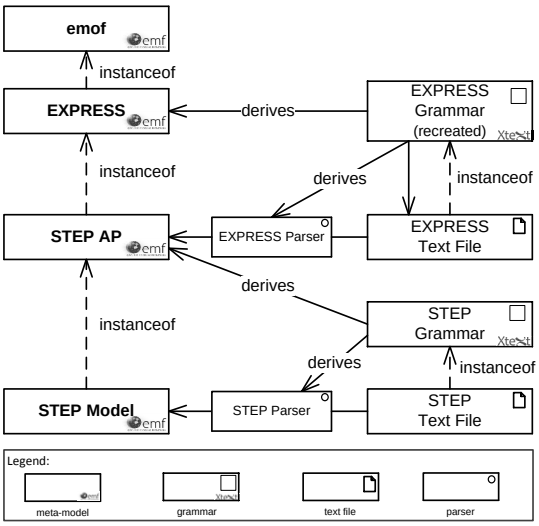


Figure 6: Overview of the developed meta-models and their relationships.

the ISO 10303 prior to the Meta Object Facility (MOF) (OMG, 2015a) standard of the OMG. Thus, the EXPRESS information modeling language does not comply to the OMG standard and modern model-driven development techniques are not yet applicable.

For this reason, we developed our own MOF-compliant meta-model of the EXPRESS information modeling language based on the Eclipse Modeling Framework and the Xtext framework. We used the Xtext framework to recreate the concrete textual syntax of the EXPRESS information modeling language by means of a grammar (cf. EXPRESS Grammar in Figure 6). While using the generation workflow of the Xtext framework, we derive an Ecore-based meta-model of the EXPRESS information modeling language. Furthermore, we derive a EXPRESS parser that reads textual STEP application protocol files that correspond to the defined grammar.

As mentioned in Section 2, a STEP application protocol is defined by means of the EXPRESS information modeling language. Thus, after defining the EXPRESS grammar and deriving its meta-model as well as a corresponding parser, we are able to read and write STEP application protocols. However, in the current stage of our implementation, we are only able to process the basic EXPRESS elements *types* and *entities*. The processing of the remaining EXPRESS elements is left for future work.

A STEP application protocol only defines the structure of the product information, and not the product data itself. Hence, we apply the same technologies to create a grammar representing the product information specified in a STEP application protocol (cf. STEP Grammar in Figure 6). Furthermore, the STEP Grammar defines the structure of the STEP Text File following the structure defined for STEP physical files (cf. Section 2). As depicted in Figure 6, after the execution of the Xtext workflow, we derive a meta-model for STEP files that reflects the product information defined in the STEP application protocol.

Figure 7 depicts the execution of the automatic derivation process of the central data model for our running example by means of a UML Activity Diagram. After the OEM has performed the process step Select STEP Extensions depicted in Figure 5, the specification of the central data model in our running example encompasses the STEP AP214 as Base STEP AP, and the two extensions STEP_Extension Power_Consumption: EXPRESS and STEP_Extension Admissible_Payload: EXPRESS.

In the first activity M2M-Transformation, the selected Base STEP AP and the two selected STEP extensions are merged into an Project-specific STEP AP by using a model-to-model transformation realized by a QVT-O in-place transformation. This model-to-model transformation iterates over all entities in the different :EXPRESS instances and merges them into the Project-specific STEP AP. If a naming conflict occurs or some references are not yet resolved, the transformation resolves these issues.

After the execution of the merging activity, the resulting Project-specific STEP AP is transformed into an Xtext grammar by means of a model-to-text transformation realized by Xtend (cf. M2T-Transformation). The model-to-text transformation also iterates over all entities and translates them into a grammar that also fulfills the requirements of the ISO 10303-21 for the structure of the final STEP File. Listing 5 depicts an excerpt for the resulting Xtext grammar for the STEP extension shown in Figure 4. The Xtext grammar defines the entity power_consumption, encompassing an ID, a desc, and as shown in Listing 3 a value, and a unit. In the final STEP File, the ID corresponds to the line number and acts as an identifier. The desc attribute indicates which entity is currently parsed.

Listing 5: Excerpt of the Xtext grammar for the STEP extension shown in Listing 3.

```
1 power_consumption:
2   name=ID "="
3   desc="power_consumption"
4   "("
5     component = [ product | ID ] ","
6     value= EDouble
7   ")"
8 ",";
```

Finally, the workflow of the Xtext framework is executed and as a result, we derive the central

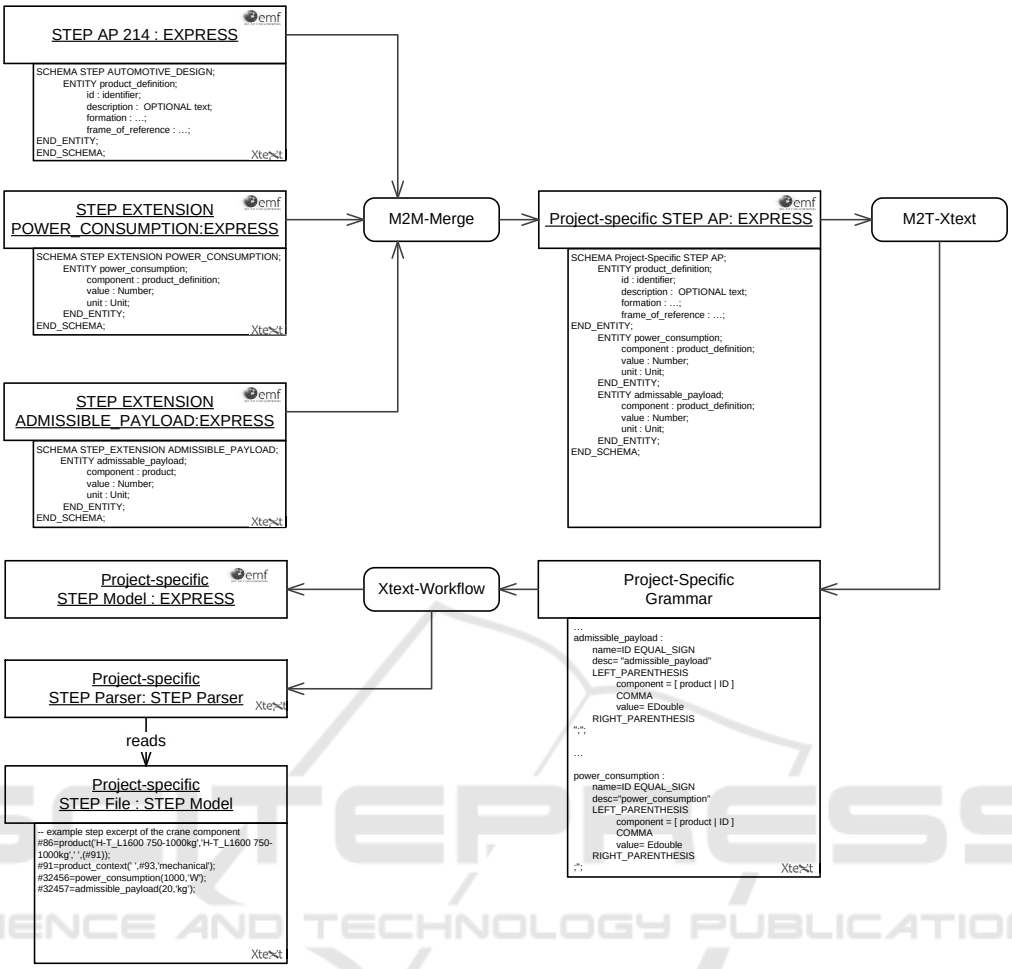


Figure 7: Overview of the automatic derivation of the central data model and the data import.

data model (cf. Project-Specific STEP Model in Figure 7). Furthermore, we derive an Project-specific STEP Parse that reads STEP files and creates data models conforming to the central data model.

## 3.3 Automatic Derivation of the CAD Plugin

In this section, we describe the realization of the process step Automatic Derivation of the CAD Plugin depicted in Figure 5. The automatic derivation approach has been prototypically implemented for the CAD tool SolidWorks.

We started the development of the automatic derivation approach with an examination of the plugin mechanism of SolidWorks and implemented a reference plugin for an extended user-interface representing further technical product information and for exchanging this information.

After implementing the reference architecture, we generalized the reference plugin, and divided the resulting code into platform, individual, and repetitive code. The platform code is provided by the CAD tool SolidWorks to enable the development of plugins using internal functionality of SolidWorks. We encapsulated the CAD tool dependent code by writing a wrapper and refer to it as individual code. Finally, the repetitive code is used to create an extended user-interface and to create the storage functionality. Since this code only uses operations provided by our own individual code, the repetitive code is independent of the used CAD tool.

In the next development step, we developed a CAD plugin generator based on the individual code. The CAD plugin generator uses the Selected STEP Extensions as input and generates the required user-interface elements for the additional technical information. Furthermore, the CAD plugin generator also generates the required code-fragments to support the exchange of the additional information.

Figure 8 depicts the automatic derivation of the

CAD Plugin for our running example. The CAD plugin generator uses the two Selected STEP Extensions STEP_Extension Power_Consumption: EXPRESS and STEP_Extension Admissible_Payload: EXPRESS and produces the user-interface elements on the right.

# 4 CASE STUDY

In this section, we conduct a case study based on the guidelines by (Kitchenham et al., 1995) for the evaluation of our approach. In our case study, we investigate the applicability and usefulness in practice of our approach. We perform the case study based on the running example in this paper and do not aim at generalizing the case study conclusions to all possible development projects using STEP for the exchange of product information.

## 4.1 Case Study Context

The objective of our case study is to evaluate whether our model-driven approach for the creation of a central data model is applicable and useful for the OEM and his suppliers, i.e., whether it reduces the manual effort in deriving tool support for the overall information exchange. For this purpose, we use the two STEP extensions of the running example and different STEP application protocols. We concentrate on the investigation of the applicability and usefulness in practice of our approach especially for the automatic derivation of the central data model and the resulting import capabilities, since the effort for extending the user-interface of the CAD tool compared to the effort of writing a correct parser is much smaller.

## 4.2 Setting the Hypothesis

We define two evaluation hypotheses for our case study. The first evaluation hypothesis *H1* is that our model-driven approach for the exchange of product information between the OEM and his suppliers presented in Section 3 reduces the manual effort in deriving tool support for the information exchange. For the evaluation of *H1*, we define response variables measuring the amounts of entities contained in the input Base STEP AP including its extensions as well as response variables measuring the code size and the generation time of the parser output. That is, we determine the number of entities contained in the selected Base STEP AP plus the number of entities contained in the used STEP extensions (response variable *H1.inputSize*), the amount of lines of code generated in particular for the parser of the central data model (*H1.outputSize*), and the time needed for generating the different code fragments (*H1.outputTime*).

The second evaluation hypothesis *H2* is that our model-driven approach for the exchange of product information produces correct models and correct parser for existing STEP application protocols like STEP AP214 and STEP AP203, and that the parsers process their input files in reasonable time. For the evaluation of *H2*, we define a response variable for measuring the number of STEP files used as input for the parser (response variable *H2.inputSize*). That is, we determine the number of files that are correctly processed without an exception (*H2.outputSize*), and the time needed for processing each file (*H2.outputTime*). To draw conclusion of the processing time, we also determine the time needed to process the same files in SolidWorks (*H2.SolidWorksTime*). We used a typical office computer[2] for all test runs.

## 4.3 Validating the Hypothesis

For the validation of the first evaluation hypothesis *H1*, we executed the model-driven approach several times with different input configurations. First, we used the STEP AP214 as Base STEP AP and the STEP AP203 as Base STEP AP without any further STEP extensions. Furthermore, we used the STEP extensions of the running example in combination with the STEP AP214 and STEP AP203. Finally, we used the STEP AP203 as an extension in combination with STEP AP214 to draw conclusions about the scalability of the approach. The determination of the number of entities contained in the Base STEP AP as well as in the used STEP extensions, the needed generation time, and the lines of code for the parser yields the results as listed in Table 1.

For the validation of the second evaluation hypothesis *H2*, we used the STEP parser generated for STEP AP214 and STEP AP203. As input files, we used free available CAD examples (http://www.steptools.com). This leads to 20 files corresponding to the STEP AP214 and 44 files corresponding to the STEP AP203. The determination of the number of correctly parsed files and the needed parsing time yields the results as listed in Table 2.

## 4.4 Analyzing the Results

The results for *H1* show two aspects. First, depending on the number of entities used for the descrip-

[2]Intel Core i7-4600U @2.10 GHZ, 8 GB DDR3 1066 MHz, 500 GB HDD, Windows 7 Pro 64 bit, Java JDK8u66, Eclipse 4.5

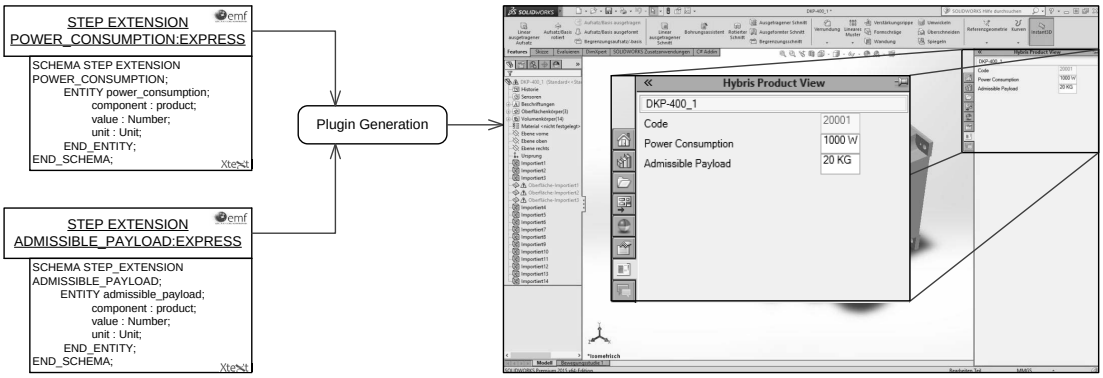


Figure 8: Overview of the automatic derivation of the CAD plugin.

tion of the central data model, the resulting parser encompasses a huge amount of source code. Without a proper tool support, no software developer would be able to produce the parser in the relatively short time. For example, the execution of the model-driven approach uses overall 915 entities and generates 357775 lines of code within 12 minutes (cf. first row of Table 1). Although the generation takes some time to complete, this does not affect the applicability of the approach, since the generation has be performed only once in the whole development project. Second, the model-driven approach scales with the number of entities used in the central data model. Thus, we consider *H1* as fulfilled. The results for *H2* show that the generated parser for the STEP AP214 and STEP AP203 is able to read original STEP files, thus, we conclude that our model-driven approach generates correct parsers. Furthermore, the comparison of *H2.outputTime* and *H2.SolidWorksTime* shows that our parser is not significant slower than the processing of SolidWorks. Thus, we consider *H2* as fulfilled. Concluding the case study, the fulfilled hypotheses indicate that our proposed model-driven approach reduces the manual effort in deriving tool-support for the creation of a central data model and the corresponding import/export capabilities. This gives rise to the assumption that out approach is applicable and useful in practice for the OEM and his suppliers.

The most important threats to validity are as follows: First, we only considered one development example and thus cannot generalize the fulfillment of the statements. Nevertheless, the example represents a typical development project and thus we do not expect large deviations for other examples. Second, the amount of lines of code generated by the approach is only a superficial metric and, therefore, might not reflect the actual development effort. Especially for small extensions like the definition of power consumption, the conceptual complexity of our approach might exceed the effort for the manual extension and/or the manual exchange of this information via another communication channel. However, the manual extension has to be performed for each development project.

# 5 RELATED WORK

STEP provides a standardized mechanism for representing and exchanging product data, and is therefore, considered as a promising product modeling resource. As mentioned in Section 1, the STEP extension mechanism has been used in several applications to describe or analyze a certain aspect of a system and to exchange the corresponding product data. For example, (Usher, 1996) presents an object-oriented product model based on STEP AP224, which defines a standard set of machining features. The authors used their object-oriented product model to support a computer-aided process planning (CAPP) analysis. (Zha and Du, 2002) present a product data exchange using a STEP-based assembly model for the concurrent integrated design and assembly planning. Concluding this paragraph, STEP is widely used in industry and academic to organize product data in a standardized representation. However, in contrast to our approach, most approaches are tailored to one particular use case and are not reusable or interoperable.

To overcome the inflexibility and interoperability, a generic product modeling system has been proposed in (Gu and Chan, 1995) and (Xie and Chen, 2009). These two approaches belong to the most related approaches using STEP to provide a generic product modeling system. However, in contrast to our approach, they do not use model-driven techniques to realize their approach, and, thus, a lot of manual effort has be done for their practical realization.

Other work has focused on applying model-driven development techniques to product data modeling in the design of mechanical systems for the purpose

Table 1: Results of the analysis for *H1*.

| Base STEP AP | Number of STEP extensions | H1.inputSize (# of Entities) | H1.outputSize (LOC) | H1.outputTime (Generation Time) |
|---|---|---|---|---|
| STEP AP214 | - | 915 | 357775 | ≈12 minutes |
| STEP AP203 | - | 254 | 108144 | ≈2 minutes |
| STEP AP214 | 2 | 917 | 358237 | ≈16 minutes |
| STEP AP203 | 2 | 256 | 108626 | ≈3 minutes |
| STEP AP214 | 1 (STEP AP203) | 1169 | 465101 | ≈20 minutes |

Table 2: Results of the analysis for *H2*.

| Base STEP AP | H2.inputSize (# of files) | H2.outputSize (# of correctly parsed files) | H2.outputTime (parsing time) | H2.SolidWorksTime (parsing time Solid Works) |
|---|---|---|---|---|
| STEP AP214 | 20 | 20 | ∅ 57 seconds | ∅ 48 seconds |
| STEP AP203 | 44 | 44 | ∅ 46 seconds | ∅ 39 seconds |

of collaboration and interoperability. For example, (Iraqi Houssaini et al., 2012) present a model-driven architecture for bringing together various product data into a model-driven engineering environment. The engineering environment is used to transform, share, and export the product data enabling the collaboration between different departments and companies involved in the design of mechanical products. (Steel et al., 2011) build a bridge between STEP/EXPRESS and the Eclipse Modeling Framework. The bridge is used to transform models based on the Industry Foundation Classes (IFC), a standardized modeling language, into a format suitable for a particular CAD tool. Beyond the pure storing and sharing of STEP-based product data in a model-driven environment based on the Eclipse Modeling Framework, our approach enables the systematic extension of the STEP standard to provide the exchange of additional technical product data. Furthermore, the OMG has published a standard for a reference meta-model for the EXPRESS information modeling language (OMG, 2015b). This meta-model has been devolved in the so-called Mexico project. However, the standard only focuses on the meta-model for the EXPRESS information modeling language, and does not describe how existing STEP application protocols can be transformed to an instance of the reference model.

Finally, (Yildiz et al., 2014) present ongoing work on a model-driven approach for the specification of product information in the context of PDM. As in our work, the authors state that the initial implementation of a PDM tool, usually does not cover all information needed by the user and that the required extensions to a PDM tool are extensive to implement. Thus, they propose a model-driven approach enabling companies to specify their own business concepts for a PDM tool, resulting in tool extensions to cover the additional information. Our approach and the approach of Yildiz et al. mainly differ in their aim. We use project-specific STEP extensions for the data exchange of product information, whereas Yildiz et al. focus on the extension of the storage capabilities of PDM tools but do not consider data exchange. However, the OEM typically applies a PDM tool with a plain artifact storage mechanism nowadays, as sketched in the introduction. Thus, a complementary combination of both approaches would lead to a more holistic tool chain, as we point out in the future work.

# 6 CONCLUSION AND FUTURE WORK

In this paper, we presented a model-driven approach for the flexible specification of STEP application protocol extensions. This particularly includes the automatic derivation of the required further capabilities for two involved tools. For one thing, we derive a central data model as well as a STEP parser for the import and interpretation tool capability on the OEM side based on the specified STEP application protocol extensions. For another thing, we derive a plugin for the CAD tool SolidWorks for the specification and the export tool capability on the supplier side. Furthermore, our approach supports reusing once specified STEP application protocol extensions. The approach is generic in the sense that arbitrary STEP application protocols can be extended.

The automatic derivation of the required tool capabilities significantly reduces the manual effort that had to be spent on the whole tool chain otherwise. Thereby, we enable the utilization of STEP application protocol extensions for project-specific needs. Moreover, the possibility of reusing extensions reduces the effort on the actual specification of the particular STEP application protocol extensions if an ex-

tension was conceived in prior projects. The generality of the approach enables to handle other parts of the STEP standard beyond the one that we exemplarily extended in this paper.

The future work encompasses several aspects. First, we want to improve the creation of STEP application protocols to support the remaining EXPRESS elements like *where-clauses* and *rules* in the resulting Ecore-based meta-model by means of Object Constraint Language (OCL, (OMG, 2014)) expressions. Second, we want to combine our model-driven approach with classical approaches from product line engineering, like feature modeling, to enable the specification of geometrical and logical constraints in the same formalism. Finally, we want to integrate our central data model into PDM tools to improve their plain artifact storage mechanism with the capability to interpret model-based information.

## ACKNOWLEDGEMENT

This research is partially funded by the Bundesministerium für Wirtschaft und Technologie (BMWi) under the grant ZIM and is managed by the AiF Projekt GmbH. Furthermore, this research is partially funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster "Intelligent Technical Systems OstWestfalenLippe" (it's OWL) and is managed by the Project Management Agency Karlsruhe (PTKA).

## REFERENCES

Eysholdt, M. and Behrens, H. (2010). Xtext: Implement your language faster than the quick and dirty way. In *OOPSLA'10*, pages 307–309. ACM.

Gu, P. and Chan, K. (1995). Product modelling using step. *Computer-Aided Design*, 27(3):163 – 179.

Industrie 4.0 Working Group (2013). Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Final report.

Iraqi Houssaini, M., Kleiner, M., and Roucoules, L. (2012). Tools interoperability in engineering design using model-based engineering. In *ASME 2012*, pages 615–623.

ISO (1994). *ISO 10303-1:1994: Industrial automation systems and integration – Product data representation and exchange –Part 1: Overview and fundamental principles*.

ISO (2002). *ISO 10303-21:2002: Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure*.

ISO (2004). *ISO 10303-11:2004: Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual*.

ISO (2007). *ISO 10303-28:2007: Industrial automation systems and integration – Product data representation and exchange – Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas*.

Kitchenham, B., Pickard, L., and Pfleeger, S. L. (1995). Case studies for method and tool evaluation. *IEEE Software*, 12(4):52–62.

Kramer, T. and Xu, X. (2009). STEP in a Nutshell. In *Advanced Design and Manufacturing Based on STEP*, pages 1–22. Springer.

Min, H. (2000). Electronic data interchange in supply chain management. In *Encyclopedia of Production and Manufacturing Management*, pages 177–183. Springer.

OMG (2011). *Business Process Model and Notation (BPMN): Version 2.0.2*.

OMG (2014). *Object Constraint Language (OCL): Version 2.4*.

OMG (2015a). *Meta Object Facility (MOF) Core Specification: Version 2.5*.

OMG (2015b). *Reference Metamodel for the EXPRESS Information Modeling Language (EXPRESS): Version 1.1*.

Steel, J., Duddy, K., and Drogemuller, R. (2011). A transformation workbench for building information models. In *ICMT 2011*, pages 93–107. Springer.

Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2008). *EMF: Eclipse Modeling Framework*. Addison-Wesley, 2nd edition.

Usher, J. M. (1996). A STEP-based object-oriented product model for process planning. *Computers & Industrial Engineering*, 31(1-2):185–188.

Vogel-Heuser, B., Legat, C., Folmer, J., and Feldmann, S. (2014). Researching evolution in industrial plant automation: Scenarios and documentation of the pick and place unit. Technical report, Institute of Automation and Information Systems, Technische Universität München.

Xie, Q. S. and Chen, W.-L. (2009). A Generic Product Modelling Framework for Rapid Development of Customised Products. In *Advanced Design and Manufacturing Based on STEP*, pages 331–352. Springer.

Yildiz, O., Aouadi, N., Karkouch, A., Pernelle, P., Gzara, L., and Tollenaere, M. (2014). MDA Based Tool for PLM' Models Building and Evolving. In *APMS 2014*, pages 315–322. Springer.

Zha, X. and Du, H. (2002). A PDES/STEP-based model and system for concurrent integrated design and assembly planning. *Computer-Aided Design*, 34(14):1087 – 1110.