Article

Compatibility Improvement of Interrelated Items in Exchange Files—A General Method for Supporting the Data Integrity of Digital Twins

Johannes Mohr *, Claudia Kleinschrodt, Stephan Tremmel and Frank Rieg

Engineering Design and CAD, University of Bayreuth, Universitaetsstr. 30, 95447 Bayreuth, Germany
* Correspondence: johannes.mohr@uni-bayreuth.de

Abstract: Stakeholders in the industry are increasingly using digital twins to take advantage of continuous digitization. The widely used methods for transferring partial models of digital twins within various heterogeneous systems rely on standardized, neutral file-based exchange. However, using differently implemented routines in the pre- and postprocessors of the systems engaged during data transmission leads to compatibility problems. Complete information transfer is not guaranteed, although potentially all information is available in the individual exchange file. To utilize the full potential of digital twins, this paper presents a method for directly adapting the content stored in an exchange file to systematically achieve compatibility. In the first step, we define a general structure to specify interrelated, nonconforming objects that are stored in the exchange file. We present five conditions that specify a compatibility problem in the following steps. On this basis, the applicant can solve various exchange problems for the indicated scenario in the third step. After explaining the approach in general terms, we demonstrate its generality by discussing two diverging use cases based on the exchange formats STEP and INP. We implemented the method in software terms, and the implementation indicates that this method can fix compatibility problems in an automated way.

Keywords: Abaqus INP; compatibility improvement; data transfer; digital twin; industry 4.0; STEP

check for updates

Citation: Mohr, J.; Kleinschrodt, C.; Tremmel, S.; Rieg, F. Compatibility Improvement of Interrelated Items in Exchange Files—A General Method for Supporting the Data Integrity of Digital Twins. *Appl. Sci.* **2022**, *12*, 8099. https://doi.org/10.3390/app12168099

Academic Editor: Dimitris Mourtzis

Received: 12 June 2022
Accepted: 10 August 2022
Published: 12 August 2022

1. Introduction

Flexible, individualized mass production is increasingly dominating the industry. In addition, companies see themselves confronted with enormous competitive pressure. To meet these challenges, companies are using digital twins to harness the advantages of powerful computers [1–4]. To date, there is no distinct definition of the term digital twin [5,6]. However, the broad consensus is that it represents a virtual image of a physical asset, meaning that a digital twin needs a physical complement. Sensors or databases connect the corresponding asset and its virtual representation. Overall, the use of digital twins offers a variety of benefits. For example, they enable prototypes to be tested in detail in the early product development phase. Moreover, they provide the possibility of indicating defects or of monitoring and regulating processes autonomously during the usage phase of the product. In addition, digital twins allow the applicant to determine the reliability of a product at the end of its life. This then helps to prevent early, uneconomical disposal [5,7,8].

A digital twin usually consists of several partial models that are used in computer-aided design (CAD) or electronic systems, for example. These models either represent a three-dimensional (3D) geometry or characterize two-dimensional (2D) layouts, such as integrated circuits. Further partial models include analysis models used in computer-aided engineering (CAE) systems to perform calculations, such as strength analyses, flow simulations, and electromagnetic field analyses [9–12].

Various exchange files store and transfer partial models. The data integrity of digital twins and their partial models saved in exchange files is essential. This paper refers to the general classical definition of data integrity according to Courtney [13]. For Courtney, data have integrity if they meet or exceed requirements related to the user's expected quality. However, requirements are highly context-dependent. In general, we can summarize that integer data must be error-free, up-to-date, complete, self-explanatory, and reliable [13–16].

Providing and processing high-quality data still poses a severe problem, however. For instance, exchanging information between different systems is a frequent source of data quality issues. Moreover, the correct interpretation of an exchange file is not always possible despite the presence of all potential information. One leading cause for this is a deviating implementation of export and import routines in the pre- and postprocessors. This means that a preprocessor of a source system writes out conflicting information that the postprocessor of a target system cannot interpret correctly. Frequently, the standards on which the exchange format is founded merely represent framework conditions for implementation. A correct interpretation remains open to the system manufacturers involved. Adopting a transfer file to the requirements of a target system guarantees error-free import [15,17]. Accordingly, this paper shows a general method for solving exchange problems by adapting the information stored in an exchange file to ensure that a postprocessor can interpret the file without compatibility errors.

To demonstrate this method, we proceed as follows. Section 2 first explains basic information regarding data exchange and extraction, as well as regarding file formats used for transferring partial models. It also provides an overview of the Standard for the Exchange of Product Data (STEP) and Abaqus Input (INP) exchange formats. Section 3 describes the proposed compatibility improvement method. We first explain the three steps necessary to increase the compatibility of a transfer file. After providing an overview of the approach, we explain each step in detail. Section 4 outlines two use cases based on the INP and STEP formats explained in the previous sections. To validate the method, we implemented it in the form of a software solution, which is addressed in Section 5. This is followed by a discussion of the results in Section 6 and the conclusions in Section 7.

2. Materials

2.1. Fundamentals of Data Exchange and Extraction

In recent years, the importance of neutral file-based exchange methods has increased. To ensure continuous, loss-free data flow between different systems, several standards have been developed, such as Drawing Exchange Format (DXF) [18], INP [19], and STEP [20]. To provide semantic interoperability, researchers increasingly use ontologies to exchange geometric and semantic product information consistently, especially in the context of digital twins and the semantic web. For this purpose, different possibilities for creating an ontology are available, such as the open source ontology editor Protégé developed by Stanford University [21]. Standardized languages, such as the Web Ontology Language (OWL), are widely used for the creation process [22]. The translation is usually done by mapping the data semantics using an intermediate common model data ontology. Up- and downstream are either system-specific ontologies or neutral interchange formats. However, conversion of the complete content may result in information loss, or in the worst case, direct mapping is not possible [23–27]. Furthermore, none of the approaches consider directly adapting the contents of a file to establish compatibility between systems, thus minimizing the conversion process and loss. Ontology exchange methods are also difficult to apply when data are not available in an analog ontology form. To intercept a wide range of different partial models, the present work focuses on standardized, neutral file-based exchange since this type of exchange is still standard and widely used [9,15,25,28].

During data exchange through neutral, system-independent exchange files, the source system exports the file in the first step. Here, a preprocessor converts the system-specific internal binary format to a neutral data format. A postprocessor in the target system converts the file to the internal structure of the target system [29,30]. Figure 1 shows the process below.

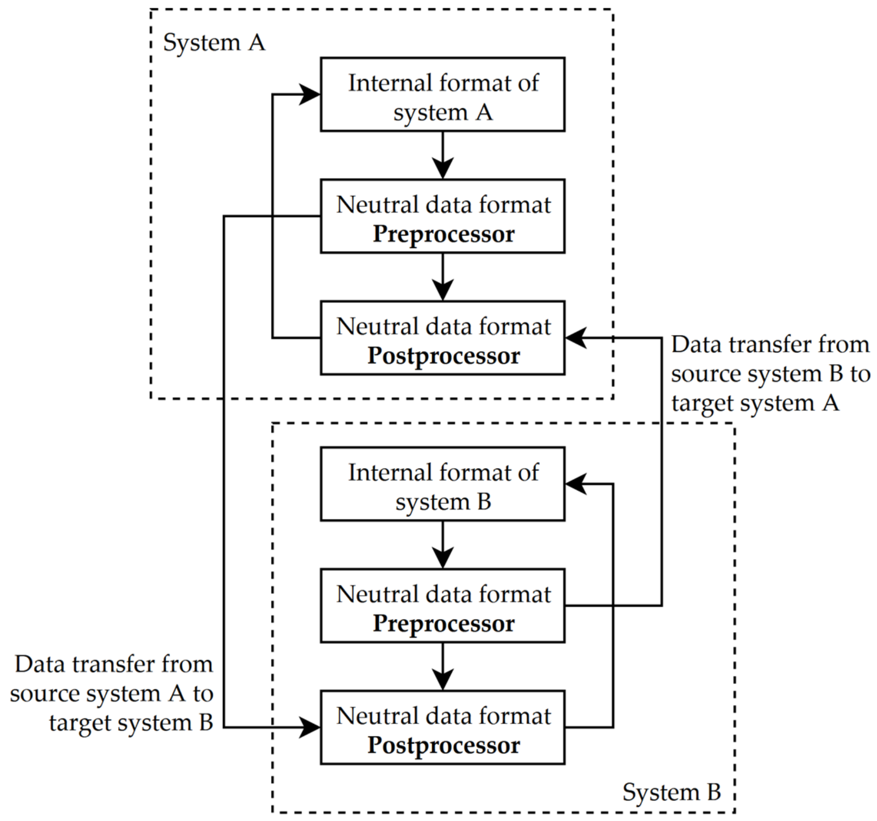

System A
Internal format of system A
Neutral data format **Preprocessor**
Neutral data format **Postprocessor**
Data transfer from source system B to target system A
Internal format of system B
Neutral data format **Preprocessor**
Neutral data format **Postprocessor**
Data transfer from source system A to target system B
System B

Figure 1. Process flow of neutral file-based data exchange.

To manipulate data internally, directly at the file level, we first need to extract it and make it accessible. Using external methods, such as application programming interfaces (APIs), is unsuitable because they do not allow information to be adapted to another target system [31]. There are several approaches in the engineering context in the sub-field of academic research on feature extraction. Some methods are not advanced enough, while others have been replaced with more efficient techniques [29,31–34]. In general, graph-based approaches [35–37], heuristic-based methods [38–42], volume decomposition, and, most recently, artificial neural networks [33,43,44] are currently being used. The present contribution refers to heuristic approaches, as they offer the possibility of extracting items and specifying context-specific compatibility issues precisely. In the course of heuristic-based methods, features are generalized as templates consisting of characteristic rule patterns. If the predefined conditions are matched, then the appropriate pattern is recognized as the respective feature [38–42].

2.2. *Fundamentals of File Formats*

A file format is a convention or set of rules for representing data in a file. Structuring allows the underlying communication process to be standardized and facilitates the comparability of further data processing [45].

A large number of different file formats exist for storing information. Specific formats have been established for diverse areas of use. For example, the German Institute for Standardization (DIN) 26100 provides an overview of the potential formats for each domain. The norm defines a structure for the data exchange of standardized and non-standardized products. A compressed container file bundles all relevant partial models saved in exchange files. The applicator creates a separate folder in the container for each area of use in the root directory. The standard specifies a non-exhaustive selection of formats stored in the respective folder, which Table 1 lists [9].

Table 1. Valid file formats in a container file per area of use, according to DIN 26100 [9].

File Format	Area of Use	Ordinary Encoding
AVI	Multimedia	Binary
DXF	2D-documentation, -graphics, -contours	Text
GSD, TXT	General descriptions	Text
H, I	Computer numerical control (CNC) programming	Text
IO	General descriptions	Binary
JPG, PNG	Image, sketch, multimedia	Binary
JT	3D-representations—detailed and basic	Binary
NC	CNC programming	Binary
P21	ISO properties	Text
PDF	Catalog data, 2D-documentation	Binary
STEP, STL, INP	3D-representations (detailed and basic)	Text
XML	International Standards Organization (ISO) and DIN properties, general descriptions, catalog data, documentation, application data	Text

Inside a transfer file, the information is encoded in binary or plain text, mainly using the American Standard Code for Information (ASCII) or the Universal Coded Character Set Transformation Format (UTF-8). In general, both binary and plain text-encoded files are suitable for manipulation. Since text-based files are easier to trace than non-text files, the present work focuses on this file type. Hex editors also allow tracking and adaptation of the byte sequences of a binary file [45].

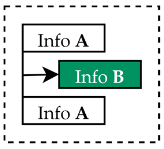
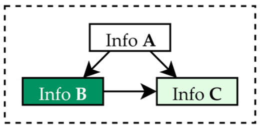
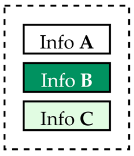
Regarding file adaption, the relationships within a file are of particular importance. If, for example, information A references information B, a change to A may lead to the invalidity of B. The three following basic referencing types can be distinguished:

- Hierarchy—A tree structure arranges the stored data. A root element represents the top level and contains the entirety of the information saved in the document. Child elements divide the root element, which can be subdivided into deeper nestings. The lowest level in each case represents the direct data.
- Relation—Each stored piece of information has a unique identification key that links the data using references. The keys can be either numbers or strings.
- Block—Blocks in the file store the data. Each block represents a coherent piece of information, and unions have no or only a weak connection to one another.

One of the commonalities among the reference types is that single units, which we refer to as "items," bundle logically related information. In a given communicative context, a single piece of information is a minor representation of facts that can be interpreted independently and fixed permanently as discrete signs [46]. The items are advantageous because they can be extracted, analyzed, and improved. Table 2 below shows three exemplary items based on the hierarchical Extensible Markup Language (XML) format, the relational format STEP, and the block Heidenhain (H) format [20,47,48].

The method presented in this paper for resolving compatibility issues is suitable for all three reference types mentioned. Special considerations arise due to the identification of keys for relational structures, which is why the present contribution focuses on analyzing the representative, relational, and widespread exchange formats STEP and INP.

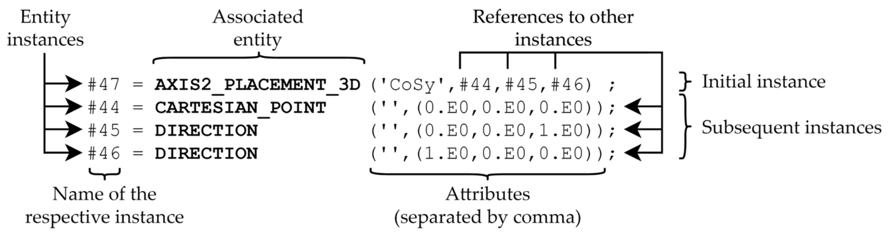
Table 2. Basic reference types in exchange files.

Reference Type	Schematic Representation	File Format	Exemplary Area of Use	Exemplary Item
Hierarchy		XML	General Description	Product feature
Relation		STEP	3D-representations	Coordinate system
Block		H	CNC programming	Drill hole

2.3. Fundamentals of the STEP File Format

STEP is an international standard for exchanging, storing, archiving, and transforming product data, which the ISO 10303 series defines. From the beginning, the focus of the conceptual design is on the representability of industry-independent product information for the entire life cycle. Consequently, applicants use the standard in a wide variety of application areas, such as CAD, computer-aided manufacturing (CAM), and product data management (PDM) systems [49]. Due to this wide range of uses, separate application protocols (AP) define these areas. Furthermore, individual parts of the standard describe single protocols. All examples in this paper rely on AP 214 (“Core data for automotive mechanical design processes”) due to its sufficiently wide distribution [15,50].

A STEP file, whose formal structure is defined in ISO 10303-21 [51], is provided as an uncompressed ASCII file. Figure 2 shows a section of a STEP file. The reader can see that instances of entities store the information line by line. Entities defined in APs are comparable to classes in conventional object-oriented programming languages. Accordingly, an AP specifies unique entities that can be instantiated multiple times in a STEP file. Each piece of information holds an inimitable identification key (e.g., #47). Furthermore, an entity can obtain various attributes, such as strings and numerical values, which are appended to the instance in parentheses. Referencing different instances is done by specifying the key of the referenced information as an attribute.



```
#47 = AXIS2_PLACEMENT_3D ('CoSy',#44,#45,#46) ;
#44 = CARTESIAN_POINT    ('',(0.E0,0.E0,0.E0));
#45 = DIRECTION          ('',(0.E0,0.E0,1.E0));
#46 = DIRECTION          ('',(1.E0,0.E0,0.E0));
```

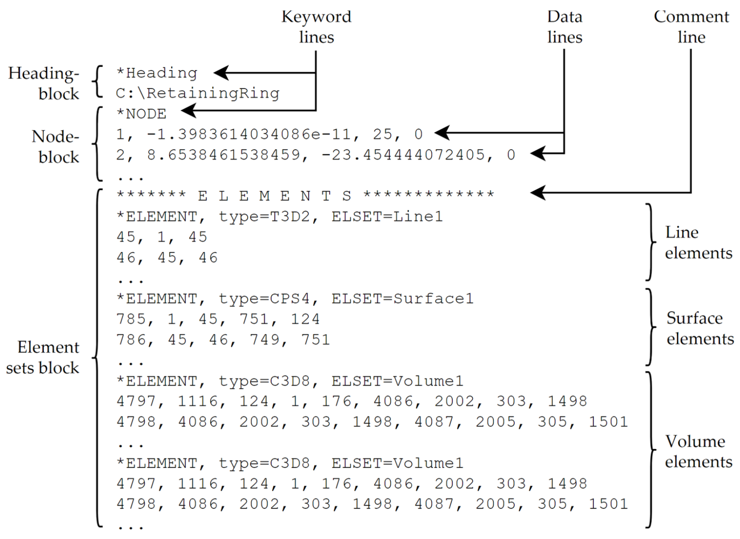
Entity instances
Associated entity
References to other instances
Initial instance
Subsequent instances
Name of the respective instance
Attributes (separated by comma)

Figure 2. Required instances in a STEP-AP214-exchange file to represent the item coordinate system in the CAD system Creo Parametric 6.0.5.0.

An item bundles different, related instances. Figure 2, for example, shows the requirements in a STEP exchange file for representing a 3D coordinate system in the CAD program Creo Parametric 6.0.5.0. The starting point of the item is an initial instance of the entity AXIS2_PLACEMENT_3D (#47), which links subsequent unit parts. The initial reference refers to the three following instances (#44, #45, #46). The entity CARTESIAN_POINT (#44) defines the origin of the coordinate system. In our case, this matches the global coordinate system. Accordingly, the x-, y-, and z-coordinates equal 0.E0. Two instances of the entity DIRECTION (#45 and #46) define the orientation of the two axes of the coordinate system. The third axis is orthogonal to the two directions and is not part of the write-out of the preprocessor [15].

2.4. Fundamentals of the INP File Format

Abaqus input files are encoded in ASCII and contain finite element analysis models. They store nodes, elements, materials, and the initial conditions of the respective model for performing simulations in the CAE context. Since the format documentation is publicly provided, systems other than the CAE software Abaqus also use specifications to import and export analysis models [19]. Figure 3 below shows an example of the basic structure of an INP file for saving geometry, as this knowledge is needed to follow the described use case in Section 4.2.



```
*Heading
C:\RetainingRing
*NODE
1, -1.3983614034086e-11, 25, 0
2, 8.6538461538459, -23.454444072405, 0
...
******* E L E M E N T S *************
*ELEMENT, type=T3D2, ELSET=Line1
45, 1, 45
46, 45, 46
...
*ELEMENT, type=CPS4, ELSET=Surface1
785, 1, 45, 751, 124
786, 45, 46, 749, 751
...
*ELEMENT, type=C3D8, ELSET=Volume1
4797, 1116, 124, 1, 176, 4086, 2002, 303, 1498
4798, 4086, 2002, 303, 1498, 4087, 2005, 305, 1501
...
*ELEMENT, type=C3D8, ELSET=Volume1
4797, 1116, 124, 1, 176, 4086, 2002, 303, 1498
4798, 4086, 2002, 303, 1498, 4087, 2005, 305, 1501
...
```

Figure 3. Basic internal structure of geometry of an Abaqus INP file.

Different blocks divide the file content into discrete areas, with each area introduced by a keyword line. A keyword line starts with precisely one asterisk, while at least two asterisks introduce a comment line, which a postprocessor ignores during parsing. A keyword line can contain optional parameters in addition to the introductory label, and commas separate the parameters from one another. Furthermore, the heading block describes the name of the analysis (C:\RetainingRing). The data lines of the following node block then give the x-, y-, and z-coordinates of the mesh nodes describing the geometry. The first number per data line represents the corresponding node number and is assigned in ascending order. The element sets grouped by type follow the nodes. Figure 3 presents the two-node truss element (T3D2), four-node rigid element (CPS4), and eight-node brick element (C3D8) types. The T3D2 elements represent lines, the CPS4 elements characterize surfaces, and the T3D2 elements show volumes. Lastly, referenced, summarized nodes build up the elements. For example, the nodes with references 1, 45, 751, and 124 compose the surface with reference 785 from Figure 3 [19,52,53].

3. Proposed Method

3.1. General Description of the Steps of the Compatibility Improvement Approach

To adapt an exported exchange file, we suggest the three steps shown in Figure 4. Knowledge of the target system-compliant item structure (TSI) of the respective item is indispensable for determining the cause of the error. In addition, to detect the faulty item in the transfer file and identify differences, we also need to use the source system-compliant item structure (SSI). Accordingly, the first step covers the universal, file-independent definition of SSI and TSI. In the second step, we suggest comparing the SSI and TSI instances to highlight the discrepancies that cause misinterpretation. Pairwise conditions for the respective instances are suitable for this purpose, as described in the following subsection. Knowledge about the SSI, TSI, and connecting conditions allows us to adapt corresponding items in different exchange files in the last step.

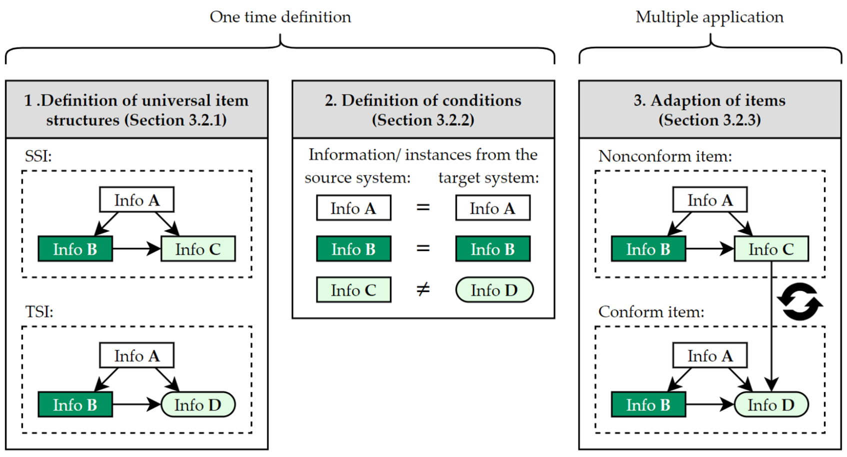

One time definition
Multiple application
1 .Definition of universal item structures (Section 3.2.1)
SSI:
Info A
Info B
Info C
TSI:
Info A
Info B
Info D
2. Definition of conditions (Section 3.2.2)
Information/ instances from the source system: target system:
Info A = Info A
Info B = Info B
Info C ≠ Info D
3. Adaption of items (Section 3.2.3)
Nonconform item:
Info A
Info B
Info C
Conform item:
Info A
Info B
Info D

Figure 4. Schematic illustration of the compatibility improvement approach.

3.2. Description of the Individual Steps of the Compatibility Improvement Approach

3.2.1. First Step—Definition of Universal Item Structures

In the course of defining universal item structures, we recommend analyzing a transfer file from the source and target systems separately. This procedure helps identify the entities and instances for the respective unit. In both programs, the item to be identified has to be created and exported analogously. It is essential that only the required information to represent an item is determined. Various methods are suitable for the recognition process, as has been elaborated on by previous researchers [15,54].

However, a challenge arises with the extracted units. The determined structures only refer to the considered file and cannot be used independently of the use case. This occurs due to internal factors, such as deviating keys, references, and attributes. For example, the name, position, and orientation of a coordinate system all depend on the user's preferences. We propose using operations representing parts of the item to describe these discrepancies.

Moreover, we distinguish between special and standard operations. The former are parts that can differ across items, such as the orientation and the specific name of a coordinate system. Reference and differentiation operations are sub-specifications of special operations. With standard operations, however, we do not subdivide further. We assume that this categorization is sufficient for fully describing diverse items, as we have tested it on various examples. Our work resulted in the following three operation types:

- Reference operation—This is a part of an instance representing a key or reference within an item.
- Differentiation operation—This is a segment of an instance that describes attributes that vary across items.
- Standard operations—This is a part of an instance without any special meaning. It is a static string that does not differ across items, and standard operations serve as the basic framework.

For example, Figure 5 shows the operations for the item coordinate system from Figure 2. The keys and references, such as #47, are declared as reference operations. The name, the values of the coordinates of the origin, and orientation represent differentiation operations since they may differ across items. As all other data are standard operations that are identical for each item, we transfer them directly to Figure 5.

```
ref1 = AXIS2_PLACEMENT_3D ('att1',ref2 ,ref3 ,ref4);
ref2 = CARTESIAN_POINT    ('',(att2, att3, att4));
ref3 = DIRECTION          ('',(att5, att6, att7));
ref4 = DIRECTION          ('',(att8, att9, att10));
```

refx	= reference operation	special operations
attx	= differentiation operation	
text	= standard operation	

Figure 5. General structure of the item coordinate system for the CAD system Creo Parametric 6.0.5.0.

In summary, the result of the first step is a universal, file-independent item structure for both the source and the target system.

3.2.2. Second Step—Definition of Conditions

In this step, it is necessary to highlight the differences between SSI and TSI by defining opposing conditions. Analyzing the determined structures from the previous step provides the basis for identifying error causes during imports. The relationships are, in many cases, non-trivial. Moreover, repeated modification of the exchange file from the source system according to the required instances for an error-free import into the target system helps trace correlations.

An examination of various use cases shows that the differentiation of the five conditions is sufficient to cover all the considered scenarios. To begin, we distinguish between conditions for instances that exist in both the SSI and the TSI:

- Homogeneous—An instance is needed in the source and the target system to interpret correctly and is identical in both setups.
- Heterogeneous—An instance is present in both structures to be compared but differs in certain places. For example, a reference to another instance is missing.

Moreover, we develop the following three conditions to identify instances found either in the SSI or TSI:

- Positive—An instance only exists in the TSI, and this is necessary for an error-free import.
- Negative—An instance only exists in the SSI, and this leads to a misinterpretation in the target system.
- Neutral—An instance only exists in the SSI, and this does not lead to a misinterpretation in the target system.

We assume that this condition classification is sufficient to describe a large number of different use cases. To our knowledge, there are no more cases to distinguish between.

3.2.3. Third Step—Adaptation of Items

Before the actual adaptation, we extract the items under consideration. Since the file originates from the source system, the SSI is the basis for the extraction process. Regular expressions (regex) are particularly suitable for the search. A regex describes an abstract pattern that matches a string of characters [55]. In particular, a regex is ideal for improved tracking. Furthermore, wildcards offer the opportunity to reach special operations. Search patterns can directly include standard operations since they represent the basic structure of an item and do not differ across items.

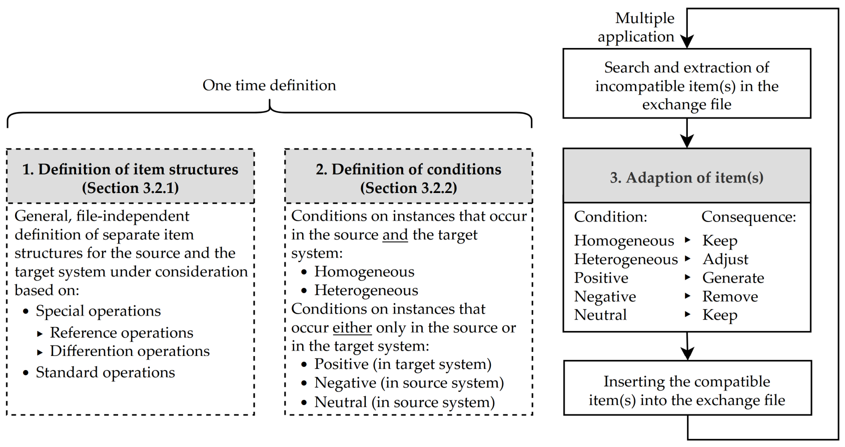
If items are available, the applicant should first check whether they are already compatible. Conformity is present if only homogeneous or neutral conditions exist. Heterogeneous, positive, or negative conditions imply that individual instances of the item structures between the source and target systems differ. These then serve as the cause of the compatibility problem.

If incompatibility can be determined, all defined conditions must be processed iteratively to consider all relationships and progressively create the compatible item. It is advantageous to start with the conditions on the SSI since newly generated information of the TSI may refer to already existing instances. Conditions involving SSI are either homogeneous, heterogeneous, negative, or neutral. We keep information with a homogeneous condition unchanged since they do not differ. We need to remove the error-causing negative instances and adapt differing heterogeneous instances. Neutral conditions are a particular case. We can either keep or remove them since they are neither needed for a correct interpretation nor interfere with it. However, as retention ensures compatibility with the source system, this option is preferred.

Subsequently, iteration of the instances of the TSI takes place. Here, we only consider positive ones since the pairwise homogeneous and heterogeneous conditions are already handled during the SSI iteration. Moreover, negative and neutral conditions occur only in connection with instances of SSI. We need to generate information with a positive condition since it does not exist in the transfer file so far. The values for the special operations result from references within the item and permissible values from the context or standards on which the format is based.

At this point, we note a dilemma related to negative, positive, and heterogeneous conditions. More precisely, removing an instance with a negative condition, generating new information with a positive condition, and modifying data with a heterogeneous condition can all lead to misinterpretations in the source system. This circumstance is difficult to avoid. Consequently, creating a functional transfer file for both the source and target systems is challenging. We recommend a repeated inverse source system conform adjustment of the transfer file to reimport it error-free into the originating system.

In summary, the results of the iterations of the third step are conform and insertable items. Figure 6 represents an overview of the conditions with their respective consequences on the right side and a summary of the first two steps described in the previous two subsections, outlined in dashed lines.



One time definition
1. Definition of item structures (Section 3.2.1)
General, file-independent definition of separate item structures for the source and the target system under consideration based on:
• Special operations
▸ Reference operations
▸ Differention operations
• Standard operations
2. Definition of conditions (Section 3.2.2)
Conditions on instances that occur in the source and the target system:
• Homogeneous
• Heterogeneous
Conditions on instances that occur either only in the source or in the target system:
• Positive (in target system)
• Negative (in source system)
• Neutral (in source system)
Multiple application
Search and extraction of incompatible item(s) in the exchange file
3. Adaption of item(s)
Condition: Consequence:
Homogeneous ▸ Keep
Heterogeneous ▸ Adjust
Positive ▸ Generate
Negative ▸ Remove
Neutral ▸ Keep
Inserting the compatible item(s) into the exchange file

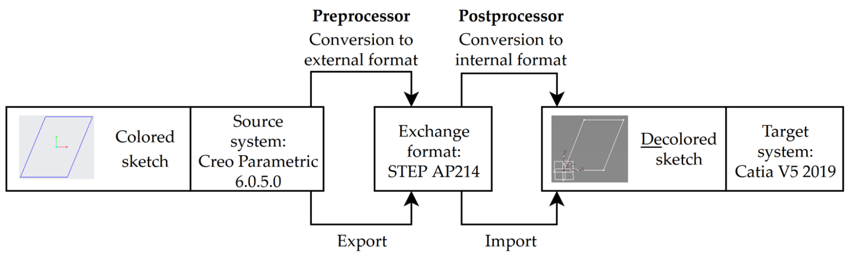
Figure 6. Key findings of the item adaption.

4. Case Studies

This section describes two use cases that demonstrate the method developed in the previous section. The first example, the exchange of a sketch, can be classified as moderate. The postprocessor of the target system used here is capable of importing the sketch. However, certain necessary information is lost. We rank the second example, the transfer of a retaining ring model, as severe because the target system cannot import the object. We want to highlight that the described examples are relatively simple compatibility use cases. In other words, they are trivial enough to show that the approach is developed yet complex enough to address all relevant aspects. Nevertheless, the method presented in this paper is also suitable for more complex use cases and other exchange formats.

4.1. Sketch Exchange via STEP

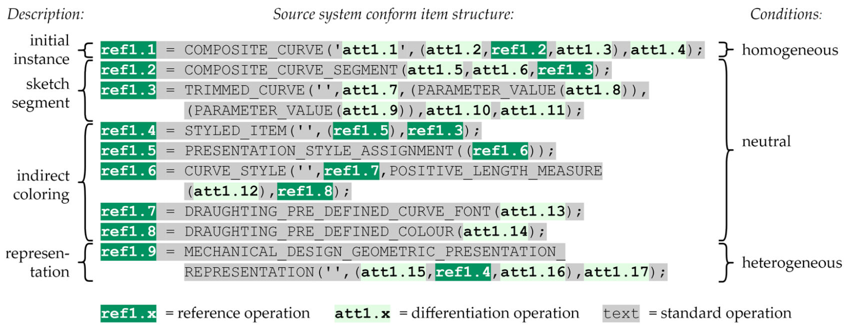
As the first example, we use the exchange of a rhombic-shaped sketch between the CAD systems Creo Parametric 6.0.5.0 (Parametric Technology Corporation, Bosten, MA, USA) and Catia V5 2019 (Dassault Systèmes, Vélizy-Villacoublay, France), which Figure 7 illustrates below. Sketches are a commonly used design element. A significant feature is that they are colored after being transferred from a source to a target system. For example, automatic recognition uses coloring to determine specific geometries [56]. However, the import of a colored sketch, which is created in Creo and exported as a STEP-AP214 file and then imported into Catia with standard settings, results in a loss of color information, even though potentially all data for the coloring is available in the transfer file [15]. As a result, the sketch is blanket colored in white by the postprocessor of Catia.



Preprocessor
Conversion to external format
Postprocessor
Conversion to internal format
Colored sketch
Source system: Creo Parametric 6.0.5.0
Exchange format: STEP AP214
Decolored sketch
Target system: Catia V5 2019
Export
Import

Figure 7. Lossy data exchange of a sketch between the CAD systems Creo and Catia.

The analysis of a STEP-AP214 exchange file from Creo and Catia results in generally valid item structures in the first step. Each file stores a sketch, and Figures 8 and 9 illustrate the extracted assemblies. Attributes, such as the names and lengths of individual instances and the references used, are generalized to make the structures comparable, independent of the concrete use case.

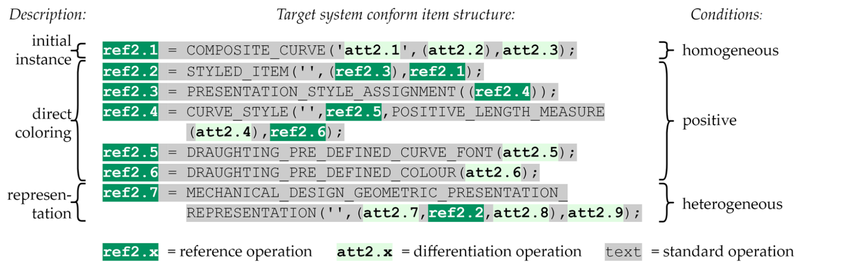


Description:
Source system conform item structure:
Conditions:
initial instance
sketch segment
indirect coloring
represen-tation
homogeneous
neutral
heterogeneous

```
ref1.1 = COMPOSITE_CURVE('att1.1',(att1.2,ref1.2,att1.3),att1.4);
ref1.2 = COMPOSITE_CURVE_SEGMENT(att1.5,att1.6,ref1.3);
ref1.3 = TRIMMED_CURVE('',att1.7,(PARAMETER_VALUE(att1.8)),
         (PARAMETER_VALUE(att1.9)),att1.10,att1.11);
ref1.4 = STYLED_ITEM('',(ref1.5),ref1.3);
ref1.5 = PRESENTATION_STYLE_ASSIGNMENT((ref1.6));
ref1.6 = CURVE_STYLE('',ref1.7,POSITIVE_LENGTH_MEASURE
         (att1.12),ref1.8);
ref1.7 = DRAUGHTING_PRE_DEFINED_CURVE_FONT(att1.13);
ref1.8 = DRAUGHTING_PRE_DEFINED_COLOUR(att1.14);
ref1.9 = MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_
         REPRESENTATION('',(att1.15,ref1.4,att1.16),att1.17);
```

ref1.x = reference operation att1.x = differentiation operation text = standard operation

Figure 8. Source system (Creo) conforms to the item structure of the sketch coloring.



Description:
Target system conform item structure:
Conditions:
initial instance
direct coloring
represen-tation
homogeneous
positive
heterogeneous

```
ref2.1 = COMPOSITE_CURVE('att2.1',(att2.2),att2.3);
ref2.2 = STYLED_ITEM('',(ref2.3),ref2.1);
ref2.3 = PRESENTATION_STYLE_ASSIGNMENT((ref2.4));
ref2.4 = CURVE_STYLE('',ref2.5,POSITIVE_LENGTH_MEASURE
         (att2.4),ref2.6);
ref2.5 = DRAUGHTING_PRE_DEFINED_CURVE_FONT(att2.5);
ref2.6 = DRAUGHTING_PRE_DEFINED_COLOUR(att2.6);
ref2.7 = MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_
         REPRESENTATION('',(att2.7,ref2.2,att2.8),att2.9);
```

ref2.x = reference operation att2.x = differentiation operation text = standard operation

Figure 9. Target system (Catia) conforms to the item structure of the sketch coloring.

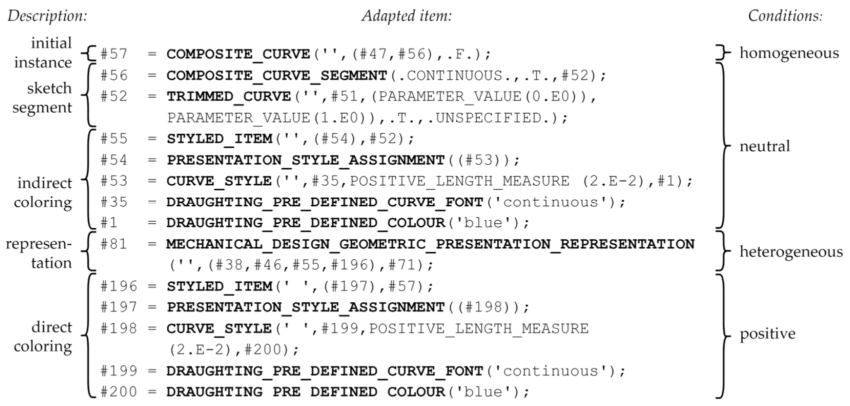
Figure 8 demonstrates that the preprocessor applies indirect coloring in the Creo case. More precisely, the DRAUGHTING_PRE_DEFINED_COLOUR (ref1.8) instance, which contains the color information in att1.14, is linked to a single curve segment via the STYLED_ITEM (ref1.4) instance. The curve segment is then connected to the initial instance, COMPOSITE_CURUVE (ref1.1), by references ref1.2 and ref1.3.

An analysis of the TSI in Figure 9 shows that the postprocessor of Catia expects direct coloring of the initial instance COMPOSITE_CURVE (ref2.1). Accordingly, the entity SYLED_ITEM (ref2.2) directly refers to the initial instance.

In both cases, the entities MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESENTATION (ref1.9 and ref2.7) provide a representation. The instances link to direct and indirect coloring, respectively.

In the second step, defining the relationships between the item structures developed in the previous step is necessary to highlight the differences that cause incompatibility. Figures 8 and 9 show the conditions on the right side. The initial instances COMPOSITE_CURVE (ref1.1 and ref2.1) are identical in both setups, and the data are thus homogeneous. As the postprocessor of Catia expects immediate coloring, which the preprocessor in Creo does not provide, a direct coloring of the initial instance must be newly generated. Due to non-existence, we declare information ref2.2 to ref2.6 as positive. Catia does not require the original indirect coloring for an error-free import, but it does not interfere if it is retained. Thus, instances ref1.2 to ref1.8 are neutral. Both structures contain the entity MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESENTATION (ref1.9 and ref2.7). Furthermore, the exported instance misses a reference to the entity STYLED_ITEM, which we need to generate in the context of immediate coloring. Due to missing information, data ref1.9 and ref2.7 must be classified as heterogeneous.

The third step is the actual adaptation of the item. Figure 10 shows a specific adapted item. In the case of the generalized, newly added attributes att2.4 and att2.5, standard-compliant values from ISO 10303-214 are inserted [50]. For att2.6, the attribute "blue" is inputted, which is already found in the extracted item from Creo. The heterogeneous entity MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESENTATION references the STYLED_ITEM (#196) instance of the newly added direct coloring with the references #196 to #200. The names of the generated data have been newly added. Keeping instances with a neutral condition ensures error-free interpretation in both the source and target systems. Inserting the item from Figure 10 into the previously incompatible exchange file provides the correct coloring of the sketch in Catia and Creo, which then solves compatibility issues.



```
#57  = COMPOSITE_CURVE('',(#47,#56),.F.);
#56  = COMPOSITE_CURVE_SEGMENT(.CONTINUOUS.,.T.,#52);
#52  = TRIMMED_CURVE('',#51,(PARAMETER_VALUE(0.E0)),
       PARAMETER_VALUE(1.E0)),.T.,.UNSPECIFIED.);
#55  = STYLED_ITEM('',(#54),#52);
#54  = PRESENTATION_STYLE_ASSIGNMENT((#53));
#53  = CURVE_STYLE('',#35,POSITIVE_LENGTH_MEASURE (2.E-2),#1);
#35  = DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
#1   = DRAUGHTING_PRE_DEFINED_COLOUR('blue');
#81  = MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESENTATION
       ('',(#38,#46,#55,#196),#71);
#196 = STYLED_ITEM(' ',(#197),#57);
#197 = PRESENTATION_STYLE_ASSIGNMENT((#198));
#198 = CURVE_STYLE(' ',#199,POSITIVE_LENGTH_MEASURE
       (2.E-2),#200);
#199 = DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');
#200 = DRAUGHTING PRE DEFINED COLOUR('blue');
```

Figure 10. Adapted, compatible sketch coloring for an error-free import into Catia.

4.2. Retaining Ring Exchange via INP

The second example is the transfer of a retaining ring model. The freeware program Gmsh 4.8.4 (developed by Christophe Geuzaine and Jean-François Remacle) is widely used for finite element mesh generation. We use it to create a grid structure of the retaining ring [57]. As export settings, we choose the INP format with default settings. Standard linear tetrahedra (element type C3D8) build up the mesh. To perform a linear-static analysis based on the exported mesh, we import the INP file into the freeware program Z88Aurora V5 (Engineering Design and CAD, University of Bayreuth, Bayreuth, Germany). The latter software is utilized in the field of finite element method (FEM) [58]. Again, we have chosen standard import settings. However, loading the model is not possible, even if all relevant information is available in the exchange file. Figure 11 illustrates this process.

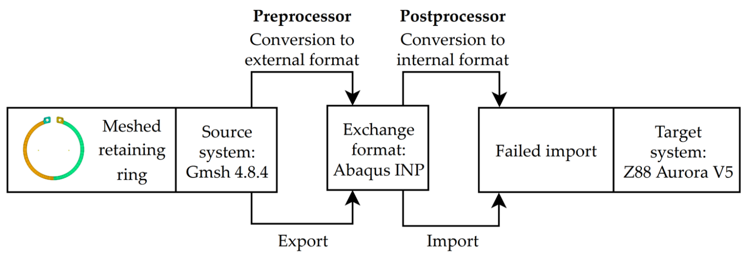
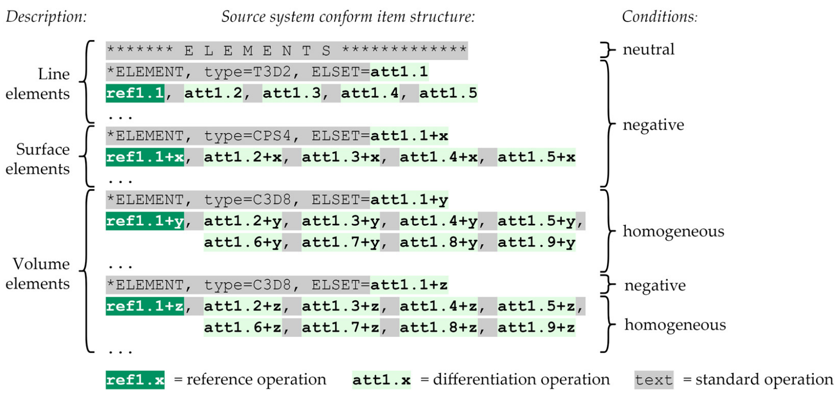

Preprocessor
Conversion to external format
Postprocessor
Conversion to internal format
Meshed retaining ring
Source system: Gmsh 4.8.4
Exchange format: Abaqus INP
Failed import
Target system: Z88 Aurora V5
Export
Import

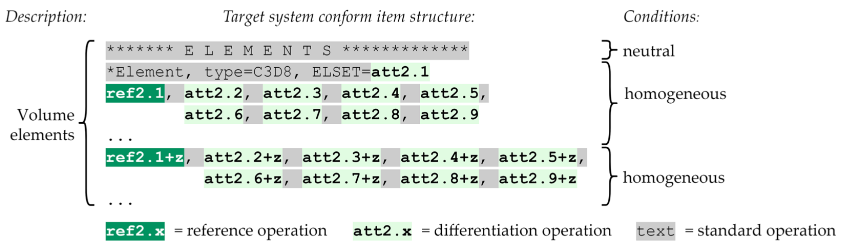
Figure 11. Failed data exchange of a retaining ring model between the finite element mesh generator Gmsh and FEM software Z88.

We trace the exchange error back to faulty element sets. Comparing the SSI from Figure 12 with the TSI from Figure 13, we note that the preprocessor in Gmsh exports unwanted line and surface element blocks (ELSET att1.1 to exclusively att1.1+y) in addition to the relevant volume elements (ELSET form inclusively att1.1+y). However, the postprocessor of Z88 does not expect line and surface elements. It also demands only one volume element set block.


Description:
Source system conform item structure:
Conditions:
Line elements
Surface elements
Volume elements
neutral
negative
homogeneous
negative
homogeneous
refl.x = reference operation
att1.x = differentiation operation
text = standard operation

```
******* E L E M E N T S *************
*ELEMENT, type=T3D2, ELSET=att1.1
ref1.1, att1.2, att1.3, att1.4, att1.5
...
*ELEMENT, type=CPS4, ELSET=att1.1+x
ref1.1+x, att1.2+x, att1.3+x, att1.4+x, att1.5+x
...
*ELEMENT, type=C3D8, ELSET=att1.1+y
ref1.1+y, att1.2+y, att1.3+y, att1.4+y, att1.5+y,
          att1.6+y, att1.7+y, att1.8+y, att1.9+y
...
*ELEMENT, type=C3D8, ELSET=att1.1+z
ref1.1+z, att1.2+z, att1.3+z, att1.4+z, att1.5+z,
          att1.6+z, att1.7+z, att1.8+z, att1.9+z
...
```

Figure 12. Source system (Gmsh) conforms to the item structure of the element set blocks.


Description:
Target system conform item structure:
Conditions:
Volume elements
neutral
homogeneous
homogeneous
ref2.x = reference operation
att2.x = differentiation operation
text = standard operation

```
******* E L E M E N T S *************
*Element, type=C3D8, ELSET=att2.1
ref2.1, att2.2, att2.3, att2.4, att2.5,
        att2.6, att2.7, att2.8, att2.9
...
ref2.1+z, att2.2+z, att2.3+z, att2.4+z, att2.5+z,
          att2.6+z, att2.7+z, att2.8+z, att2.9+z
...
```

Figure 13. Target system (Z88) conforms to the item structure of the element set blocks.

In the second step, we assign conditions based on the structures from the previous step. We classify the additional line and surface element sets as negative because they must be removed to ensure error-free importing of the considered INP file in Z88. Notably, deleting element sets leads to peculiarity. There may be nodes that do not appear in any element set. Unused nodes should be evaluated as negative since they must be removed from the INP file to ensure an error-free data transfer. A merging of the remaining volumes is necessary to ensure compatibility, and we only need to keep the first keyword line of the volume element set blocks. We rank the remaining keyword lines as negative since we must delete them from the file to ensure compatibility. The concrete specification of the node numbers of each element set and their references must be kept, which is why we classify them as homogeneous. Furthermore, we rate the introductory comment line as neutral since the postprocessor ignores the line.

In the third step, we adapt the incompatible items. Figure 14 shows an example of a modified element-set block. All line and area element sets are no longer part of the item, and all remaining volume element sets are merged by removing all keyword lines except the first one. The replacement of the element set block extracted from the original INP file, and replacing it with the adapted block shown in Figure 14 ensures an error-free import. The process allows a structural mechanics analysis of the retaining ring model in Z88.

Description:	*Adapted item:*	*Conditions:*
Comment line	`******* E L E M E N T S *************`	neutral

First element set block	`*Element, type=C3D8, ELSET=Volume1`
	`4797, 1116, 124, 1, 176, 4086, 2002, 303, 1498`
	`4798, 4086, 2002, 303, 1498, 4087, 2005, 305, 1501`
	`...`
		homogeneous

Second element set block	`5709, 751, 45, 1, 124, 4641, 2270, 303, 2002`
	`5710, 4641, 2270, 303, 2002, 4642, 2273, 305, 2005`
	`...`
		homogeneous

Figure 14. Adapted, compatible element sets for an error-free import into Z88.

5. Validation

The presented method was implemented in software within the ViWAT project funded by the European Regional Development Fund (EFRE) [15,59–61]. This research project aims to provide product developers and engineers of small- and medium-sized enterprises (SMEs) with an efficient and easy-to-use software program for the independent analysis, control, and correction of exchange data for partial tool models according to DIN 4000/4003. The peculiarity is related to the models being available in heterogeneous sources, according to DIN 26100 [9]. The project addresses the challenges of item extraction, cross-file information validation, and item adaptation based on the extraction. The implementation of the method demonstrates that it is capable of extracting and adapting items from a wide variety of sources.

For example, Table 3 shows two exchange models for tools according to DIN 4003-87 [62] and DIN 4003-126 [63]. The tools are designed in Creo according to standards. The table displays the tools in CATIA after importing both with and without previous item adaptations. Here, the program can automatically recognize and adapt the cutting part line for several models, which is identified by a blue-colored sketch according to the standard. In addition, Table 4 below shows the import result in Z88 for the meshes of the T-slot cutter, according to DIN 4003-87 [62]. It also displays the described retaining ring with and without item adaption. We meshed the geometries in Gmsh. Overall, these examples show that the presented method is suitable for automated adaptation of various models.

Table 3. Demonstration of the item adaptation using the example of sketch coloring (for clarity, the sketches have been highlighted).

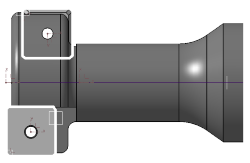
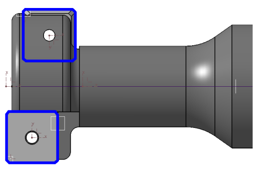
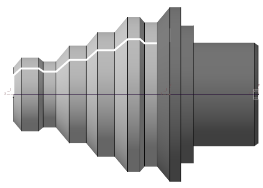
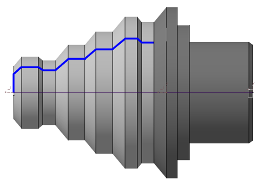
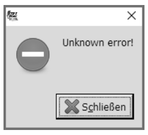
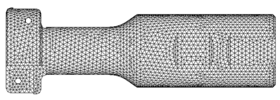
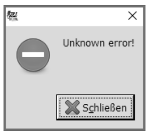
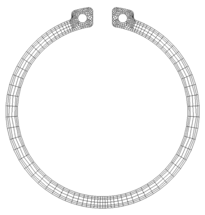
Classification	Model in CATIA without Previous Item Adaption	Model in CATIA with Previous Item Adaption
T-slot cutter according to DIN 4003-87 [62] (excerpt)		
Step reamer according to DIN 4003-126 [63]		

Table 4. Demonstration of item adaption using the example of meshed parts.

Classification	Tool in Z88 without Previous Item Adaption	Tool in Z88 with Previous Item Adaption
T-slot cutter according to DIN 4003-87 [62]	 Unknown error! Schließen	
Retaining Ring according to DIN 471 [64]	 Unknown error! Schließen	

6. Discussion

The item adaptation presented in Section 3 can be classified as a heuristic-based method for extracting and adapting interrelated objects from exchange files. The definition of item structures results in clear rules based on which objects of interest can be obtained. Since compatibility issues are highly specific to each use case, it is beneficial to use distinct heuristics, as they can allow one to define the problem precisely. The stated conditions allow for the description and resolution of even complex compatibility issues. In other words, this method enables the applicant to improve the data integrity of digital twins by modifying partial models that are stored and transmitted in different exchange files. By enhancing the data, it can be kept error-free and complete, ensuring high quality. Another benefit of heuristics-based methods is that they are usually easy to understand and uncomplicated to use for a wide range of items. However, the disadvantages are that expert knowledge is required, and the creation process is time-consuming, especially for complex features. We also note that corresponding logic needs to be defined only once per use case and must also be created for other methods, in addition to using heuristics for new features. Furthermore, approaches from the field of volume decomposition have the disadvantage that they may not converge [32,33]. In addition, the presented method has the advantage that it is also suitable for manipulating data whose structure does not correspond to that of a graph, as is necessary for graph-based methods. If the heuristics are stored, they provide an exact result, which is not the case with neural networks, which are reduced to a certain predictive accuracy. However, an accurate result is necessary to match the unique implementation of the preprocessors. In addition, no conversion takes place, as is the case, for example, with artificial neural networks, which expect a uniform input vector [33,43,44].

Whereas conventional extraction methods apply to partial geometric models stored in files, such as STEP, DXF, and Initial Graphics Exchange Specification (IGES) [29,31–44], the item adaptation applies to other geometry-independent formats and information as well. This is advantageous due to the wide range of applications of digital twins. For example, an increasing amount of product manufacturing information (PMI), such as material specifications, tolerances, notes, and other metadata, is included in models in the course of a model-based definition (MBD) [65,66]. It is also essential to comprehensively extract and validate data that exceeds geometric considerations to ensure continuous data integrity. The explicit definition of item structures allows for the extraction and adaption of various objects from various contexts. However, missing keys and references in hierarchical and block formats makes it challenging to unambiguously identify information. In principle, items can be defined analogously to the presented methodology. To delimit contiguous blocks, formats usually specify delimiters, such as semicolons and introductory and terminating strings, to detect a segment. Combining delimiters and characterizing strings enables unique item identification.

Furthermore, we highlight that external factors can impact how a postprocessor stores an item in an exchange file. External factors characterize the type and number of pieces of information used to represent an item. In other words, the chosen system and its version influence the structure. Moreover, even if we use the same program, the configurations affect the storage in the transfer file. For example, the selected AP can influence the item structure in a STEP file. In addition, studies have shown that design methodology, such as the choice and order of design elements, affects the deposit [17]. Accordingly, different instances represent the same item, even though the export is from the same system. This challenge can be circumvented by providing alternatives to various external factors. Furthermore, different item structures are necessary to reflect external aspects. Once all variants are defined, reliability is assured.

7. Conclusions

Digital twins are increasingly being used in the industry. They are usually built from different partial models, which are then applied depending on the context. Various exchange formats are used to archive partial models and transfer them between multiple systems from different contexts, such as CAD, CAM, and CAE. The data integrity of digital twins is essential. Providing and processing high-quality data still poses a severe problem. Furthermore, exchanging information between different, partly divergent systems is a frequent source of data quality issues, even though all the information required for correct processing is potentially available in the exchange file. This paper presents a method for systematically improving the compatibility of exchange files by adapting stored objects ("items").

It is first necessary to determine the item structure in the source and target systems, as well as to define the structures generally. A comparison of the structures enables the identification of error-causing differences. For this purpose, we suggest five conditions (homogeneous, heterogeneous, positive, negative, and neutral) when performing a comparison. Based on the determined structures and conditions, this paper highlights the adaption of items to ensure faultless processing. After explaining the approach in general, we demonstrate the use of the method based on two diverse use cases from differing domains. The first example is the exchange of a sketch between the source system Creo and the target system Catia using a STEP file in the field of CAD. The second example is the transfer of a meshed retaining ring between the finite element mesh generator Gmsh and the FEM software Z88 via an INP file. As Figure 15 shows, the initially wrongly colored sketch can be correctly interpreted in Catia after applying item adaption. The three steps of the method can also rectify the initially non-functioning data exchange of the retaining ring. However, we want to highlight the following conclusions:

- Various norms, standards, and approaches uniform the exchange of data. Neutral file-based exchange is widely used, for example, in the context of tools, according to DIN 26100. The influences on data exchange and structure are manifold, which can be a source of compatibility problems. One primary influence is the different implementations of the routines in the pre- and postprocessors of the source and target systems involved during data transmission.
- A way to solve these transfer problems is to adapt information internally, directly at the file level, which minimizes the conversion process, as only necessary information is modified. Since compatibility problems are mostly context-dependent, heuristic-based methods are suitable for solving compatibility problems since the distinct issues can be described precisely.
- Specifying heuristics allows us to extract and adapt relevant objects with absolute certainty. A precise procedure is necessary because postprocessors usually require a unique data representation.
- A challenge, however, is that expert knowledge is necessary to specify the required rules, especially for complex items. Nevertheless, item adaptation is especially suitable for standard parts, such as tools, which are characterized by a relatively small number of features but a high number of variants.

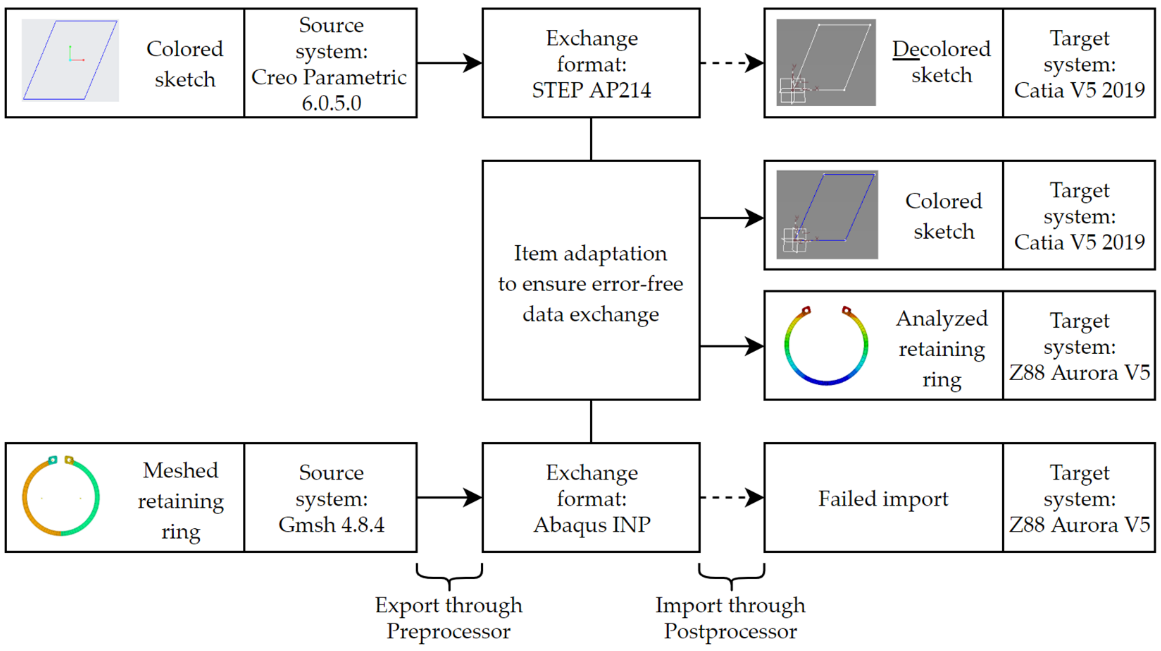

Colored sketch
Source system: Creo Parametric 6.0.5.0
Exchange format: STEP AP214
Decolored sketch
Target system: Catia V5 2019
Item adaptation to ensure error-free data exchange
Colored sketch
Target system: Catia V5 2019
Analyzed retaining ring
Target system: Z88 Aurora V5
Meshed retaining ring
Source system: Gmsh 4.8.4
Exchange format: Abaqus INP
Failed import
Target system: Z88 Aurora V5
Export through Preprocessor
Import through Postprocessor

Figure 15. Compatibility Improvement of partial models of a sketch and a retaining ring using item adaptation.

The shown approach is ideally suited for automation. It is possible to dynamically determine item structures from transfer files and generalize them regarding file-independent processing. However, there is still a significant challenge in detecting general item structures in the first step of item adaptation, especially when the units of interest become more extensive. Obtaining these manually demands an elaborate, time-consuming discovery process. Since the structures are based primarily on standardized formats, it might be possible to use automatic methods, such as machine learning algorithms, to work out relationships. In this course, validation for different use cases is beneficial. For example, further research can vary parameters, such as the item or file size, and verify their effects on processing speed. The automation and validation of the presented method will enable high-quality digital twins in the future. This approach can help move us toward the vision of continuous, all-embracing digitization.

Author Contributions: Conceptualization, J.M.; methodology, J.M. and C.K.; validation, J.M. and C.K.; formal analysis, J.M.; investigation, J.M.; resources, S.T. and F.R.; data curation, J.M.; writing—original draft preparation, J.M.; writing—review and editing, J.M., C.K. and S.T.; visualization, J.M.; supervision, S.T.; project administration, S.T. and F.R.; funding acquisition, F.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Regional Development Found (ERDF) within the ViWAT project. Further development was funded by the European Social Fund (ESF) within the Fit4HPC project (StMBW-W-IX.4-6-210088).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon reasonable request to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study, in the collection, analyses, and interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

References

1. Liu, G.; Shah, R.; Schroeder, R.G. The relationships among functional integration, mass customisation, and firm performance. *Int. J. Prod. Res.* **2012**, *50*, 677–690. [CrossRef]
2. Marks, P.; Yu, Q.; Weyrich, M. Survey on Flexibility and Changeability Indicators of automated Manufacturing Systems. In Proceedings of the 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin, Italy, 4–7 September 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 516–523, ISBN 9781538671092.
3. Anderl, R.; Haag, S.; Schützer, K.; Zancul, E. Digital twin technology—An approach for Industrie 4.0 vertical and horizontal lifecycle integration. *Inf. Technol.* **2018**, *60*, 125–132. [CrossRef]
4. Gartner. Gartner 2018 Hype Cycle for IT in GCC Identifies Six Technologies That Will Reach Mainstream Adoption in Five to 10 Years. Available online: https://www.gartner.com/en/newsroom/press-releases/2018-12-13-gartner-2018-hype-cycle-for-it-in-gcc-identifies-six-technologies-that-will-reach-mainstream-adoption-in-five-to-10-years (accessed on 9 May 2022).
5. Wilking, F.; Schleich, B.; Wartzack, S. Digital Twins—Definitions, Classes and Business Scenarios for Different Industry Sectors. *Proc. Des. Soc.* **2021**, *1*, 1293–1302. [CrossRef]
6. VanDerHorn, E.; Mahadevan, S. Digital Twin: Generalization, characterization and implementation. *Decis. Support Syst.* **2021**, *145*, 113524. [CrossRef]
7. Schleich, B.; Anwer, N.; Mathieu, L.; Wartzack, S. Shaping the digital twin for design and production engineering. *CIRP Ann.* **2017**, *66*, 141–144. [CrossRef]
8. Schroeder, G.N.; Steinmetz, C.; Pereira, C.E.; Espindola, D.B. Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange. *IFAC-PapersOnLine* **2016**, *49*, 12–17. [CrossRef]
9. DIN German Institute for Standardization. *DIN 26100:2021-05: Container File—Summary of Different Product Files for the Data Exchange*; Beuth: Berlin, Germany, 2021.
10. Tao, F.; Sui, F.; Liu, A.; Qi, Q.; Zhang, M.; Song, B.; Guo, Z.; Lu, S.C.-Y.; Nee, A.Y.C. Digital twin-driven product design framework. *Int. J. Prod. Res.* **2019**, *57*, 3935–3953. [CrossRef]
11. Lee, J.; Lapira, E.; Bagheri, B.; Kao, H. Recent advances and trends in predictive manufacturing systems in big data environment. *Manuf. Lett.* **2013**, *1*, 38–41. [CrossRef]
12. Gabor, T.; Belzner, L.; Kiermeier, M.; Beck, M.T.; Neitz, A. A Simulation-Based Architecture for Smart Cyber-Physical Systems. In Proceedings of the 2016 IEEE International Conference on Autonomic Computing (ICAC), Wuerzburg, Germany, 17–22 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 374–379, ISBN 978-1-5090-1654-9.
13. Courtney, R. Some Informal Comments About Integrity and the Integrity Workshop. In Proceedings of the International Workshop on Data Integrity, Gakhersburg, MD, USA, 25–27 January 1989; Ruthberg, Z.G., Polk, W.T., Eds.; National Institute of Standards and Technology: Gaithersburg, MD, USA, 1989; pp. 1–18.
14. Sandhu, R.S. On five definitions of data integrity. In *DBSec*; Citeseer: University Park, PA, USA, 1994; pp. 257–267.
15. Kleinschrodt, C. Analyse und Optimierung des Datenaustauschs von 3D-Modellen: Übertragung von CAD-Werkzeugmodellen mittels STEP. Ph.D. Thesis, University of Bayreuth, Bayreuth, Germany, 2019.
16. ISO International Organization for Standardization. *ISO 9000:2015: Quality Management Systems—Fundamentals and Vocabulary*; Beuth Verlag GmbH: Berlin, Germany, 2015.
17. Kleinschrodt, C.; Rieg, F. Einfluss der Modellierung auf die Austauschqualität von CAD-Modellen. *Konstruktion* **2020**, *72*. [CrossRef]
18. Autodesk. DXF Reference. Available online: http://images.autodesk.com/adsk/files/autocad_2012_pdf_dxf-reference_enu.pdf (accessed on 9 August 2022).
19. Abaqus. Abaqus Model Definition. Available online: https://abaqus-docs.mit.edu/2017/English/SIMACAEMODRefMap/simamod-c-model.htm (accessed on 5 May 2022).
20. ISO International Organization for Standardization. *Industrial Automation Systems and Integration—Product Data Representation and Exchange (ISO 10303): Part 1: Overview and Fundamental Principles*; ISO: Geneva, Switzerland, 2021.
21. Protégé. A free, Open-Souce Ontology Editor and Framework for Building Intelligent Systems. Available online: https://protege.stanford.edu/ (accessed on 21 July 2022).
22. McGuinnes, D.L.; van Harmelen, F. OWL web ontology language overview. *W3C* **2004**, *10*, 2004.
23. Barbau, R.; Krima, S.; Rachuri, S.; Narayanan, A.; Fiorentini, X.; Foufou, S.; Sriram, R.D. OntoSTEP: Enriching product model data using ontologies. *Comput. Aided Des.* **2012**, *44*, 575–590. [CrossRef]
24. Chaparala, R.T.; Hartman, N.W.; Springer, J. Examining CAD Interoperability through the Use of Ontologies. *Comput. Aided Des. Appl.* **2013**, *10*, 83–96. [CrossRef]
25. Qin, Y.; Lu, W.; Qi, Q.; Liu, X.; Zhong, Y.; Scott, P.J.; Jiang, X. Status, Comparison, and Issues of Computer-Aided Design Model Data Exchange Methods Based on Standardized Neutral Files and Web Ontology Language File. *J. Comput. Inf. Sci. Eng.* **2017**, *17*, 010801-1–010801-8. [CrossRef]
26. Ramnath, S.; Haghighi, P.; Venkiteswaran, A.; Shah, J.J. Interoperability of CAD geometry and product manufacturing information for computer integrated manufacturing. *Int. J. Comput. Integr. Manuf.* **2020**, *33*, 116–132. [CrossRef]
27. Li, H.; Lu, J.; Zheng, X.; Guoxin, W.; Dimitris, K. Supporting Digital Twin Integration Using Semantic Modeling and High-Level Architecture. In *Advances in Production Management Systems, Artificial Intelligence for Sustainable and Resilient Production Systems*; Dolgui, A., Bernard, A., Lemoine, D., von Cieminski, G., Romero, D., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 228–236. ISBN 978-3-030-85909-1.

28. Klein, M.; Maschler, B.; Zeller, A.; Tallkhestani, A.; Jazdi, N.; Rosen, R.; Weyrich, M. Architektur und Technologiekomponenten eines digitalen Zwillings. In Proceedings of the VDI-Kongress Automation, Baden-Baden, Germany, 2–3 July 2019; pp. 89–102.
29. Nasr, E.A.; Kamrani, A.K. *Computer-Based Design and Manufacturing: An Information-Based Approach*; Springer: New York, NY, USA, 2007; ISBN 0-387-23323-7.
30. Rao, P.N. *CAD/CAM—Principles and Applications*, 2nd ed.; Tata McGraw-Hill: New Delhi, India, 2004; ISBN 0-07-058373-0.
31. Babic, B.; Nesic, N.; Miljkovic, Z. A review of automated feature recognition with rule-based pattern recognition. *Comput. Ind.* **2008**, *59*, 321–337. [CrossRef]
32. Shi, Y.; Zhang, Y.; Xia, K.; Harik, R. A Critical Review of Feature Recognition Techniques. *Comput. Aided Des. Appl.* **2020**, *17*, 861–899. [CrossRef]
33. Yeo, C.; Kim, B.C.; Cheon, S.; Lee, J.; Mun, D. Machining feature recognition based on deep neural networks to support tight integration with 3D CAD systems. *Sci. Rep.* **2021**, *11*, 22147. [CrossRef]
34. Shah, J.J.; Anderson, D.; Kim, Y.S.; Joshi, S. A Discourse on Geometric Feature Recognition from CAD Models. *J. Comput. Inf. Sci. Eng.* **2001**, *1*, 41–51. [CrossRef]
35. Joshi, S.; Chang, T.C. Graph-based heuristics for recognition of machined features from a 3D solid model. *Comput. Aided Des.* **1988**, *20*, 58–66. [CrossRef]
36. Marefat, M.; Kashyap, R.L. Geometric reasoning for recognition of three-dimensional object features. *IEEE Trans. Pattern Anal. Mach. Intell.* **1990**, *12*, 949–965. [CrossRef]
37. Holland, P.; Standring, P.; Long, H.; Mynors, D. Feature extraction from STEP (ISO 10303) CAD drawing files for metalforming process selection in an integrated design system. *J. Mater. Process. Technol.* **2002**, *125-126*, 446–455. [CrossRef]
38. Tan, C.F.; Kher, V.K.; Ismail, N. Design of a Feature Recognition System for CAD/CAM Integration. *World Appl. Sci. J.* **2013**, *21*, 1162–1166. [CrossRef]
39. Malleswaria, V.N.; Vallib, P.M. Automatic Recognition of Machining Features using STEP Files. *Int. J. Eng. Res. Technol.* **2013**, *2*, 1–11.
40. Vandenbrande, J.H.; Requicha, A. Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Trans. Pattern Anal. Mach. Intell.* **1993**, *15*, 1269–1285. [CrossRef]
41. Regli, W.C.; Gupta, S.K.; Nau, D.S. Extracting alternative machining features: An algorithmic approach. *Res. Eng. Des.* **1995**, *7*, 173–192. [CrossRef]
42. Sivakumar, S.; Dhanalakshmi, V. An approach towards the integration of CAD/CAM/CAI through STEP file using feature extraction for cylindrical parts. *Int. J. Comput. Integr. Manuf.* **2013**, *26*, 561–570. [CrossRef]
43. Babić, B.R.; Nešić, N.; Miljković, Z. Automatic feature recognition using artificial neural networks to integrate design and manufacturing: Review of automatic feature recognition systems. *Artif. Intell. Eng. Des. Anal. Manuf.* **2011**, *25*, 289–304. [CrossRef]
44. Zhang, D.; He, F.; Tu, Z.; Zou, L.; Chen, Y. Pointwise geometric and semantic learning network on 3D point clouds. *ICA* **2019**, *27*, 57–75. [CrossRef]
45. Wildgrube, E. Datenformat. In *Lexikon der Informatik und Datenverarbeitung*, 3rd ed.; Schneider, H.-J., Ed.; Oldenbourg: München, Germany, 1991; pp. 187–188. ISBN 3-486-21514-0.
46. Klein, J. *Datenintegrität in Heterogenen Informationssystemen: Ereignisorientierte Aktualisierung globaler Datenredundanzen*, 1st ed.; Deutscher Universitätsverlag: Wiesbaden, Germany, 1992; ISBN 978-3824401079.
47. Heidenhain. User's Manual Conversational Programming. Available online: https://content.heidenhain.de/doku/tnc_guide/pdf_files/TNC400/286180-xx/bhb/291_016-24.pdf (accessed on 25 January 2022).
48. DIN German Institute for Standardization. *DIN 4000-102:2021-05: Tabular Layouts of Properties—Part 102: Data Exchange for Tabular Layouts of Properties with XML Schema*; Beuth: Berlin, Germany, 2021.
49. Anderl, R.; Trippner, D. *STEP STandard for the Exchange of Product Model Data: Eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303 (STEP)*, 1st ed.; Teubner: Stuttgart, Germany; Leipzig, Germany, 2000; ISBN 978-3-519-06377-3.
50. ISO International Organization for Standardization. *Industrial Automation Systems and Integration—Product Data Representation and Exchange (ISO 10303): Part 214: Application Protocol: Core Data for Automotive Mechanical Design Processes*; ISO: Geneva, Schwitzerland, 2010.
51. ISO International Organization for Standardization. *Industrial Automation Systems and Integration—Product Data Representation and Exchange (ISO 10303): Part 21: Implementation Methods: Clear Text Encoding of the Exchange Structure*; ISO: Geneva, Schwitzerland, 2002.
52. Abaqus. Characterizing Elements. Available online: https://abaqus-docs.mit.edu/2017/English/SIMACAEELMRefMap/simaelm-c-general.htm (accessed on 5 May 2022).
53. Abaqus. Input Syntax Rules. Available online: https://abaqus-docs.mit.edu/2017/English/SIMACAEMODRefMap/simamod-c-inputsyntax.htm (accessed on 4 May 2022).
54. Kleinschrodt, C.; Rieg, F. Normalisation of STEP files for improving the data compatibility of transferred tool models. *Tech. Gaz.* **2017**, *5*, 201–205. [CrossRef]
55. Nagy, Z. *Regex Quick Syntax Reference: Understanding and Using*; Apress: Berkeley, CA, USA, 2018; ISBN 978-1-4842-3875-2.
56. DIN German Institute for Standardization. *DIN 4000-81:2017-09: Tabular Layouts of Properties—Part 81: Drills and Countersinking Tools with Non-Indexable Cutting Edges*; Beuth: Berlin, Germany, 2017.

57. Gmsh. A Three-Dimensional Finite Element Mesh Generator with Built-In Pre- and Post-Processing Facilities. Available online: https://gmsh.info/ (accessed on 4 May 2022).
58. LSCAD. Z88AURORA. Available online: https://en.z88.de/z88aurora/ (accessed on 6 May 2022).
59. Mohr, J.; Kleinschrodt, C.; Diwisch, P.; Rieg, F. Betrachtung von Konfigurationsdateiformaten und GUI-Frameworks für Programme zur Aufbereitung von Austauschdateien. In Proceedings of the 18th Gemeinsames Kolloquium Konstruktionstechnik: Nachhaltige Produktentwicklung, Duisburg, Germany, 1–2 October 2020; Corves, B., Gericke, K., Grote, K.-H., Lohrengel, A., Löwer, M., Nagarajah, A., Rieg, F., Scharr, G., Stelzer, R., Eds.; University of Duisburg-Essen: Duisburg, Germany, 1718. [CrossRef]
60. Mohr, J.; Kleinschrodt, C.; Siegel, T.; Rieg, F. Entwicklung einer Beschreibungssprache zur Analyse und Behebung von Datenaustauschproblemen. In Proceedings of the 17th Gemeinsames Kolloquium Konstruktionstechnik: Agile Entwicklung physischer Produkte, Aachen, Germany, 1–2 October 2019; Corves, B., Gericke, K., Grote, K.-H., Lohrengel, A., Müller, N., Nagarajah, A., Rieg, F., Scharr, G., Stelzer, R., Eds.; RWTH Aachen: Aachen, Germany. [CrossRef]
61. Mohr, J.; Kleinschrodt, C.; Zimmermann, M.; Rieg, F. Konzeptionelles Design zur softwaregestützten Analyse und Modifikation von Produktdaten. In Proceedings of the 16th Gemeinsames Kolloquium Konstruktionstechnik: Digitalisierung und Produktentwicklung, Bayreuth, Germany, 11–12 October 2018; Brökel, K., Corves, B., Grote, K.-H., Lohrengel, A., Müller, N., Nagarajah, A., Rieg, F., Scharr, G., Stelzer, R., Eds.; University of Bayreuth: Bayreuth, Germany, 2018. ISBN 978-3-00-059609-4.
62. DIN German Institute for Standardization. *DIN 4003-87:2021-10: Concept for the Design of 3D Models Based on Properties According to DIN 4000—Part 87: End Mills for Indexable Inserts*; Beuth: Berlin, Germany, 2021.
63. DIN German Institute for Standardization. *DIN 4003-126:2012-10: Concept for the Design of 3D Models Based on Properties According to DIN 4000—Part 126: Reamers with Non-Indexable Cutting Edges*; Beuth: Berlin, Germany, 2012.
64. DIN German Institute for Standardization. *DIN 471: Retaining Rings for Shafts—Normal Type and Heavy Type*; Beuth: Berlin, Germany, 2011.
65. Martin, D. What Does MBD Mean? Available online: https://www.ptc.com/en/blogs/cad/what-does-mbd-mean (accessed on 18 April 2022).
66. Miller, A.M.; Alvarez, R.; Hartman, N. Towards an extended model-based definition for the digital twin. *Comput. Aided Des. Appl.* **2018**, *15*, 880–891. [CrossRef]