Doctoral theses at NTNU, 2020:416

Remi Krister Sylvain Lanza

Improving and implementing the STEP ISO 10303 standard for design, analysis and structural test data correlation

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of Science and Technology

Remi Krister Sylvain Lanza

Improving and implementing the STEP ISO 10303 standard for design, analysis and structural test data correlation

Thesis for the Degree of Philosophiae Doctor

Trondheim, December 2020

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Engineering
Department of Mechanical and Industrial Engineering

ISBN 978-82-471-9399-0 (printed ver.)
ISBN 978-82-471-9676-2 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2020:416

Printed by NTNU Grafisk senter

Table of Contents

List of Figures

List of Abbreviations

AM	Application Module
AP	Application Protocol
API	Application Programming Interface
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CFD	Computational Fluid Dynamics
CNC	Computer Numerical Control
DAQ	Data Acquisition system
EDM	Express Data Manager
EDMS	Express Data Manager Supervisor
ESA	European Space Agency
FEA	Finite Element Analysis
FEM	Finite Element Method
HDF5	Hierarchical Data Format version 5
IAR	Integrated Application Resources
IGR	Integrated Generic Resources

ILS	Integrated Logistics Support
IoT	Internet of Things
ISO	International Organization for Standardization
NFR	The Research Council of Norway
PDM	Product Data Management
PLM	Product Lifecycle Management
SDAI	Standard Data Access Interface
SDM	Simulation Data Management
STEP	Standard for the Exchange of Product model data
XML	Extensible Markup Language

Preface

This study was realized as an industrial Ph.D. scheme; a collaboration between NTNU (Norwegian University of Science and Technology) and Jotne EPM Technology AS, and partly funded by NFR (The Research Council of Norway). Terje Rølvåg as the main supervisor from NTNU, and Jochen Haenisch as the main supervisor from Jotne. The work was performed over the course of 3.5 years; one year at the Department of Mechanical and Industrial Engineering at NTNU, and the remaining at Jotne's offices.

Three academic papers were written during the study; one published, one under review, and one submitted. These are included in an appendix of this document.

The thesis consists of an introduction, relevant background information for the attached papers, including details on how this work was used and implemented in Jotne's projects and software.

Acknowledgements

First of all I would like to give a huge thanks to Jotne, which gave me the opportunity for both a position in their company, and a Ph.D. education through NTNU. I also appreciate they let me be involved in projects involving interesting companies such as Lockheed Martin Aeronautics, ESA (European Space Agency), Boeing, and others. They gave me a chance to combine both my background education of Mechanical Engineering, and IT which was until that time only a hobby.

A very special thanks to my supervisor Jochen Haenisch, who has lead me through all my work over the last few years. His shared knowledge and advice has been crucial for completing this thesis, and I couldn't have done it without his support. His talent of performing very thoroughly and detailed checks of my papers and thesis has been very valuable!

A big thanks to my supervisor at NTNU, Terje Rølvåg, even though not being in the same location as me, Terje has always been easy to reach and quick to help. He has been very helpful in all the mechanical engineering side of this work, and has always had a good answer when I was stressed about which direction to go in this study.

From Jotne, I also want to thank Kjell Bengsston, aka. my boss. Kjell has also given me a lot of guidance, pushed me to go to conferences, and been very efficient to get people's attention towards my work. My co-worker, Olav Liestøl, deserves a special thanks for all his help and support when it comes to programming. The uncountable hours of debugging wouldn't be the same without his help!

I'd also like to thank everyone else at Jotne for helping me in different situations, and I look forward working with them for more years to come.

At the beginning of this study, a complicated mechanical structure had to be built. This was more complicated than expected, and I want to give a big thanks to the crew at the workshop, especially Børge and Carl-Magnus, for bringing it to completion.

Finally, and most importantly, my wife, Jahzel, to whom I got married in the middle of my Ph.D. studies, deserves all my gratitude for her encouragement and motivational support during my studies. My parents, Jean-Michel and Ingrid, and my sisters, Solveig and Karen, deserve of course a personal thanks, for their encouragement, and for making me think I have control over what I'm working on!

Abstract

The objective of this thesis was to improve the effectiveness of data management of physical test and simulation data, in the context of digital twins, PDM (Product Data Management)/PLM (Product Lifecycle Management) and SDM (Simulation Data Management).

Many different software applications, from different vendors are used for simulation, CAD (Computer Aided Design), PDM/PLM and other engineering activities. Most of them use different data storage formats and methods that are customized for the specific use of their application.

How can the lack of data interoperability among collaborating engineering applications be solved? Within and across companies sharing of data, for example, for creating and maintaining digital twins, becomes cumbersome when different parties use different applications. Files need to be converted, or tasks need to be re-done, potentially leading to loss of information. Managing project data becomes difficult, and with data originating from different sources, stored in different locations and companies, it is tough to keep track of what is where and in which version. This is especially important for digital twins, where different domain data need to be accessed by automatic processes.

This thesis addresses this problem specifically for the domains of FEM (Finite Element Method) and structural testing data. However, the fact that this is part of a larger context involving multiple other domains, is taken in consideration.

The STEP ISO 10303 standard is highly in focus throughout the study. This standard contains data models designed to cover as much as possible of the different engineering domains, across development life cycle stages.

The study shows how this standard can also be applied to represent and manage structural test data, including its relations to corresponding FEM analyses; this has never been done before. Implementations using and validating the new concepts were performed in converter applications and in a SDM tool.

To increase the FEM domain coverage of the standard, certain extensions of the standard are recommended after having been implemented and validated as part of this thesis. With these extensions the standard can be used for nonlinear FEM analysis, thus, allowing also such advanced analysis data to be shared among FEM solvers.

Chapter 1

Introduction

1.1 Industrial Context

Structural testing, analysis and their correlation has always been important when designing complex products and systems. These fields are tightly linked to the idea of a digital twin. The concept of digital twin, a virtual model of a product or process, has become increasingly important in many industries [1, 2]. A digital twin should collect all relevant information of a certain physical product or process, throughout its lifetime. Depending on the use-case, this would include data from many different domains. There are many technical aspects of a digital twin, such as, connectivity (IoT; *Internet of Things*), sensors, analytics, accessibility, data correlation, etc. This study focuses on the data representation aspect. Digital twin data need to be collected in a consistent product data model to enable the harvesting of the digital twin vision.

There exist multiple implementations of digital twins in different contexts, but as a relatively newly emerging topic, there are no standards for managing all the data that is involved. (The importance of standards, and the implications of the absence of standards, in the context of digital twins are discussed in [1, 3, 4, 5].)

In engineering industries, data management systems are used to keep a consistent and comprehensive overview of all data related to the development, operation, optimization loops and data tracing of products. These systems collect data and references to data, in a repository (in one location or distributed) to facilitate data management. Applications that handle this are known as PDM (Product Data Management) and PLM (Product Lifecycle Management) applications [6]; however, as they are not standard based, they do not give the industry full control over their own data.

Within engineering data management, there is also Simulation Data Management (SDM)

[7]. While PDM is data management on a higher level, SDM applications handle the organization of simulation data at a more detailed level [8]. These systems also lack standards compliance, and users are dependent on the specific tools supported by the SDM application they use.

In the context of digital twins, PDM, PLM and SDM, interoperability between data originating from different applications, from different domains, and across the product's entire lifecycle, is necessary to establish a consistent product data model.

Currently, data management applications, used by digital twins or not, handle interoperability by either (or both):

1. Holding references between related data on a file level, for example; a simulation file is related to a CAD file, or the documentation of a sensor is related to the results file of a physical test.

2. Holding references between related data on a data object level, for example; an object representing a sensor relates to its test result values, and a load case simulating the test.

The second option though, being more attractive, locks the user to the CAE (Computer Aided Engineering) applications provided by the vendor of their SDM system, and thus interoperability between applications is very limited.

The STEP ISO 10303 [9] standard was created as an interoperable (and common) data model across many engineering domains. It allows to relate engineering data cross-domain, and is thus very useful not only for domain specific applications, but also for PLM [10, 11, 12, 13, 14], SDM [15, 16], and digital twin models that span many domains.

The data scope of this thesis applies to the digital twins, for example, for predictive maintenance where measured data are compared to existing analysis and design data. The digital twin may derive, from analyzing the comprehensive and integrated data sets, the need to deviate from planned maintenance procedures. This thesis validates the completeness of STEP ISO 10303 AP209 [17] data sets for such queries. Other technical aspects of digital twin use cases, such as, data sample rate, data filtering and analytics, algorithms, frequency of analyses and comparisons, etc. are independent of this validation of the data availability aspect. The detailed recommendations of this thesis for AP209 are applicable to a few digital twin use cases; methodology and general conclusions concerning the use of STEP, however, should also be considered for a wider range of scenarios.

1.2 Objective

This thesis is a contribution to a critical review of the suitability of STEP ISO 10303 to capture all engineering information related to structural testing, analysis and their association. Within this wide field, the goal is to establish and validate methods to improve the effectiveness of simulation and test data management.

The focus of this thesis is mainly on the standard representation of:

- Structural test data, i.e. sensor information, sensor data, and test result data
- FEM (Finite Element Analysis) data
- The relation between the above two items
- All of the items above in the context of SDM applications

The objectives of this thesis in the context of data management, long term archiving, data interoperability and data traceability are as follows:

O.1 Validate that the ISO STEP 10303 standard may be used for storing, sharing, managing, and correlating design, simulation, and structural test data.

O.2 Identify the obstacles in using ISO STEP 10303 for data management and data exchange of FEM data.

These objectives are achieved by developing STEP based interoperability solutions among several commercial FEM and testing applications, and by integrating such data in a STEP compliant repository.

1.3 Structure of Thesis

Chapter 1 introduces the background and the goals of this study. Since the STEP ISO 10303 standard holds such central part of this thesis, a complete chapter, Chapter 2, is reserved for introducing and presenting its structure, architecture, and application methods. Chapter 3 presents some of the applications and projects which have been developed during the study or involved in the study. Summaries of the author's published and submitted academic articles, are discussed in Chapter 4. Finally we conclude the study and propose future work in Chapter 5.

The authors articles are included in Appendix A and B:

1. A.1 Relating Structural Test and FEA data with STEP AP209 [18]

- Published in *Advances in Engineering Software* (Main author)

2. A.2 ISO 10303 AP209 - Why and how to embed nonlinear FEA [19]

   - Under review with *Advances in Engineering Software* (Main author)

3. A.3 Extending STEP AP209 for Nonlinear Finite Element Analysis [20]

   - Under review with *Advances in Engineering Software* (Main author)

4. B.1 Open Simulation Data Management and Testing - The CRYSTAL Project [21]

   - Published in *NAFEMS World Congress 2017* (Co-author)

An additional Appendix C presents a study which was originally intended as an article, but due to limitations is here added as an appendix.

Chapter 2

ISO 10303 STEP

2.1 The STEP Standard and Industrial Use

ISO 10303, officially called *Industrial automation systems and integration - Product data representation and exchange*, and commonly known as STEP (*STandard for the Exchange of Product model data*), is an ISO standard defining data models for the representation of product, product development, and product usage information. The standard covers multiple engineering domains, including, but not limited to; PLM, PDM, CAD (Computer Aided Design), FEA (Finite Element Analysis), and CFD (Computational Fluid Dynamics).

Most applications use proprietary formats for their data storage. The problems with such formats are, 1) exchanging data between different systems is not always possible, and 2) systems may change their storage formats when introducing new versions.

To be freed from proprietary storage formats, the standard provides data models for all relevant engineering domains. The purpose of the standard is to enable a more seamless data integration for applications, both within the same, and across different domains. Throughout product development steps, multiple applications are used to perform activities both within the same, and across all steps. A lot of information may overlap across these activities, and without a common and central data model, consistency becomes difficult.

Although each domain covered by STEP have their own data models, as will be described in more details in section 2.2, all models share a common *sub*-data model, thus enabling interaoperability between them. Figure 2.1 shows the concept of overlapping domain data, were PLM is part of each domain.

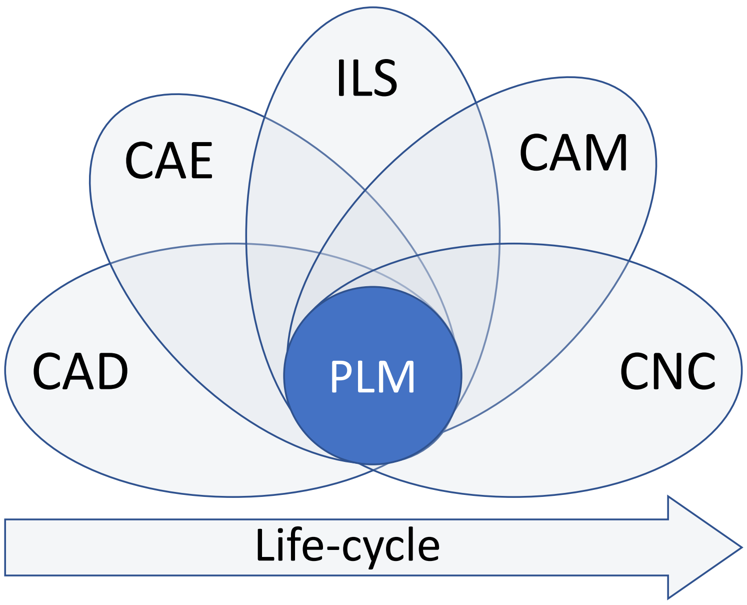


Figure 2.1: Multiple domains have certain overlap. A very central overlap is the PLM information.

The data models are written in the EXPRESS [22] language, discussed in 2.2.1, and can be mapped to any proprietary system that wishes to be STEP compliant.

2.2 Architecture of the STEP Standard

ISO 10303 is a collection of hundreds of documents each describing and defining different parts of the standard. The documents are divided in different categories. The main categories are the following;

- Description Methods
- Implementation Methods
- Integrated Generic Resources (IGR)
- Integrated Application Resources (IAR)
- Application Modules (AM)
- Application Protocols (AP)

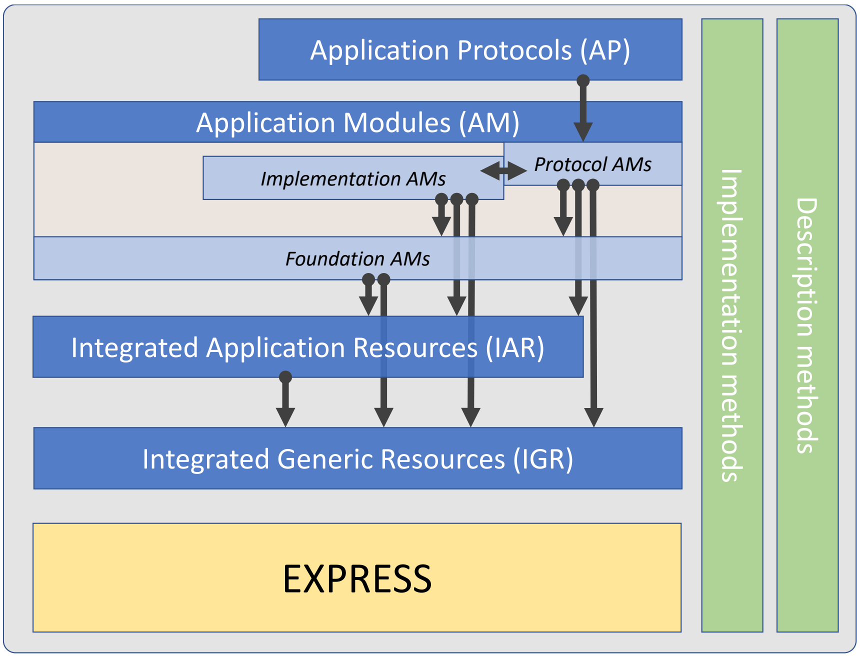


Figure 2.2: STEP architecture and modules. Each category boxes contains ISO documents. The arrows show between which categories there are allowable references.

2.2.1 Description and Implementation Methods

STEP EXPRESS

STEP represents the data models for these domains in the computer-interpretable language EXPRESS, which is itself documented by the standard.

The main component of EXPRESS data models are *entities*. Entities represent objects with properties and rules, and can be compared to classes in object-oriented languages. The properties, or attributes, may be classical data types such as; integers, reals and booleans, specialized data types defined by the data model, or other entities. As with classes, entities may inherit properties from other entities.

An example of an EXPRESS entity, **volume_3d_element_representation**, describing a FEA volume element, and its inherited parents, **representation** and **element_representation**, is shown in Listing 2.1. (In this document, EXPRESS entities are shown in **bold**.)

```

ENTITY representation
  SUPERTYPE OF ( left out for simplicity  ) ;
    name                : label;
    items               : SET[1:?] OF representation_item;
    context_of_items : representation_context;
  DERIVE
    id                  : identifier := get_id_value (SELF);
    description         : text := get_description_value (SELF);
  WHERE
    WR1: left out for simplicity;
    WR2: left out for simplicity;
END_ENTITY;

ENTITY element_representation
  SUPERTYPE OF ( left out for simplicity )
  SUBTYPE OF (representation);
    node_list                      : LIST [1:?] OF node_representation;
  WHERE
    WR1: SIZEOF (QUERY(item <* node_list |
      'AP209_MULTIDISCIPLINARY_ANALYSIS_AND_DESIGN_MIM_LF.' +
      'GEOMETRIC_NODE' IN TYPEOF (item))) = 0;
END_ENTITY;

ENTITY volume_3d_element_representation
  SUBTYPE OF (element_representation);
    model_ref                      : fea_model_3d;
    element_descriptor             : volume_3d_element_descriptor;
    material                       : element_material;
  UNIQUE
    UR1: model_ref, SELF\representation.name;
  WHERE
    WR1: left out for simplicity;
    WR2: left out for simplicity;
    WR3: left out for simplicity;
    FU1:  required_3d_nodes (
      SELF\element_representation.node_list,
      element_descriptor.shape,
      element_descriptor\element_descriptor.topology_order);
END_ENTITY;
```

Listing 2.1: EXPRESS model of **representation**, **element_representation** and **volume_3d_element_representation**

The terms *SUBTYPE* and *SUPERTYPE* in Listing 2.1 refers to parent-child relationships, and are used to specify if an entity inherits from, or is the parent of a specific entity. In the snippet we also see how a volume element has attributes such as element material, element descriptor, FEA model, and list of nodes (inherited from its supertype **element_representation**). These attributes are other entities (or containers of entities) in the data model. The *WRx* and *FUx*, are so called *domain rules* or *WHERE rules*. These set restrictions on what is a valid occurrence of the entity. The specifications for such rules may be complex, and most are therefore left out from the example. *FU1*, in **volume_3d_**

element_representation, checks, with the use of the function *required_3d_nodes*, that the correct number of nodes are included in the *node_list* attribute. The required number of nodes is dependent on the shape and topological order of the element, and are therefore arguments to the function.

The rules, specified in EXPRESS, can be interpreted by applications and be used for automated validations of populations of the data model.

Storage forms of STEP

The data models describe the structure and semantics of the data. A population of this data can be stored in different ways:

- As an ASCII file following the STEP format defined in ISO 10303-21 [23].
- As a binary file using the HDF5 (Hierarchical Data Format Version 5) format [24]. ISO 10303-26 [25] defines the mapping of EXPRESS structures to HDF5 structures of STEP data.
- As an XML file following the specification in ISO 10303-28 [26].
- As a relational database by using the EXPRESS data model as a database schema.

Listing 2.2 shows a snippet of the ISO 10303-21 representation of a data population. It contains three FEA volume elements and some of its referenced instances. Each instance of an entity is identified by an identifier starting with # and is used when other instances reference it through attributes. For example `#1136` is an instance of the entity **volume_3d_element_representation**. Its first attribute, *name*, inherited from the entity **representation**, is defined as a *label* (which is defined as a string), is `'1'`. The subsequent attributes, inherited from **element_representation**, and defined in **volume_3d_element_representation**, are references to other instances. The attribute *element_descriptor*, references a **volume_3d_element_descriptor** with id `#1147`. This entity specifies the shape (hexahedron) and order (linear) of the element. This specification is used by the rule *FU1* under a validation of the model, to check that the required number of nodes is set.

```

#73= CARTESIAN_POINT('1',(0.,-4.,0.));
#75=  NODE('1',(#73),#28,#62);
#84=  NODE('2',(#82),#28,#62);
#104= NODE('7',(#102),#28,#62);

#62= FEA_MODEL_3D('Nastran job EAS test case ATS4m5',(#13),#28,
    'NASTRAN BDF Converter v1.0.1',('NASTRAN'),
    'AnalysisModelType');
#1130= MATERIAL_PROPERTY('1.2',$,#1109);
#1133= DESCRIPTION_ATTRIBUTE('TangentCTE',#1129);
#1134= ELEMENT_MATERIAL('1','Fea Material',(#1116,#1123,#1129));

#1145= ARBITRARY_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM('',#13);
#1147= VOLUME_3D_ELEMENT_DESCRIPTOR(.LINEAR.,
    'LINEAR_HEXAHEDRON.CHEXA',
    (ENUMERATED_VOLUME_ELEMENT_PURPOSE(.STRESS_DISPLACEMENT.)),
    .HEXAHEDRON.);

#1136= VOLUME_3D_ELEMENT_REPRESENTATION('1',(#1145),#1137,
    (#75,#84,#104,#100,#140,#144,#164,#160),#62,#1147,#1134);
#1149= VOLUME_3D_ELEMENT_REPRESENTATION('2',(#1145),#1137,
    (#84,#88,#108,#104,#144,#148,#168,#164),#62,#1147,#1134);
#1152= VOLUME_3D_ELEMENT_REPRESENTATION('3',(#1145),#1137,
    (#88,#92,#112,#108,#148,#152,#172,#168),#62,#1147,#1134);
```

Listing 2.2: STEP file representation

Officially these files are called ISO 10303-21 files, but are in general referred to as *STEP files*. Another term is STEP P21 files, where P21 refers to the part of the ISO 10303 standard which specifies the format.

STEP data on this form are the most common in implementations and usage. Most CAD applications will offer the functionality of exporting and importing data in this form.

Listing 2.3 shows a snippet of the ISO 10303-28 (XML) representation of a single occurrence of an entity. This form (P28) is less popular than the P21 form. It is however used in some cases in the context of PLM where it may represent an assembly of a product, where each assembly component references P21 files containing the geometry of the parts.

```

<Volume_3d_element_representation id="i6758">
        <Name>1</Name>
        <Items exp:cType="set">
                <Arbitrary_volume_3d_element_coordinate_system
                 xs:nil="true" ref="i6765"/>
        </Items>
        <Context_of_items>
                <Geometric_representation_context-
                parametric_representation_context xs:nil="true"
                ref="i6759"/>
        </Context_of_items>
        <Node_list exp:cType="list">
                <Node xs:nil="true" ref="i6217"/>
                <Node xs:nil="true" ref="i6222"/>
                <Node xs:nil="true" ref="i6232"/>
                <Node xs:nil="true" ref="i6230"/>
                <Node xs:nil="true" ref="i6250"/>
                <Node xs:nil="true" ref="i6252"/>
                <Node xs:nil="true" ref="i6262"/>
                <Node xs:nil="true" ref="i6260"/>
        </Node_list>
        <Model_ref>
                <Fea_model_3d xs:nil="true" ref="i6209"/>
        </Model_ref>
        <Element_descriptor>
                <Volume_3d_element_descriptor xs:nil="true"
                 ref="i6766"/>
        </Element_descriptor>
        <Material>
                <Element_material xs:nil="true" ref="i6757"/>
        </Material>
</Volume_3d_element_representation>
```

Listing 2.3: XML file representation of **volume_3d_element_representation**

Implementing STEP

The standard also specifies how to create APIs interfacing the STEP data model for different programming languages;

- C++, as defined in ISO 10303-23 [27]
- C, as defined in ISO 10303-24 [28]
- Java, as defined in ISO 10303-27 [29]

The language bindings above are all dependent of ISO 10303-22 [30], which defines the *standard data access interface* (SDAI). The SDAI is independent of any programming language and defines an abstract API to manage data models and repositories, and access STEP data in a database. It provides mechanisms for STEP data access regardless of the underlying database format. Some SDAI implementations are discussed in [31, 32, 33].

For applications to work with STEP data, a language interface to the data model is needed. Since EXPRESS is computer readable, APIs may be generated by parsing the data models provided by STEP. The bindings listed above describe how this process can be done for the mentioned languages.

An example of generated C++ code, from Jotne's [34] application EDMS (*EXPRESS Data Manager* ^{TM} [35]), from an EXPRESS data model, is shown in listing 2.4. In this code extract, we see the class declaration of the generated class representing **volume_3d_element_representation**. It contains *get* and *put* functions for the different attributes defined in the EXPRESS specification. As the entity has a supertype, the generated class inherits from its parent class. The definition of the functions, not shown in the code listing, uses SDAI functions to access the data in the database implementation of the EXPRESS data model.

```

class volume_3d_element_representation
    :  public element_representation
{
protected:
   volume_3d_element_representation() {}
public:
   static const entityType eType =
        et_volume_3d_element_representation;
   List<node_representation*>*           get_node_list();
   void          put_node_list(List<node_representation*>* v);
   void          unset_node_list() { unsetAttribute(5); }
   bool          exists_node_list() { return isAttrSet(5); }
   SdaiAggr      get_node_list_aggrId();
   void          put_node_list_element(node_representation*);
   fea_model_3d* get_model_ref();
   InstanceId    get_model_ref_id(entityType *etp = NULL);
   void          put_model_ref(fea_model_3d* v);
   void          put_model_ref_id(InstanceId id);
   void          unset_model_ref() { unsetAttribute(6); }
   bool          exists_model_ref() { return isAttrSet(6); }
   volume_3d_element_descriptor* get_element_descriptor();
   InstanceId    get_element_descriptor_id(entityType *etp = NULL);
   void put_element_descriptor(volume_3d_element_descriptor* v);
   void put_element_descriptor_id(InstanceId id);
   void unset_element_descriptor() { unsetAttribute(7); }
   bool exists_element_descriptor() { return isAttrSet(7); }
   element_material* get_material();
   InstanceId    get_material_id(entityType *etp = NULL);
   void put_material(element_material* v);
   void put_material_id(InstanceId id);
   void unset_material() { unsetAttribute(8); }
   bool exists_material() { return isAttrSet(8); }
   void* operator new(size_t sz, Model *m) {
        return m->allocZeroFilled(sz);
    }
   volume_3d_element_representation(
        Model*     m,
        entityType et = et_volume_3d_element_representation) {
            dbInstance::init(m, et, this);
        }
   volume_3d_element_representation(
        Model*     m,
        InstanceId id,
        entityType et=et_volume_3d_element_representation) {
            dbInstance::init(m, et, this, id);
        }
};
```

Listing 2.4: Generated C++ class declaration for **volume_3d_element_representation**

2.2.2 Integrated Generic and Application Resources (IGR/IAR)

The *Integrated Generic* and *Integrated Application Resources* (IGR and IAR), are where the "core" of the STEP data models lies. The documents in these categories define the core of all the data model entities and types (in EXPRESS) in STEP. Each document handles different topics; some *generic*, and some *application* specific (as the category names suggested). For example, in the code in Listing 2.1, the representation of the **representation** entity, a generic entity which is inherited by a large amount of other entities, comes from an *Integrated Generic Resource* document. The **element_representation** and **volume_3d_element_representation**, which are *application* specific (FEM), comes from a document in the *Integrated Application Resources*.

Some examples of such documents are:

- *Integrated Generic Resources*:
  - ISO 10303-41 - Fundamentals of product description and support
  - ISO 10303-42 - Geometric and topological representation
  - ISO 10303-50 - Mathematical constructs
- *Integrated Application Resources*:
  - ISO 10303-101 - Draugthing
  - ISO 10303-104 - Finite element analysis
  - ISO 10303-110 - Computational fluid dynamics data

Even if entities are specified in each of the different documents, they can reference each other (for example by entities inheriting from entities in other documents), with the exception that data models in the IGRs, may not reference from the IARs.

2.2.3 Application Modules (AM)

The architecture of STEP is layered and modular; data models are divided in *modules* that can be reused by other modules depending on which *layer* they belong to. The lowest layer contains the data models in the IGRs (generic), above this layer are the IARs (application), and on top of these are the *Application Modules* (AM).

AMs can "collect" (by referencing) different parts of the IARs, IGRs and other AMs, thus modularizing data model content. This layering and modularizing, results in a tree structure of modules of AM, IAR, and IGR data models, where the IGRs are leaf nodes. There are multiple top nodes in this tree structure, these are all AMs, and more specifically, *Protocol* AMs.

As seen in figure 2.2, AMs are further divided in *Foundation*, *Implementation*, and *Protocol* categories. Where foundation AMs are the lowest layer, and provide data model content that is highly reusable, and implementation AMs are more specific to a certain usage. Protocol AMs are essentially the same as implementation AMs, however, as "top nodes", they define the content in *Application Protocol*s, where their name comes from.

2.2.4 Application Protocols (AP)

The highest level of the STEP architecture are the *Application Protocols*.

The Application Protocols are the complete data models defined by the aggregated content from a specific protocol AM. Each of them are tailored for a specific engineering application domain, and are the only semantically complete data models.

When an application, tool, or system wants to support STEP for a certain domain, it does so by implementing the data model specified in an AP. The AP can then be used to specify a database dictionary, API, or exchange format for the application. The developers implementing this support, need not to know the whole architecture of STEP (IGR, IAR, AM, etc) but only understand the content of the AP and its language EXPRESS and implementation methods. It is STEP experts, together with domain experts, that design the data models in each STEP module, with high focus on reusing existing modules, for a specific application.

Examples of APs are:

- AP203 Configuration controlled 3D design of mechanical parts and assemblies [36]
  - AP203 was the first application protocol and was intended for CAD data. It was, and is still implemented in many major CAD applications to exchange STEP files (P21) by import and export functionality. It supports 2D/3D geometry, assemblies, annotations, modeling history and much more.
- AP214 Core data for automotive mechanical design processes [37]
  - AP214 came later, also implemented as much as AP203, and supports everything in AP203, but adds support for representing configuration control, tolerance data, kinematics, and more.

- AP242 Managed Model Based 3D Engineering [38]
  - AP242 is the newest application protocol for CAD. It extends the content of AP203 and AP214 with PDM, tessellation (for visualization puroposes), composites representation, and more. This AP is implemented in the newest versions of major CAD tools.
- AP209 Multidisciplinary Analysis and Design
  - AP209 supports simulation information; mainly FEM, but also CFD. It also contains the whole content of AP242, and can therefore represent CAD, simulation, and relations between them.
- AP239 Product Life Cycle Support (PLCS) [39]
  - AP239 [40] is intended for PLM/PDM systems and support topics such as logistics, risk management, tasks, planing, etc.
- AP238 Integrated CNC Machining (STEP-NC) [41]
  - AP238 is first of all a replacement for G and M code for CNC manufacturing. It also supports CAD geometry representations and can relate this to the machining operations.

A lot of content in the above APs overlap, and because this is done by the APs sharing the same modules, interoperability between them is achieved.

2.3 ISO 10303 AP209 - Multidisciplinary Design and Analysis

2.3.1 AP209 Scope

AP209 is the core data model that is used for this study. As mentioned in the previous section, it supports CAD, FEA, CFD, as well as PLM and other CAE related information. All these domains are relevant in the context of digital twins. Some previous studies presenting the use of AP209 are presented in; [42, 43, 44, 45] related to exchange of analysis and composite data, [15, 46] for translation of analysis data, [47, 48] related to electromagnetism and thermal data exchange.

This part of the STEP standard can be used as a format for file exchange between different simulation solvers, and as database schema for SDM applications. It has however not gained the same support by software vendors as AP203, AP214, and AP242 has for the CAD domain.

The first edition of the standard was released in 2001, and was later updated to edition 2 in 2014. A "bug fix" of the current edition is ongoing, and will be released as edition 3

[49]. Jotne is also currently involved in the planing of an international project which will outline the content of a 4th edition. This will focus on nonlinear FEA support, and will take into account the recommendations outlined as a result of this thesis.

The scope of the standard includes, among many other items, the following;

- CAD data (geometry, product information, assemblies of parts)
- FEM load cases; including linear static and dynamic analysis
- FEM loads
- FEM boundary conditions
- FEM mesh (multiple element types, including generic element definition)
- FEM material properties
- Topological relations between FEM mesh and loads, and geometry
- Composite material including its FEM representation

2.3.2 High Level Entities

In STEP, every data representation can be traced back to a **product**, a **product_definition_formation**, and a **product_definition** entity of which they belong. Be it an analysis, a geometric shape, a sensor, or a material type. These entities, together with other related entities, are often referred to as *high level* entities. They hold product information, such as meta data and PLM data.

This section will not present all the details of PLM data representations, but introduce the basic entities that define a *product*.

Product descriptions in STEP, including the above entities, are defined in Parts 41 [50] and 44 [51] of the standard. These documents are focused on what defines a product in terms of its constituents. A product has an identification, categorizations, relations with other products, relation to a specific life cycle stage or discipline view, and may have multiple versions. With the use of those type of entities, this information can be represented semantically.

The formal definitions of the mentioned *product* entities are very generic. According to the documentations, the mentioned *product* entities, and their context and categorization entities, are defined as follows:

- **product**:

  - (...) a representation of a product or a type of product.
  - (...) depends on one or more instances of **product_context** specifying a frame of reference that determines the validity of the information held about the product or class of products.

- **product_definition_formation**:
  - (...) a collector of definitions of a product.

- **product_definition**:
  - (...) a representation of an aspect of a product, or of a class of products, for an identified life cycle stage.
  - (...) may represent particular products that are the members of an identified class of products.
  - (...) acts as an aggregator for information about the properties of products.

- **application_context**
  - (...) is the identification of an application protocol.
  - (...) represents various types of information that relate to product data and may affect the meaning and usage of that data.

- **product_context**
  - (...) is a type of **application_context_element** that represents life cycle independent information about a product. This information describes the discipline in which data about the product are created.

- **product_definition_context**
  - (...) is a type of **application_context_element** that represents information about the stage in the product life cycle for which a **product_definition** is created or used.

- **product_category**
  - (...) is a classification that applies to products.

In many implementations, for example in CAD tools, when a STEP model is exported, *product* related entities will not hold much more information than the product's name and ID, which very often are the same. For assemblies of parts, each part, and each assembly representation will have a **product**, **product_definition_formation**, and a **product_definition**. Also, when exporting from a CAD tool, context and categorization entities will usually hold default values or non-informative data. A CAD tool doesn't necessarily *care* of PLM information. STEP models in such context are more of a method for transfering design data between different applications. However, in a PLM/SDM environment these entities may be populated with information that gives more meaning. In such environment, relation entities may be used to relate different models together, and this is done with product level (high level) entities.

2.3.3 Low Level Entities

In STEP, by low level entities, we mean all the entities describing the details making up the complete data set. Figure 2.3 shows how some of the main content in a sample AP209 file is related. In blue are the "high level" entities representing the analysis as a *product*. The *product* has a shape, which is a FEM model, with a *structural response property* that relates to the details of the FEA model (in green). Further down, in yellow, are the load cases and their associated results. Only the top level entities of the results are shown in the figure for simplicity. Each load case has a tree-structure-like breakdown collecting all the loads and boundary conditions used in the load case. These are composed of *states* and *state_relationships*. In the figure, the breakdown of two load cases are shown. In pink, what relates to loads, and in cyan what relates to boundary conditions. In this specific example, the same boundary conditions are used in both load cases, while the loads have both independent loads and shared loads.

When using AP209 for structural testing, as presented in Paper 1, direct references between the instances representing structural test specific data (tests, test results, and sensors) and the FEA specific data discussed above, are possible. Such tight coupling is required for being able to correctly represent digital twin related data, where physical results, and simulation results, need to be strongly connected.

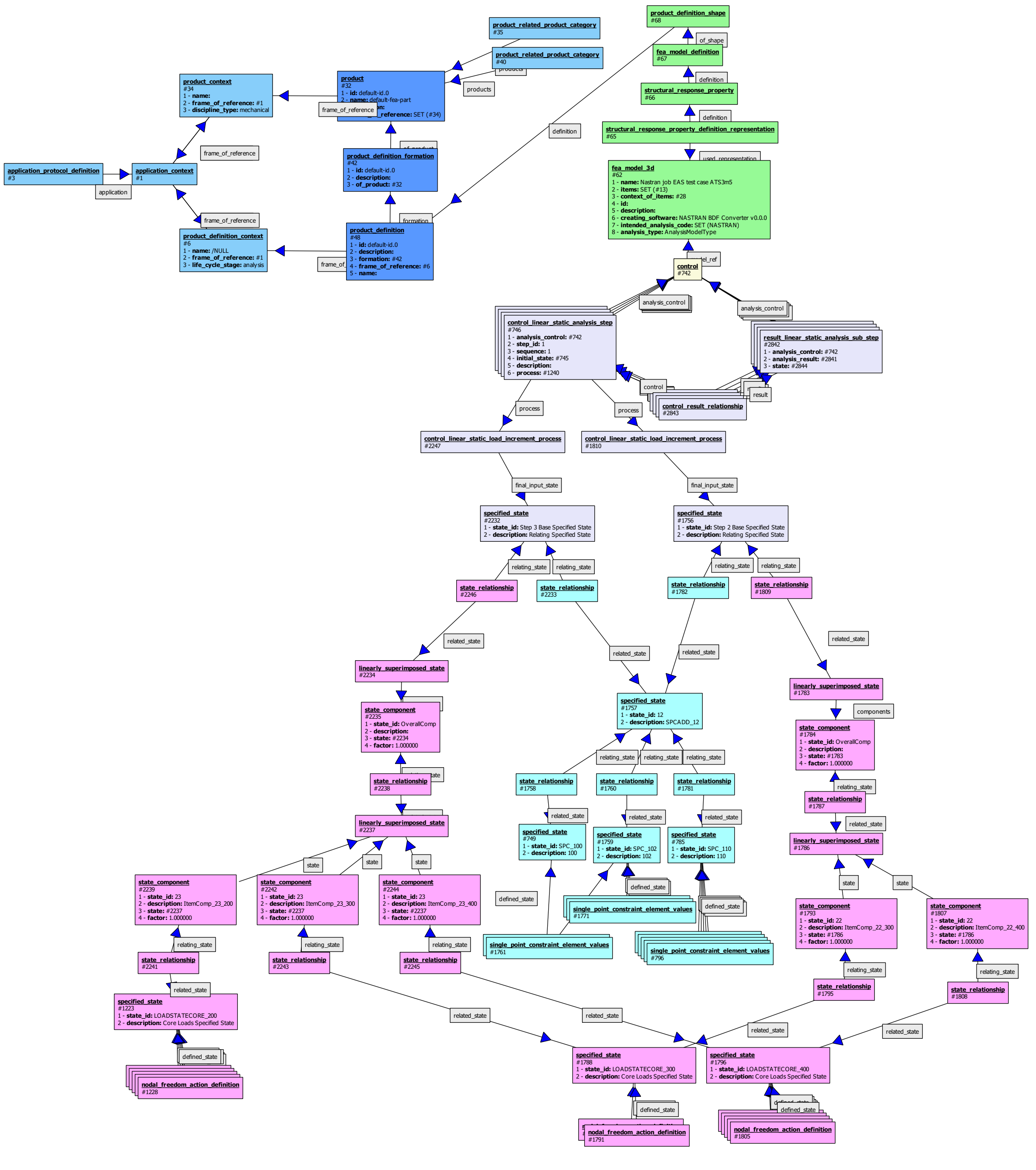


Figure 2.3: Detailed overview of a STEP AP209 population

Chapter 3

Application development

3.1 STEP Converters

To integrate FEM and structural test data in STEP compliant systems (in our case, a SDM system) we need to translate the data from their original source to STEP. A major task of this study was to develop such converters and embed them in a SDM application.

This thesis discusses STEP in the context of data management, and as a format for long term storage. However, converters, especially for FEM, are useful regardless of this context. Being able to quickly convert analysis files from one solver format to another, can save a lot of time for the engineering analyst. In our case, such converters are created by having AP209 as a central format, which every file translation converts to or from. Instead of having direct conversion functionality between each format, every format need only one to- and one from-conversion implementation (Figure 3.1).

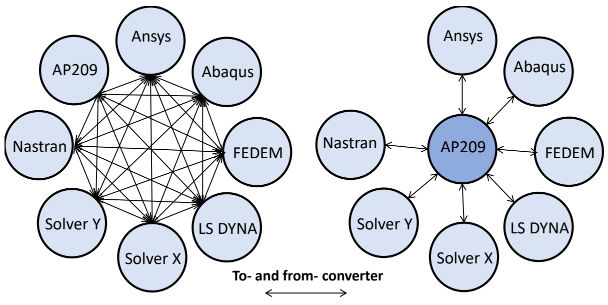


Figure 3.1: AP209 as a central format for converters

This section is dedicated to highlight some of the details and technical aspects involved in the converter development.

3.1.1 FEM Converter

The developed FEM converter works by taking in a source file, a source format, a target file, a target format, and some optional additional configuration parameters. The accepted formats are currently NASTRAN (bulk data format .nas/.bdf/.dat), ABAQUS (input file format, .inp), ANSYS (APDL file format), and AP209, with the following implemented conversions:

NASTRAN	$\Longleftrightarrow$	AP209
ABAQUS	$\Longrightarrow$	AP209
ANSYS	$\Longleftarrow$	AP209

If none of the target and source formats are AP209, the converter converts first to AP209, then to the target format (if that conversion is implemented).

The converter is a command line application created with C++. To work with AP209 data, it uses EDMS and a generated C++ AP209 interface as discussed in 2.2.1. When writing and reading AP209, the converter has direct access to any part of the model. For the native formats however, which are in ASCII, the data needs to be processed sequentially (line by line). Performance-wise, it would be beneficial to work with the solvers binary formats using an API and have direct access to the required parts of the model. Because these formats are proprietary and not open, with no available API, this was not done.

Different strategies may be implemented for converting sequential files. The current method used, is that the application first reads the complete file, store the information internally in memory as custom data structures, then converts the data to AP209.

The main drawback with this is that it sets a limit on the size of model to be converted, based on the available memory. However, such implementation is faster to develop, which has been a crucial benefit during this study.

Alternative implementations, that could be developed, are:

1. As the sequential model is read, convert data directly if possible. If not possible, store the data in memory until the missing information is read, then convert.

2. Same as above, but instead of temporarily storing in memory, store in a database.

3. Read the complete model first, and store everything in a database. This could be a STEP database with a custom AP schema. In that case, an API for the model could be generated and used for the conversion process.

3.1.2 Structural Test Converter

The structural test converter reads input files specifying sensor and test information, as well as result files holding sensor measurement data, and creates AP209 data in a STEP database model. An existing FEM analysis (in AP209) is also input to the converter, such that the created structural test data may relate to the relevant FEM data.

Paper 1 presents how the resulting model from this converter is structured.

The formats of the input files used for sensor and test definition are in ASCII, and based on a format provided by Lockheed Martin Aeronautics in connection with the CRYSTAL project which is discussed in 3.3.2. The files may be written manually, or when integrating the converter in an application, may be generated.

The converter was divided in three different modes:

- **Sensor definition mode:** Creates sensors as product data structures in a specified AP209 database model. Based on the data in the sensor input files, sensors are referencing finite elements on the input FEM mesh, and their orientation, position, type, identification, and other relevant information are added as properties.

  The sensors also reference the STEP representation of the physical product that the sensors are placed on.

- **Test definition mode:** Creates the representation of an executed or planned structural test in the same model.

  Based on the data in the test input files, each test gets linked to their corresponding load case that simulates them in a FEM model. They also reference the specific sensors that are used in each tests, and the physical product which the tests are applied on.

- **Test result mode:** Reads tests results from files generated from DAQ (Data acquisition) systems, such as CATMAN [52], and an input file specifying how each set of result data relates to a specific sensor. The results are collected in STEP data structures and related to their sensors and test cases.

3.2 STEP Explorer

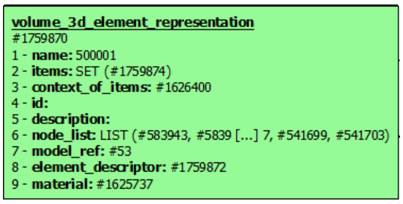
3.2.1 Background

STEP is a complex data model covering a huge amount of concepts. In this study a lot of work was spent exploring, understanding, and extending the standard with new concepts, including generating STEP files with developed converters. This type of work involves a lot of debugging and inspection of generated STEP files, which could be done by reading STEP files in a text editor, or viewing its content in Jotne's database manager, EDMS. This proved to be time consuming for complex files, especially when attempting to explore deeply nested tree-like data structures. Because of this, the study devoted a certain amount time on developing an application, STEP Explorer, for doing such tasks more efficiently.

3.2.2 Overview

The main purpose of STEP Explorer is to give an easy way of exploring the content of STEP files and models. This is achieved by providing a 2D graphical view of STEP instances with their relations.

STEP Explorer (see Figure 3.3) requires that a STEP schema is specified, then a STEP file can be imported. The application initially gives an overview of the different entities (Figure 3.4) that exist in the STEP file. For each different entity, a tabular view can be displayed showing every instance of that type and its attribute values (Figure 3.6).



volume_3d_element_representation
#1759870
1 - **name:** 500001
2 - **items:** SET (#1759874)
3 - **context_of_items:** #1626400
4 - **id:**
5 - **description:**
6 - **node_list:** LIST (#583943, #5839 [...] 7, #541699, #541703)
7 - **model_ref:** #53
8 - **element_descriptor:** #1759872
9 - **material:** #1625737

Figure 3.2: Example of instance box

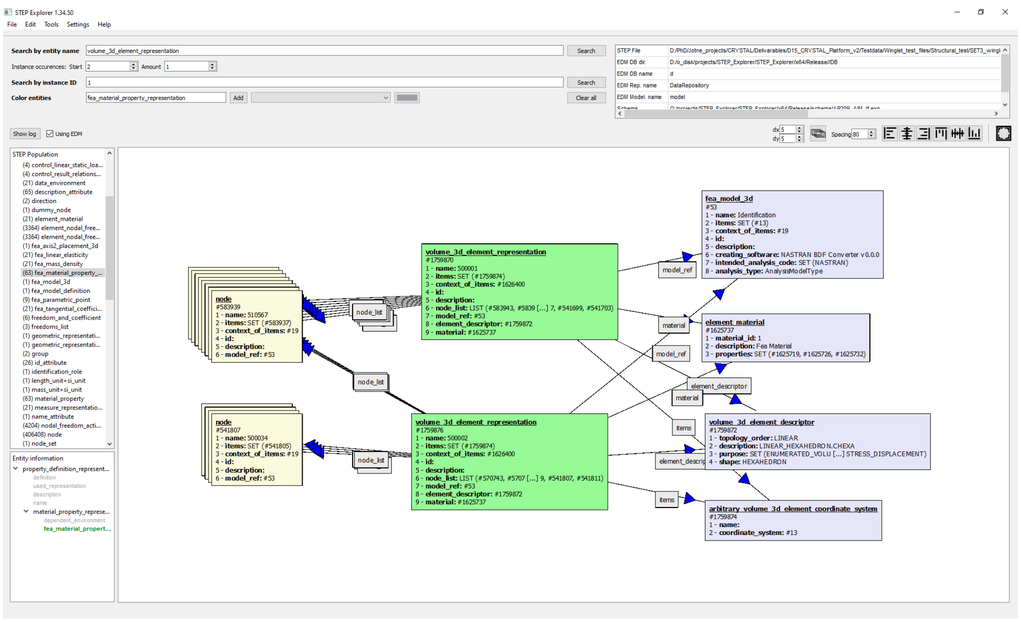
By selecting instances in the tabular instance overview, or by searching for an entity name or instance ID (Figure 3.5), instances can be displayed in the graphical 2D view.
In the viewer, displayed instances are represented by boxes, containing their entity name, instance ID, and attribute values. When attributes are references to other instances, the referenced instance ID is displayed. An example of such box can be seen in Figure 3.2.

The displayed instance boxes have a context menu with the following main available functionalities:

- **Get all attributes:** every attribute of the selected instances which are instance references are displayed as additional boxes.

- **Get attribute:** gives the choice to select any of the attributes that are instance references and display it as an additional box.

- **Get references:** A query is done to find all instances that reference the selected instances. If this results in many results, they are displayed in a table form and a selection can be done of the desired instances, which are then displayed as additional boxes. If there are few results, additional instance boxes are directly created for each.



Search by entity name	volume_3d_element_representation
Instance occurences: Start	2
Amount	1
Search by instance ID	1
Color entities	fea_material_property_representation

☑ Using EDM

dx 5, dy 5, Spacing 80

STEP File	D:/PhD/Jotne_projects/CRYSTAL/Deliverables/D15_CRYSTAL_Platform_v2/Testdata/Winglet_test_files/Structural_test/SET3_wingl
EDM DB dir.	D:/o_disk/projects/STEP_Explorer/STEP_Explorer/x64/Release//DB
EDM DB name	d
EDM Rep. name	DataRepository
EDM Model. name	model
Schema	O:/projects/STEP_Explorer/STEP_Explorer/x64/Release/schema/AP209_AIM_lf.exp

STEP Population
(4) control_linear_static_loa...
(4) control_result_relations...
(21) data_environment
(65) description_attribute
(2) direction
(1) dummy_node
(21) element_material
(3364) element_nodal_free...
(3364) element_nodal_free...
(1) fea_axis2_placement_3d
(21) fea_linear_elasticity
(21) fea_mass_density
(63) fea_material_property_...
(1) fea_model_3d
(1) fea_model_definition
(9) fea_parametric_point
(21) fea_tangential_coeffici...
(6) freedom_and_coefficient
(3) freedoms_list
(1) geometric_representati...
(1) geometric_representati...
(2) group
(26) id_attribute
(1) identification_role
(1) length_unit+si_unit
(1) mass_unit+si_unit
(63) material_property
(21) measure_representatio...
(1) name_attribute
(4204) nodal_freedom_acti...
(406408) node
(1) node_set

Entity information
property_definition_represent...
definition
used_representation
description
name
material_property_represe...
dependent_environment
fea_material_propert...

node
#583939
1 - **name:** 510567
2 - **items:** SET (#583937)
3 - **context_of_items:** #19
4 - **id:**
5 - **description:**
6 - **model_ref:** #53

node
#541807
1 - **name:** 500034
2 - **items:** SET (#541805)
3 - **context_of_items:** #19
4 - **id:**
5 - **description:**
6 - **model_ref:** #53

volume_3d_element_representation
#1759870
1 - **name:** 500001
2 - **items:** SET (#1759874)
3 - **context_of_items:** #1626400
4 - **id:**
5 - **description:**
6 - **node_list:** LIST (#583943, #5839 [...] 7, #541699, #541703)
7 - **model_ref:** #53
8 - **element_descriptor:** #1759872
9 - **material:** #1625737

volume_3d_element_representation
#1759876
1 - **name:** 500002
2 - **items:** SET (#1759874)
3 - **context_of_items:** #1626400
4 - **id:**
5 - **description:**
6 - **node_list:** LIST (#570743, #5707 [...] 9, #541807, #541811)
7 - **model_ref:** #53
8 - **element_descriptor:** #1759872
9 - **material:** #1625737

fea_model_3d
#53
1 - **name:** Identification
2 - **items:** SET (#13)
3 - **context_of_items:** #19
4 - **id:**
5 - **description:**
6 - **creating_software:** NASTRAN BDF Converter v0.0.0
7 - **intended_analysis_code:** SET (NASTRAN)
8 - **analysis_type:** AnalysisModelType

element_material
#1625737
1 - **material_id:** 1
2 - **description:** Fea Material
3 - **properties:** SET (#1625719, #1625726, #1625732)

volume_3d_element_descriptor
#1759872
1 - **topology_order:** LINEAR
2 - **description:** LINEAR_HEXAHEDRON.CHEXA
3 - **purpose:** SET (ENUMERATED_VOLU [...] STRESS_DISPLACEMENT)
4 - **shape:** HEXAHEDRON

arbitrary_volume_3d_element_coordinate_system
#1759874
1 - **name:**
2 - **coordinate_system:** #13

Figure 3.3: STEP Explorer interface

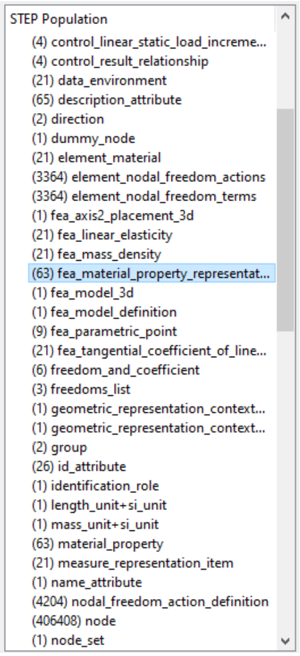
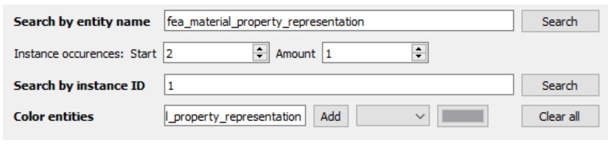
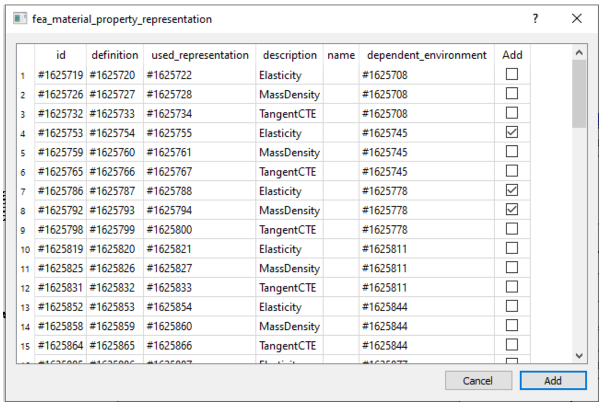

STEP Population
(4) control_linear_static_load_increme...
(4) control_result_relationship
(21) data_environment
(65) description_attribute
(2) direction
(1) dummy_node
(21) element_material
(3364) element_nodal_freedom_actions
(3364) element_nodal_freedom_terms
(1) fea_axis2_placement_3d
(21) fea_linear_elasticity
(21) fea_mass_density
(63) fea_material_property_representat...
(1) fea_model_3d
(1) fea_model_definition
(9) fea_parametric_point
(21) fea_tangential_coefficient_of_line...
(6) freedom_and_coefficient
(3) freedoms_list
(1) geometric_representation_context...
(1) geometric_representation_context...
(2) group
(26) id_attribute
(1) identification_role
(1) length_unit+si_unit
(1) mass_unit+si_unit
(63) material_property
(21) measure_representation_item
(1) name_attribute
(4204) nodal_freedom_action_definition
(406408) node
(1) node_set

Figure 3.4: Entity list showing which entities exist in the STEP model.



Search by entity name	fea_material_property_representation
Instance occurences: Start	2
Amount	1
Search by instance ID	1
Color entities	l_property_representation

Figure 3.5: Search instance by entity name or instance ID


fea_material_property_representation

	id	definition	used_representation	description	name	dependent_environment	Add
1	#1625719	#1625720	#1625722	Elasticity		#1625708	☐
2	#1625726	#1625727	#1625728	MassDensity		#1625708	☐
3	#1625732	#1625733	#1625734	TangentCTE		#1625708	☐
4	#1625753	#1625754	#1625755	Elasticity		#1625745	☑
5	#1625759	#1625760	#1625761	MassDensity		#1625745	☐
6	#1625765	#1625766	#1625767	TangentCTE		#1625745	☐
7	#1625786	#1625787	#1625788	Elasticity		#1625778	☑
8	#1625792	#1625793	#1625794	MassDensity		#1625778	☑
9	#1625798	#1625799	#1625800	TangentCTE		#1625778	☐
10	#1625819	#1625820	#1625821	Elasticity		#1625811	☐
11	#1625825	#1625826	#1625827	MassDensity		#1625811	☐
12	#1625831	#1625832	#1625833	TangentCTE		#1625811	☐
13	#1625852	#1625853	#1625854	Elasticity		#1625844	☐
14	#1625858	#1625859	#1625860	MassDensity		#1625844	☐
15	#1625864	#1625865	#1625866	TangentCTE		#1625844	☐

Figure 3.6: Instance list, showing all occurrences of an entity in the STEP model.

When an attribute or reference is displayed through one of the above functions, an arrow is displayed showing the relationship between them. This functionality makes it possible to create diagrams that show complex STEP data structures in a simplistic way. Instance boxes can be moved, aligned, centered, colored, and more, to allow the user to create detailed diagrams as seen in Figure 3.7. These may be exported to PDF or image files to be shared with others.

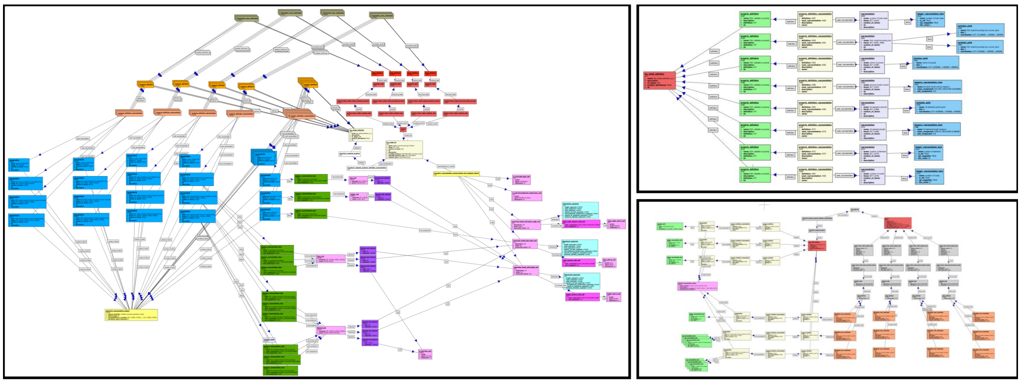


Figure 3.7: Example of generated diagrams

An additional functionality are scripts. A script, using a simple syntax, can be written to define small template-like instructions of which instances, and how instances, should be displayed in the viewer. These template scripts can be used on multiple STEP files to quickly generate standard diagrams that can be compared.

3.3 STEP in Simulation Data Management

As already mentioned, STEP, and more precisely STEP AP209, is a data model that can be used in a simulation data management system. The data model provides multiple PLM data structures for general data management, and for low level details of simulations.

In this study, the research has been heavily related to the use of, and the extension of Jotne's SDM application; EDMopenSimDM^{TM} [53]. This has been used to validate the use of AP209 for managing design, testing, and simulation, as stated in objective **O.1**.

As part of this validation, a use case was introduced, involving design, FEA, manufacturing, testing, and data management of an aircraft winglet, Figure 3.8. The design of the generic winglet was provided by Lockheed Martin Aeronautics, it was simulated with NX Nastran and manufactured and tested at NTNU. EDMopenSimDM with the extensions provided by this study and the CRYSTAL project discussed in 3.3.2, and the converters discussed in 3.1, was used to manage all data.

The winglet use-case was designed to involve the data relevant for a digital twin. Though it didn't take into account any real-time connection between the physical and virtual twin, it satisfies the coverage of the data domains of the thesis' objective. A real life complex industrial digital twin, will in most cases involve some method of feedback between an analysis and sensor measurements on a product in operation. The data transfer mechanism for such a system, can be implemented in a multitude of ways, and is out of scope of this study. In our use-case, the winglet, rather than being in operation, is tested in a lab and the data is collected and imported to the SDM application, to be integrated with the virtual digital twin data; the design and FEM analysis.

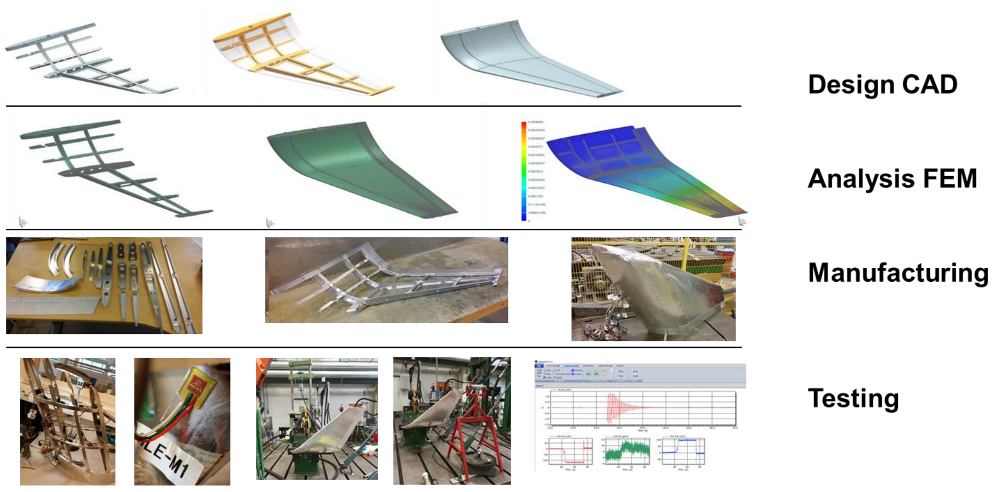


Figure 3.8: Winglet use case

3.3.1 EDMopenSimDM

EDMopenSimDM^{TM} addresses simulation data management and engineering data archival and retention. The tool is composed of a client and a server application, which enables for collaboration between people and teams within the same, or across different companies.

The main concept of EDMopenSimDM is that it is based on the AP209 data model, which is used as an underlying database dictionary.

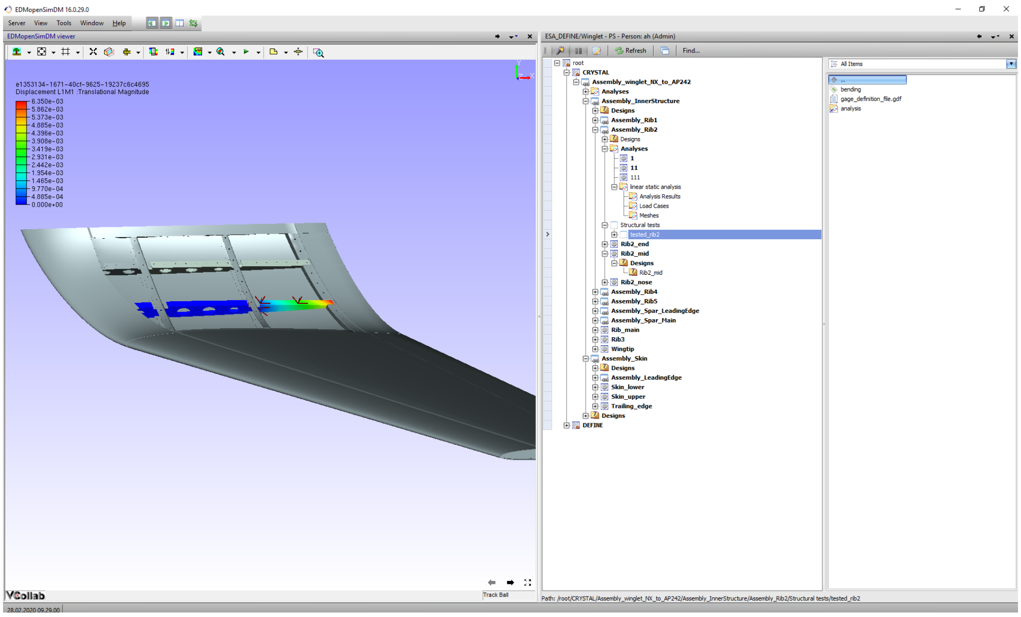


Figure 3.9: EDMopenSimDM interface

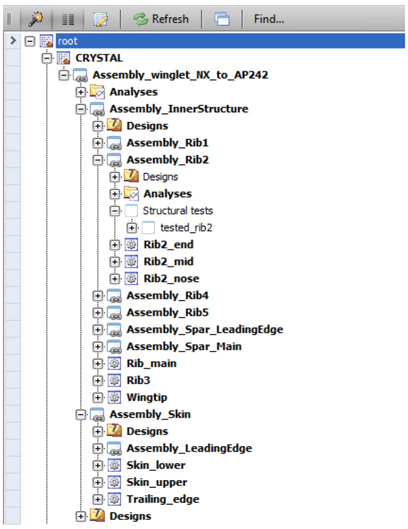


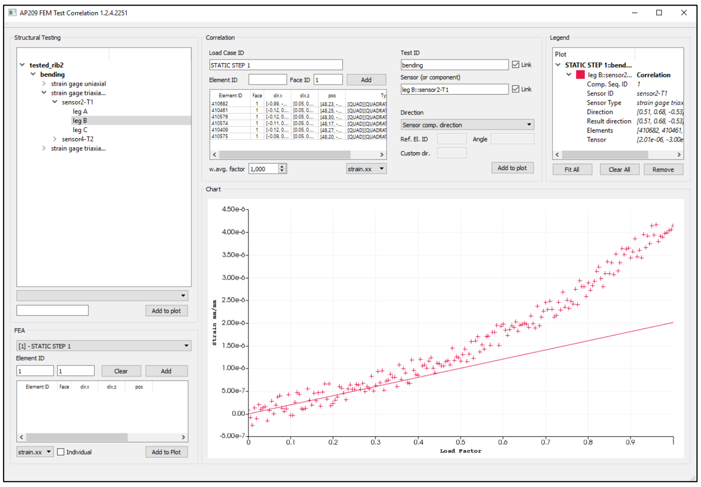
Figure 3.10: EDMopenSimDM product structure

All management concepts are stored as AP209 data structures, which includes; people and organization with access restrictions, tasks and tasks methods, approvals, versioning, etc. The application accepts files in any formats, but has special processing methods for STEP files.
If a STEP file containing a CAD assembly is imported, the assembly structure may be used to define a product structure in a project. Product structures are presented as seen in Figure 3.10. In this particular example the product breakdown structure is a result of importing the CAD assembly of the winglet exported from NX [54].

EDMopenSimDM has integrated the concept of *federated model*. A federated model implies that every model (STEP models) imported is treated as sub-models of the overall federated model of a project. Relations between sub-models are managed through a special *link* model (using the STEP AP209 schema). Each model, may be of different domains, including; CAD, FEA, and structural testing. Section 2.3.2 presented high level entities (**product**, **product_definition**, etc), and how they hold product management information. Relations between sub-models are first of all done on these entities. In addition, for specific cases, low level relations are also created. This includes for example the representation of a test case in a structural test model, and a load case in a FEA model, or a sensor representation and a FEA element. These types of relations are done with the converter presented in 3.1.2. In theory, more detailed relations could be done, such as relations between FEA nodes, elements, loads, and their related CAD geometric constituents such as surfaces and edges. This is however not implemented as it would require integrated CAD and FEA kernels.

Having all this data and information in the SDM application, in one data model, facilitates the integration of data processing tools that requires cross-domain information. An example of such a tool, which further validates objective **O.1**, is the *FEM-Test Correlation tool* (Figure 3.11). This application was developed during this study, and embedded into the SDM application. It accesses the information of a selected structural test and related FEM model, and allows the user to select sensors and test cases. These are related to test result values and FEM results. For each sensor, their test measurements and results from analysis can be plotted individually or overlapped, enabling them to be compared and checked for correlation. Additional information for each sensor is also available, such as orientation, position, type, etc.

Another example of service developed, which takes the advantage of tightly coupled FEM and sensor data, is the visualization of sensors on its related FEM mesh (Figure 5).


AP209 FEM Test Correlation 1.2.4.2251

Structural Testing

- tested_rib2
  - bending
    - strain gage uniaxial
    - strain gage triaxia...
      - sensor2-T1
        - leg A
        - leg B
        - leg C
      - sensor4-T2
    - strain gage triaxia...

Correlation

Load Case ID: STATIC STEP 1

Element ID: Face ID: 1

Element ID	Face	dir.x	dix.z	pos	Ty
410682	1	[-0.99, -...	[0.05, 0....	[48.23, -...	[QUAD][QUADRAT
410461	1	[-0.12, 0....	[0.05, 0....	[48.25, -...	[QUAD][QUADRAT
410576	1	[-0.12, 0....	[0.05, 0....	[48.30, -...	[QUAD][QUADRAT
410574	1	[-0.11, 0....	[0.05, 0....	[48.17, -...	[QUAD][QUADRAT
410409	1	[-0.12, 0....	[0.05, 0....	[48.27, -...	[QUAD][QUADRAT
410575	1	[-0.09, 0....	[0.05, 0....	[48.20, -...	[QUAD][QUADRAT

w.avg. factor: 1,000 strain.xx

Test ID: bending [✓ Link]

Sensor (or component): leg B::sensor2-T1 [✓ Link]

Direction: Sensor comp. direction

Ref. El. ID: Angle:

Custom dir.:

Legend

Plot	
STATIC STEP 1::bend...	
🟥 leg B::sensor2...	Correlation
Comp. Seq. ID	1
Sensor ID	sensor2-T1
Sensor Type	strain gage triax
Direction	[0.51, 0.68, -0.53]
Result direction	[0.51, 0.68, -0.53]
Elements	[410682, 410461,
Tensor	[2.01e-06, -3.00e

FEA

[1] - STATIC STEP 1

Element ID: 1 1

Element ID	Face	dir.x	dix.z	pos

strain.xx ☐ Individual

Figure 3.11: FEM-Test Correlation tool interface

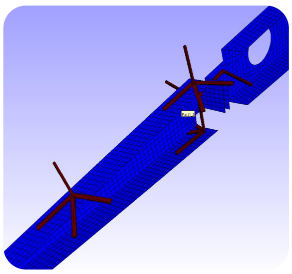


Figure 3.12: Sensors visualized on FEM mesh

3.3.2 CRYSTAL Project

The study of this thesis started while Jotne had a project named CRYSTAL with Lockheed Martin Aeronautics Company. Part of the objectives of this thesis were also requirements in the scope of CRYSTAL. The project's goal was to explore and implement a system for a central storage of design, simulation, and structural test data. This system was EDMopenSimDM, and many of the features presented in 3.3.1, were developed during the project using results from this thesis, especially Paper 1. Paper 4 presents the vision and scope of CRYSTAL.

3.3.3 DEFINE Project

The DEFINE project is an ongoing ESA (European Space Agency) project done together with Jotne. Its main objective is to increase the integration of 3D digital models in order to improve the efficiency and effectiveness of the assembly, integration, and test procedures and documentation, fully integrated into the overall space system lifecycle. It is similar to the CRYSTAL project described in section 3.3.2, however, in addition to FEA and structural test data, it introduces additional domains, including; thermo and thermo-elastic analysis, optical raytracing analysis, cloud points data, testing data from thermal scan, vibration testing, load cells, and others.

Results from this thesis will be applied in this context to further improve the functionality and domain coverage of EDMopenSimDM.

Chapter 4

Summary of Papers

Paper 1 together with the implementations in SDM (section 3.3), is the main contribution to objective **O.1**. This paper highlights the details around utilizing AP209 for structural testing.

Paper 2, contributing to **O.2**, identifies some of the obstacles for AP209 to gain more acceptance from the industry. The main concern is the lack of support of nonlinear FEM, which is the main topic addressed in this paper.

Paper 3 contributes to both **O.1** and **O.2**. The main concern from Paper 2 is addressed here, by showing how AP209 can be enhanced to cover nonlinear FEM. Nonlinear FEM is required for both digital twin data, which can involve complex dynamic simulations, and for general FEM data management that might involve complex analyses.

An abstract and summary of each paper are presented in the next sections.

4.1 Paper 1 - Relating Structural Test and FEA data with STEP AP209

Abstract

This paper proposes a method for incorporating FEA data and structural test data into one common standardized data model based on the ISO 10303 STEP Standard. The proposed method takes advantage of data structures and elements defined in STEP AP209 Edition 2 to provide traceability between analysis and testing phases; information such as sensor and finite elements, test and FEA load cases, and test and FEA results are included. It also presents an introduction to STEP and AP209e2, and discusses how it can be used in a Simulation Data Management environment.

This paper focuses on **O.1** by investigating how to handle structural test data with the STEP standard. It presents the technical aspect of how this was done, and answers the question; how can STEP AP209, which is intended for multidisciplinary design and analysis, cover structural testing?

The reason for using AP209 for this domain, is because not only should the standard represent the structural test data, but it should also represent the *relations* to data from FEM, as well as other domains. AP209 already covers generic data representations such as PLM constructs, mathematical concepts, general engineering information, generic properties, and more. These representations can all be related to FEA specific data representations. The paper presents how these generic representations can be used to represent test data and relate those to analysis data.

The study focused on how to handle the following information:

- how to represent sensors with focus on strain gauges.
- how to represent sensor components (for example in biaxial/triaxial gauges), and how to relate these to parent sensor assemblies, and to test results in a specific test.
- how sensors relate to a position and elements in the FEM model.
- how sensors relate to a specific test representation and a physically tested product.
- how the test cases relate to load cases in the FEM model.
- how to handle sensor specific properties, such as gauge factor, sensor type and model, and DAQ (Data Aquisition) connection IDs.

The converter for structural test data, discussed in section 3.1.2, was developped during the study of this paper, first as a stand alone application, then integrated in EDMopenSimDM.

4.2 Paper 2 - ISO 10303 AP209 - Why and how to embed nonlinear FEA

Abstract

ISO 10303 STEP AP209 edition 2 is a data model standard intended for data exchange and storage of simulation information. The standard has a wide coverage of FEA (Finite Element Analysis) information, but is missing certain features such as nonlinear FEA. This paper gives an introduction to the STEP AP209 standard and presents projects in which AP209 has been implemented.

The study then identifies requirements that should be supported by a standard FEA data model, but are not fully covered by AP209. Each requirements are discussed in the context of how they are supported by existing major solver applications. Without giving detailed solutions for how these should be implemented in AP209, starting points for further research is suggested.

For AP209 to be more widely used it needs to gain interest from more engineering application vendors, especially FEM solver vendors. This study presents STEP and AP209, and discusses projects and use cases where it has been implemented and used. It further tries to identify why it has not gained as much traction as the STEP application protocols for CAD (AP203, AP214, and AP242). One of the main issues identified is the lack of support for certain important FEA concepts. How major FEA solvers support these missing FEA concepts, and how these should be implemented in AP209, is investigated. No implementations or detailed solutions are discussed, but further work is suggested. Paper 3 and Annex C continue this study in more details.

4.3 Paper 3 - Extending STEP AP209 for Nonlinear Finite Element Analysis

Abstract

ISO 10303 STEP AP209 is a standard for exchanging and storing simulation information along side related PLM (Product Lifecyle Management), CAD (Computer Aided Design), and other CAE (Computer Aided Engineering) data. The AP209 standard, despite being well documented and covering a wide range of engineering information, has not been widely implemented by FEA (Finite Element Analysis) solver or SDM (Simulation Data Management) applications. This is assumed to mainly be due to AP209 not yet supporting nonlinear FEA.
The following study takes basis in the findings of ***Paper 3****, where improvements of the AP209 standard were suggested. Some of these suggestions, related to nonlinear FEA, are here further investigated and implemented, and proposed for further standardization.*
Analysis test cases using these new features are created, and converters between different FEA formats are developed.
The test cases are nonlinear, static and dynamic, with different defined time step control parameters and loading conditions. The FEA data converters translates data between AP209 and the solver specific formats. The complete data information from the analyses are preserved during the conversion, and the generated analyses are solved. To confirm that no information was lost during the process, simulation results are investigated and compared.

Paper 3 continues from the findings in Paper 2. The study shows how AP209 can be extended to support multiple types of analysis, and not only linear FEA. This is done by extending the entity hierarchy of the analysis type entities, and introducing the representation of FEA solver parameters to the standard. In addition, as it is often needed in nonlinear analyses, a recommended representation of varying load (by time or space) is described.

The improvements to the standard are implemented in a custom AP209 schema, which is implemented in the FEA converter discussed in 3.1.1. With this converter, different test cases created in Abaqus are converted to Ansys and Nastran formats via a translation to AP209. These tests are solved in the respective solvers, and to confirm that AP209 kept all the necessary information during the conversions, and the results are compared.

4.4 Summary of STEP AP209 Extensions

From Paper 1

The following summarizes the STEP usage recommendations from Paper 1 related to how to represent structural test data and its relation to FEM.

Below, the term *product structure* is referring to an instance structure of a set of **product**, **product_definition_formation**, and **product_definition** instances.

- Representation of a physically tested part
  - With relation to:
    - Design model
    - Analysis
    - Physical tests
- Representation of a sensor type
  - Represented by a *product structure*
  - With sensor type specific properties and metadata such as; manufacturer, model name, description, type identifier, angles for multi-axial strain gauges, etc.
- Representation of a sensor
  - Represented by a *product structure*
  - Assembly of multiple sensor components (for example different measurement directions in a multi-axial strain gauge)

  - With sensor specific properties and metadata such as; ID, name, geometrical location, mesh location (elements and nodes)

- Representation of a sensor component
  - Represented by a *product structure*
  - With sensor component specific properties and metadata such as; measurement direction
  - With relations to:
    - Measurement results
    - Test cases
- Representation of a physical test case
  - With relations to:
    - The physically tested part
    - Measurement results from sensor components
    - Sensors used in the test
    - Load case in analysis
- Representation of results of specific test
  - With properties and metadata such as; DAQ specific properties, sampling rate, gauge factor, channel ID, etc.
  - With relations to:
    - Sensor component
    - Test case
- Representation of collection of physical test results
  - Represented by a *product structure*
  - With relations to all tests performed on a physically tested part and the analyses simulating those tests

From Paper 3

The following summarizes the AP209 extensions and recommendations from Paper 3:

- Analysis types
  - Extensions of **control_analysis_step**, **result_analysis_step** and **control_process**

    - New entities for each type of analysis
    - Hierarchically structured

- Analysis parameters
  - Extension of **property_definition** and **property_definition_representation**
    - With entities **fea_parameter_property** and **fea_parameter_property_definition_representation**
    - Used to relate FEA parameters to specific load case or complete analysis
  - Added new type ***control_or_control_process***
    - Used by **fea_parameter_property** and **fea_parameter_property_definition_representation** to be able to either relate to a specific load case or a complete analysis
  - Recommendation of standardized FEA parameters to be used by **fea_parameter_property** and **fea_parameter_property_definition_representation** for common FEA properties, structured by categories
- Varying loads
  - Representation of loads varying by time, load factor, coordinates, or other variables

From Appendix C

- Mesh regions
  - Extension of entity **element_group**
    - Extended with element type specific groups that can hold an element aspect (face ID and edge ID)
    - Multiple groups can be related to form regions defined by sub-regions with different element aspects
- Mesh interactions (contact and glue)
  - Extension of entity **state_definition**
    - Extended with entities for specific mesh regions, and interaction types such as contact and glue
    - Relates to interaction specific properties represented by the entities **property_definition** and **property_definition_representation**
- Nonlinear materials

- Representation of elastic perfectly plastic material model defined by a yield point
- Represenstation of multilinear plasticity material model defined by a set of stress and plastic strain values

Chapter 5

Conclusion and Future Work

The STEP standard covers a large scope of industrial data. However, it is only in the CAD domain that it has been widely accepted and implemented. Today thousands of STEP CAD files are produced and archived daily world-wide [55], it has proven to simplify CAD file exchange between different systems. A similar wide implementation in other domains would greatly simplify data management and archiving in engineering industry.

With the main goal of improving the effectiveness of data management of physical test and simulation data, in the context of digital twins, PDM, PLM and SDM, this study has investigated, proposed recommendations, and applied the use of STEP for structural testing and FEM.

For the structural testing domain we showed how test data can be stored without making changes to the standard, but by defining how to structure the data in a semantical correct way. It was illustrated how to implement these recommendations in a SDM application, which can then manage both structural test and FEA data in the same format. The application, which also stores design data in STEP, is able to relate models and information together, ensuring interoperability and traceability.

The need for nonlinear FEM to be supported by AP209 was identified, and after making recommended extensions to the standard, implementations were performed to show their applicability.

While this thesis recommends many additions to the standard and makes STEP even more applicable for use in digital twins, for these recommendations to become available for industry, they need to be officially endorsed by the ISO community and published in official ISO standards and recommended practices. Furthermore, the domains of structural testing, and nonlinear FEM are still larger than what was covered in this study. This study covers the basics, but future work would be required, to apply these results in a wider set of use cases including additional types of sensors and more complex simulations.

Bibliography

[1] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 2020.

[2] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415, April 2019.

[3] Thomas H.-J. Uhlemann, Christian Lehmann, and Rolf Steinhilper. The digital twin: Realizing the cyber-physical production system for industry 4.0. *Procedia CIRP*, 61:335 – 340, 2017. The 24th CIRP Conference on Life Cycle Engineering.

[4] Qinglin Qi, Fei Tao, Tianliang Hu, Nabil Anwer, Ang Liu, Yongli Wei, Lihui Wang, and A.Y.C. Nee. Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, 2019.

[5] Wim Gielingh. An assessment of the current state of product data technologies. *Computer-Aided Design*, 40(7):750 – 759, 2008. Current State and Future of Product Data Technologies (PDT).

[6] Merja Peltokoski, Mika Lohtander, and Juha Varis. The role of product data management (pdm) in engineering design and the key differences between pdm and product lifecycle management (plm). 04 2014.

[7] NAFEMS Simulation Data Management Working Group. *What is simulation data management?* NAFEMS, 2014.

[8] Andrea Buda, Petri Makkonen, Ronan Derroisne, and Vincent Cheutet. Pdm suitability study for cae data management. 07 2011.

[9] ISO 10303-1:1994. Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles. Standard, International Organization for Standardization, Geneva (Switzerland), 1994.

[10] P. Brun, F. Fernández González, M. Lehne, A. McClelland, and H. Nowacki. A neutral product database for large multifunctional systems. In R. Vio and W. Van Puymbroeck, editors, *Computer Integrated Manufacturing*, pages 87–97, London, 1991. Springer London.

[11] Soonhung Han, Young Choi, Sangbong Yoo, and Namkyu Park. Collaborative engineering design based on an intelligent step database. *Concurrent Engineering: R&A*, 10:239–249, 09 2002.

[12] Jeongsam Yang, Soonhung Han, Matthias Grau, and Duhwan Mun. Openpdm-based product data exchange among heterogeneous pdm systems in a distributed environment. *The International Journal of Advanced Manufacturing Technology*, 40(9):1033–1043, February 2009.

[13] Han M. Shih. Migrating product structure bill of materials excel files to step pdm implementation. *International Journal of Information Management*, 34(4):489 – 516, 2014.

[14] D. Iliescu, I. Ciocan, and I. Mateias. Assisted management of product data: A pdm application proposal. In *2014 18th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 128–133, Oct 2014.

[15] Sébastien Charles and Benoit Eynard. Integration of cad and fea data in a pdm environment: Specification of a step simulation data management schema. *17th IMACS World Congress*, 02 2005.

[16] Guillaume Ducellier, Sébastien Charles, Benoit Eynard, and Emmanuel Caillaud. Traceability of simulation data in a plm environment: Proposition of a step-based system that support parameter integration. *9th International Design Conference, DESIGN 2006*, 01 2006.

[17] ISO 10303-209:2014. Industrial automation systems and integration – Product data representation and exchange – Part 209: Application protocol: Multidisciplinary analysis and design. Standard, International Organization for Standardization, Geneva (Switzerland), 2014.

[18] R. Lanza, J. Haenisch, K. Bengtsson, and T. Rlvåg. Relating structural test and fea data with step ap209. *Advances in Engineering Software*, 127:96 – 105, 2019.

[19] R. Lanza, J. Haenisch, K. Bengtsson, and T. Rlvåg. ISO 10303 AP209 - Why and how to embed nonlinear FEA. Under review, 2020.

[20] R. Lanza, J. Haenisch, K. Bengtsson, and T. Rlvåg. Extending STEP AP209 for Nonlinear Finite Element Analysis. Under review, 2020.

[21] J. Haenish, K. Bengtsson, R. Lanza, and O. Liestl. Open Simulation Data Management and Testing, The Crystal Project. *NAFEMS World Congress 2017*, page 161, 2017.

[22] ISO 10303-11:2004. Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual. Standard, International Organization for Standardization, Geneva (Switzerland), 2004.

[23] ISO 10303-21:2016. Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure. Standard, International Organization for Standardization, Geneva (Switzerland), 2016.

[24] The HDF Group. https://www.hdfgroup.org/solutions/hdf5/. Accessed on 29 March 2020.

[25] ISO 10303-26:2011. Industrial automation systems and integration – Product data representation and exchange – Part 26: Implementation methods: Binary representation of EXPRESS-driven data. Standard, International Organization for Standardization, Geneva (Switzerland), 2011.

[26] 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. Technical report, International Organization for Standardization, 2007.

[27] ISO 10303-23:2000. Industrial automation systems and integration – Product data representation and exchange – Part 23: Implementation methods: C++ language binding to the standard data access interface. Technical report, International Organization for Standardization, 2000.

[28] ISO 10303-24:2001. Industrial automation systems and integration – Product data representation and exchange – Part 24: Implementation methods: C language binding of standard data access interface. Technical report, International Organization for Standardization, 2001.

[29] ISO 10303-27:2000. Industrial automation systems and integration – Product data representation and exchange – Part 27: Implementation methods: Java TM programming language binding to the standard data access interface with Internet/Intranet extensions. Technical report, International Organization for Standardization, 2000.

[30] ISO 10303-22:1998. Industrial automation systems and integration – Product data representation and exchange – Part 22: Implementation methods: Standard data access interface. Standard, International Organization for Standardization, Geneva (Switzerland), 1998.

[31] Richard M. Botting and Anthony N. Godwin. Analysis of the step standard data access interface using formal methods. *Computer Standards & Interfaces*, 17(5):437 – 455, 1995. Formal Description Techniques.

[32] A. Goh, S.C. Hui, and B. Song. An integrated environment for product development using step/express. *Computers in Industry*, 31(3):305 – 313, 1996. Product and process data modelling.

[33] S Ma, Y Maréchal, and J.-L Coulomb. Methodology for an implementation of the step standard: a java prototype. *Advances in Engineering Software*, 32(1):15 – 19, 2001.

[34] Jotne EPM Technology AS. http://www.jotneit.no. Accessed on 14 February 2020.

[35] EXPRESS Data Manager^{TM} Software Development Kits. http://www.jotneit.no/products/express-data-manager-edm, 2019. Accessed on 14 February 2020.

[36] ISO 10303-203:2011. Industrial automation systems and integration – Product data representation and exchange – Part 203: Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies. Standard, International Organization for Standardization, Geneva (Switzerland), 2011.

[37] 10303-214:2010. Industrial automation systems and integration – Product data representation and exchange – Part 214: Application protocol: Core data for automotive mechanical design processes. Standard, International Organization for Standardization, Geneva (Switzerland), 2010.

[38] ISO 10303-242:2014. Industrial automation systems and integration – Product data representation and exchange – Part 242: Application protocol: Managed model-based 3D engineering. Standard, International Organization for Standardization, Geneva (Switzerland), 2014.

[39] ISO 10303-239:2012. Industrial automation systems and integration – Product data representation and exchange – Part 239: Application protocol: Product life cycle support. Technical report, International Organization for Standardization, 2012.

[40] PDES, Inc, AFNeT. ISO 10303 (STEP) AP 239 edition 3 Application Protocol For Product Life Cycle Support (PLCS). Technical report, 2015.

[41] ISO 10303-238:2012. Industrial automation systems and integration – Product data representation and exchange – Part 238: Application protocol: Product life cycle support. Standard, International Organization for Standardization, Geneva (Switzerland), 2014.

[42] Keith A. Hunten. Cad/fea integration with step ap209 technology and implementation. *MSC Aerospace Users Conference Proceedings*, 1997.

[43] N. Pitre K.A. Hunten, J.W. Klintworth and T.E. Mack. New standards based data exchange bridge for design (cad), analysis (cae) and manufacturing (cam) of composite structures. In *MSC 1999 Aerospace Users Conferance Proceedings*, 1999.

[44] Edward L. Stanton, Takman Mack, Hiren D. Patel, and Jamie Klintworth. Composite beam models using iso step ap209. 2003.

[45] Keith A. Hunten, Allison Barnard Feeney, and Vijay Srinivasan. Recent advances in sharing standardized step composite structure design and manufacturing information. *Computer-Aided Design*, 45(10):1215 – 1221, 2013.

[46] Peter Bartholomew and Christian Paleczny. Standardization of the finite element analysis data-exchange in aeronautics concurrent engineering. *Journal of Computing and Information Science in Engineering - JCISE*, 5, 03 2005.

[47] Eric Lebègue, Georg Siebes, and Charles Stroom. Thermal analysis data exchange between esa and nasa with step. 07 1999.

[48] Ma Singva. Step standard's evaluation for modeling in electromagnetism. *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, 18(3):311–323, March 2020.

[49] ISO/AWI 10303-209 Industrial automation systems and integration — Product data representation and exchange — Part 209: Application protocol: Multidisciplinary analysis and design - edition 3. https://www.iso.org/standard/75060.html, 2019. Accessed on 02 March 2020.

[50] ISO 10303-41:2000. Industrial automation systems and integration – Product data representation and exchange – Part 41: Integrated generic resource: Fundamentals

of product description and support. Standard, International Organization for Standardization, Geneva (Switzerland), 2000.

[51] ISO 10303-44:2019. Industrial automation systems and integration – Product data representation and exchange – Part 44: Integrated generic resource: Product structure configuration. Technical report, International Organization for Standardization, 2019.

[52] HBM. CATMAN DAQ Software. https://www.hbm.com/en/2290/catman-data-acquisition-software/. Accessed on 2March 2020.

[53] EDMopenSimDM (version 12.0). http://www.jotneit.no/edmopensimdm, 2019. Accessed on 14 February 2020.

[54] NX (version 11.0). https://www.plm.automation.siemens.com/global/en/products/nx/, 2016.

[55] C. Jackson and D. Prawel. The 2013 State of 3D Collaboration and Interoperability Report (page 17). https://www.plm.automation.siemens.com/en_us/Images/Lifecycle-Insights-2013-Collaboration-Interoperability_tcm1023-210162.pdf, 2013. Accessed on 14 February 2020.

[56] A.W.A. Konter. *Advances Finite Element Contact Benchmarks*. NAFEMS, 2006.

[57] NAFEMS. *Introduction to Non-Linear Finite Element Analysis*. NAFEMS, 2000.

Appendices

Appendix A

Main publications

A.1 Paper 1 - Relating Structural Test and FEA data with STEP AP209

Advances in Engineering Software 127 (2019) 96–105

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research paper

Relating structural test and FEA data with STEP AP209

R. Lanza*,[a],[b], J. Haenisch[a], K. Bengtsson[a], T. Rølvåg[b]

[a] Jotne EPM Technology AS, Grenseveien 107, Oslo 0663, Norway
[b] Norwegian University of Science and Technology, Richard Birkelandsvei 2B, Trondheim, Norway

ARTICLE INFO

Keywords:
STEP ISO 10303
FEM Analysis
Structural testing
Data exchange
Simulation data management

ABSTRACT

This paper proposes a method for incorporating FEA data and structural test data into one common standardized data model based on the ISO 10303 STEP Standard [1]. The proposed method takes advantage of data structures and elements defined in STEP AP209 Edition 2 [2] to provide traceability between analysis and testing phases; information such as sensor and finite elements, test and FEA load cases, and test and FEA results are included. It also presents an introduction to STEP and AP209e2, and discusses how it can be used in a Simulation Data Management environment.

1. Introduction

Simulation and structural testing plays a big role in the development of complex products. As Moore's Law continues to evolve, greater computational power and storage becomes available for use. This has led to an ever-increasing amount of simulations, especially as design optimization through simulation and analysis becomes more common. The higher computational power allows engineers to perform more complex analyses with higher fidelity than ever before. Properly applied, high fidelity methods can lead to more optimized and safer products.

Enormous amounts of data are generated by these methods that must be managed effectively and efficiently. Problems arise when these data must be stored for reuse in different domains or when they have to be archived for a longer term. The large amount of data means finding information becomes more difficult. Files in different formats, for different applications, spread over multiple locations and companies further complicates the situation. A popular solution to these difficulties is often declared to be Simulation Data Management (SDM) and Product Data Management (PDM). These solutions make organizing simulation and CAD data together with other engineering information more efficient, but have focused more on the CAD aspects of data management. The aerospace industry (among others) has recognized the growing challenges related to SDM and PDM for analysis and simulation data and have been active in promoting SDM and PDM solutions.

Still with SDM, users are often locked to proprietary formats of the software initially used for their design and simulation, causing complications when different partners are using different software. SDM is not the main focus of this paper, but as we will see, AP209 is not only used as a file format but it could also be the backbone of the data model for a software system (including Data Management tools).

The reliability of simulation data depends on their validation by physical tests. For safety critical systems, authorities may require this relationship to be traceable. Test data, therefore, need to be managed together with corresponding simulation data. This adds to the complexity of the data management task. A typical (and simplified) engineering process that involves structural testing is as follows:

1. A simulation is performed and results are saved in the CAE software' s native format.
2. Based on the results, actuator and sensor locations are chosen for a structural test.
3. Parameters for controlling the test are developed based on simulation results.
4. Tests are performed and loads and results are exported from the test equipment to a test specific format.
5. Test results and simulated results are compared and reconciled.
6. Results are summarized in test reports and delivered to consuming organizations.

Companies often have their own internal work-flows to manage interactions between the analysis and testing organizations during test planning and preparations up and throughout test execution. Additional work-flows are used to compare, reconcile, document and distribute the product testing results.

These work-flows can be performed manually or through automation but both rely on sets of agreed-upon definitions. The following types of information are a few examples of these definitions:

* Corresponding author at: Norwegian University of Science and Technology, Richard Birkelandsvei 2B, Trondheim, Norway.
E-mail addresses: remi.lanza@jotne.com (R. Lanza), jochen.haenisch@jotne.com (J. Haenisch), terje.rolvag@ntnu.no (T. Rølvåg).

https://doi.org/10.1016/j.advengsoft.2018.08.005
Received 5 March 2018; Received in revised form 27 July 2018; Accepted 12 August 2018

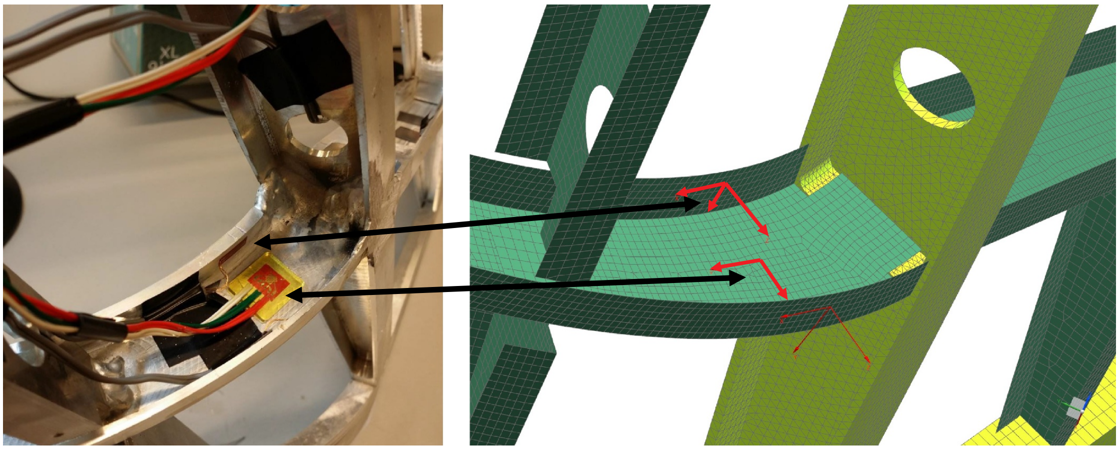


Fig. 1. Mapping sensor locations and orientations to FEM model.

1. Sensor distribution in the FE model frame of reference. (Figure 1)
2. Sensor orientations in the FE model frame of reference. (Figure 1)
3. Relation between corresponding test cases and analysis load cases.
4. Sensor mapping to channel IDs from the test equipment.
5. Information about applied filtering techniques on applied loads and sensor result data.

With these definitions, the correspondence of virtual and physical results can be validated against the testing requirements for data content and quality. Common data manipulation techniques, such as transforming the results to matching frame of reference, enable consistent predictions or comparisons and are highly dependent on the common understanding of the kinds of definitions described above.

These operations are performed in a variety of software tools with results typically output to Excel sheets for further analysis or reporting.

Data artifacts are generated at many of the steps in these workflows and must be retained to achieve full traceability. Examples of these data artifacts are the following:

1. Test Requirements
2. Test Plans and Procedures
3. FEM analysis files
4. FEM result files
5. Structural test output files
6. FEM-Structural test definition files
7. Comparison / correlation results
8. Reports

In certain industries there exist strong regulations on data retention of products. This is the case for the aerospace industry. As an example, the Federal Aviation Administration (FAA) in the United States, requires that *'Type design data must be retained and accessible for the lifespan of the product. It is possible that technical support for the original software will be terminated during the product lifespan, so your procedures manual must explain how access to the data will be retained or transitioned to a new software system.'* [3].

The goal of this paper is to validate that the AP209 data model has the capabilities to represent the above information, and keep the traceability between the different data fields. Thus, enabling the storage of a complete data set in a neutral and archive-friendly format.

Fig. 2presents an overview of the data which we want to represent in AP209, and how it relates together internally in a model.

In the next sections we briefly cover the background of the STEP ISO-10303 standard, followed by Section 3 where we present the outline of the proposed model, while Sections 4–6 go into specific details of the data model.

2. STEP ISO 10303

2.1. Background

Started in 1998, the goal of ISO 10303 was to standardize the representation of product data that are aggregated throughout the whole product life-cycle and across all relevant domains. The data model that STEP standardizes is written in the data modeling language EXPRESS [4], a lexical and graphical language which is both human and computer readable. EXPRESS is an object-oriented language using encapsulation and inheritance; it offers rich features for specifying population constraints.

Part 21 of the STEP standard [5] describes the ASCII representation of STEP, which is commonly known as the *STEP file format*. In addition, STEP defines an API to access product data in STEP compliant database repositories for data sharing. This is standardized Part 22 *SDAI, Standard Data Access Interface* [6]. Programming language interfaces for STEP data, so called language bindings, are specified in for example Part 23 [7] for C++. Having all these standardized methods for accessing STEP data, simplifies the creation of STEP based tools and software, and allows these to share a unified understanding of the data.

The standard is composed of a collection of parts, some of which covers the implementation methods of the standard, such as the parts mentioned above, while most parts specify the data models of the different product data domains supported by the standard, i.e. geometric representations, FEA, mathematical descriptions, product structures etc. Each of these are holding the definition of entities with their attributes and inheritance, which in an Object-Oriented Programming (OOP) view are essentially classes.

2.2. STEP Architecture

An important aspect of the STEP architecture is the use of higher level data models, which by using formal mapping specifications, maps to the integrated resources and the application resources of ISO 10303. Only a brief description of this process will be included in this paper. The reader is advised to study *STEP in a Nutshell* [8] and the *STEP Application Handbook* [9] for a thoroughly explanation of the STEP architecture.

The main idea is that an *Application Activity Model* (AAM) is used to describe the activities and data flows of a certain use case of the

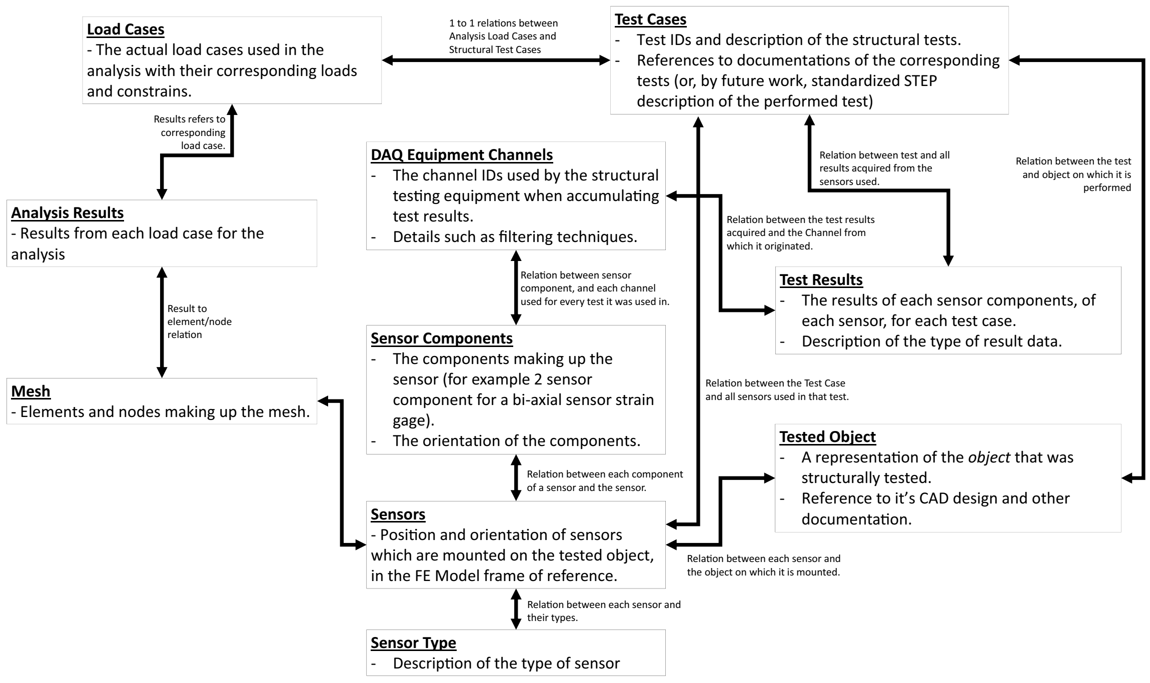

Load Cases
- The actual load cases used in the analysis with their corresponding loads and constrains.
1 to 1 relations between Analysis Load Cases and Structural Test Cases
Test Cases
- Test IDs and description of the structural tests.
- References to documentations of the corresponding tests (or, by future work, standardized STEP description of the performed test)
Results refers to corresponding load case.
DAQ Equipment Channels
- The channel IDs used by the structural testing equipment when accumulating test results.
- Details such as filtering techniques.
Relation between test and all results acquired from the sensors used.
Relation between the test and object on which it is performed
Analysis Results
- Results from each load case for the analysis
Relation between the test results acquired and the Channel from which it originated.
Relation between sensor component, and each channel used for every test it was used in.
Test Results
- The results of each sensor components, of each sensor, for each test case.
- Description of the type of result data.
Result to element/node relation
Sensor Components
- The components making up the sensor (for example 2 sensor component for a bi-axial sensor strain gage).
- The orientation of the components.
Mesh
- Elements and nodes making up the mesh.
Relation between the Test Case and all sensors used in that test.
Tested Object
- A representation of the object that was structurally tested.
- Reference to it's CAD design and other documentation.
Relation between each component of a sensor and the sensor.
Sensors
- Position and orientation of sensors which are mounted on the tested object, in the FE Model frame of reference.
Relation between each sensor and the object on which it is mounted.
Relation between each sensor and their types.
Sensor Type
- Description of the type of sensor

Fig. 2. Overview of main data represented in an AP209 model containing Analysis and Structural Test data, and how they relate.

Application Protocol (see Section 2.3). This is usually done by graphically illustrating the flows of types of data necessary for the to-be-developed STEP data model.

Based on the AAM, a formal *Application Reference Model* (ARM) can be designed; this may be modelled in the EXPRESS language. The intention is that this model is built by experts of the product data domain in question. Data objects and attributes are defined using terminology well understood in the specific domain.

The *Application Interpreted Model* (AIM) is the lowest level data model. AIMs contain the exact same information as the ARMs, but mapped by mapping specifications to a formally defined and generic format which is uniform across all usages of the STEP standard. Because of the complexity and genericness of the STEP standard, such mapping and encoding is usually done by a STEP expert in cooperation with domain experts.

The author of this paper has focused on the feasibility of using the AP209 AIM for structural test data; the formalities of publishing the findings of this study as part of ISO 10303 are not discussed in detail. This is a natural next task.

2.3. Application Protocols

Each Application Protocol (AP) focuses on a specific domain or phase in the product life-cycle. An AP specifies a single ARM to define its content, which, as described in the previous section, maps to an interoperable AIM. The AIM objects are defined in what are called *Integrated Resources* (IR). These IRs are in turn defined in the several parts of which the STEP standard consists.

A certain application or software supporting STEP, defines which AP it covers, that is, which share of the total STEP data model. APs are, thus, the view of the standard offered to implementors of data exchange, sharing and archiving solutions. STEP files refer one or several APs, but are all based on the same type of data structure, the so called PDM schema. They have the same high level definitions, allowing SDM and PDM tools to easily process files from different domains (i.e. CAD, FEA, manufacturing). STEP has also several managements concepts (such as requirements, assignments, classifications, roles, activities...) embedded within certain parts, which can be directly integrated within a Data Management tool.

Since the initial release of STEP in 1994, AP203 [10] and AP214 [11] have been the most successful Application Protocols, and are now widely used as exchange formats between CAD and PLM software.

In Fig. 3 we see how entities with inheritance and attributes are defined in an ISO 10,303 Part which in turn is used by an Application Protocol. The example shows two high level entities, **representation_item** and **representation** which belongs to Part 43 [12]. This Part has many generic entities that are used by all APs. Each entity may be a parent (*supertype*) of multiple entities which are defined in other Parts that further specializes them. For simplicity the figure shows a single inheritance branch (**representation_item** and **representation** actually have many child (*subtypes*) entities defined in other parts).

AP209, which covers the domain Analysis and Design, includes Part 42 [13], Part 43 and Part 104 [14], while AP242 [15], intended as a CAD format, includes only Part 42 and 43. Both AP209 and AP242 include shares of several other Parts which are not shown in the figure. Fig. 4 shows how **representation, element_representation** and **surface_3d_element_representation** are defined in the standard AP documents in the EXPRESS language.

A STEP file or database holds a population of instances of these entities, and can be interpreted by an application that implements the AP schema; an extract of such a STEP file is included in Section 3.1.

2.4. Application protocol 209

AP209 is called *Multidisciplinary analysis and design*, and is primarily meant to specify simulation solver relevant data for exchange, sharing and archival. An overview of the data that it can represent is shown in Fig. 5.

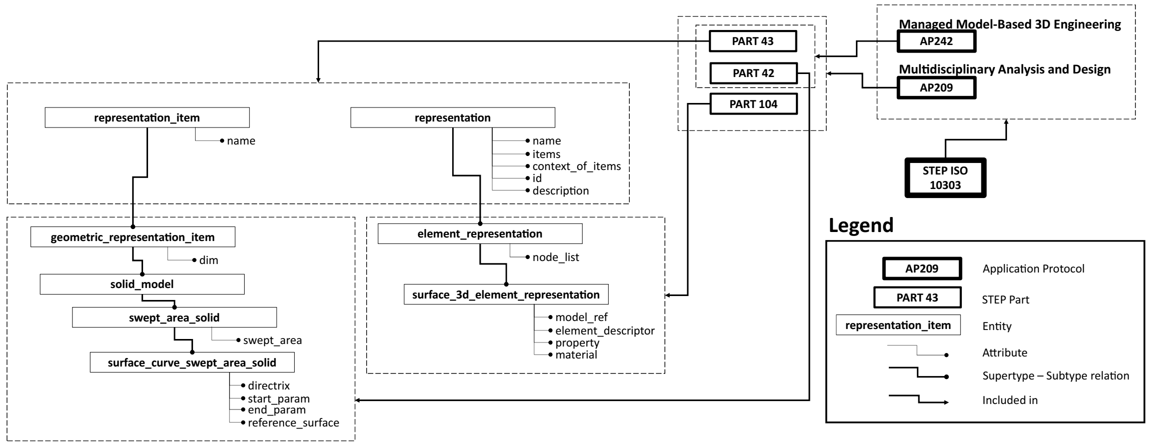


Fig. 3. Example of how entities are included in Parts which again are included in Application Protocols.

It covers the representation of composites, analysis definition and analysis results (FEM and CFD), design (CAD) and more. Note that an important aspect of AP209 is the capability of not only representing analysis and design *separately*, but also allowing the interconnection between both domains (such as relationships between mesh and loads, and the design geometry). Most modern FEM and CFD pre-processors have implemented geometry based mesh generation and load definitions in their applications, but the solver interfaces are still generally based on traditional ASCII cards formats.

Currently, only linear statics and linear modal analyses are completely supported in AP209. However, as noted by [16], the standard was designed to easily be updated to support non-linear analysis, as it already covers roughly 90% of this problem.

Multiple implementations of the standard have been performed, but these have been limited in scope, focusing on the exchange of composite data between design, analysis and manufacturing purposes or the basic FEA model entities. Several of these are summarized in [18].

Ongoing work and implementations of the standard are led by the LOTAR EAS: Engineering Analysis & Simulation Workgroup [19] which is co-chaired by Airbus and Boeing. They have been active in promoting commercial implementation of the AP209 standard with rigorous testing criteria.

As described earlier, an AP is composed of several STEP parts, which are principally schema that specify the content, form and structure of a set of entities (classes). Parts can be used in different APs, therefore many entities are general in nature. They can be viewed as *building-blocks* for representing certain classes of items or concepts. As we will see in the next section, these *building-blocks* or entities, can be used, not only to represent FEA and CAD, but also information concerning structural testing, as long as the new use of the existing structures is defined accordingly. The next sections describes an outline of a proposed structure for using AP209 to represent the additional data required for representing structural testing information. No extensions of the AP209 standard are suggested, but as will be discussed, future work may recommend changes or extensions.

3. The higher structure of a combined structural & FEA STEP model

3.1. Overview

This subsection introduces several key concepts used extensively in the subsequent sections.

In STEP high level items are represented as a **product**. By high level item we mean, *an Analysis, a CAD assembly, a CAD part, a manufactured part* etc. A product is a foundational concept that allow an item to be described, categorized, referenced, tracked, and versioned in ways that are familiar to modern day product data management users. Items that would not be considered a **product** could be a FEM element, a color definition, a property, a geometric shape etc.

```
ENTITY representation;
  name : label;
  items : SET[1:?] OF representation_item;
  context_of_items : representation_context;
DERIVE
  id : identifier := get_id_value (SELF);
  description : text := get_description_value (SELF);
WHERE
  WR1: SIZEOF (USEDIN (SELF, 'BASIC_ATTRIBUTE_SCHEMA.' + 'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
  WR2: SIZEOF (USEDIN (SELF, 'BASIC_ATTRIBUTE_SCHEMA.' + 'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY;


ENTITY surface_3d_element_representation
    SUBTYPE OF ( element_representation );
      model_ref           : fea_model_3d;
      element_descriptor  : surface_3d_element_descriptor;
      property            : surface_element_property;
      material            : element_material;
    UNIQUE
      ur1 : model_ref, SELF\representation.name;
    WHERE
      wr1:
      wr2:
      wr3:
      fu1:
END_ENTITY;
```

```
ENTITY element_representation
    SUPERTYPE OF (
        ONEOF (
            volume_3d_element_representation,
            axisymmetric_volume_2d_element_representation,
            plane_volume_2d_element_representation,
            surface_3d_element_representation,
            axisymmetric_surface_2d_element_representation,
            plane_surface_2d_element_representation,
            curve_3d_element_representation,
            axisymmetric_curve_2d_element_representation,
            plane_curve_2d_element_representation,
            point_element_representation,
            directionally_explicit_element_representation,
            explicit_element_representation,
            substructure_element_representation ) )
    SUBTYPE OF ( representation );
      node_list  : LIST [1 : ?] OF node_representation;
    WHERE
      wr1:
    END_ENTITY;
```

Fig. 4. Extract of the content in the AP209 document. (Some fields are left out for simplicity.)

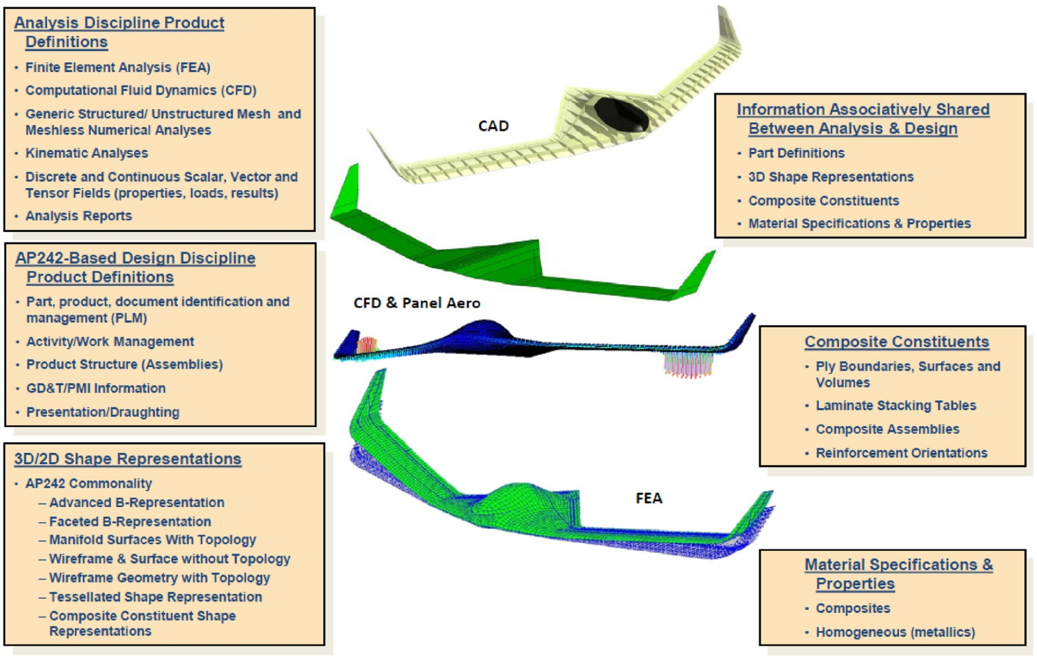

Analysis Discipline Product Definitions
• Finite Element Analysis (FEA)
• Computational Fluid Dynamics (CFD)
• Generic Structured/ Unstructured Mesh and Meshless Numerical Analyses
• Kinematic Analyses
• Discrete and Continuous Scalar, Vector and Tensor Fields (properties, loads, results)
• Analysis Reports
AP242-Based Design Discipline Product Definitions
• Part, product, document identification and management (PLM)
• Activity/Work Management
• Product Structure (Assemblies)
• GD&T/PMI Information
• Presentation/Draughting
3D/2D Shape Representations
• AP242 Commonality
– Advanced B-Representation
– Faceted B-Representation
– Manifold Surfaces With Topology
– Wireframe & Surface without Topology
– Wireframe Geometry with Topology
– Tessellated Shape Representation
– Composite Constituent Shape Representations
CAD
CFD & Panel Aero
FEA
Information Associatively Shared Between Analysis & Design
• Part Definitions
• 3D Shape Representations
• Composite Constituents
• Material Specifications & Properties
Composite Constituents
• Ply Boundaries, Surfaces and Volumes
• Laminate Stacking Tables
• Composite Assemblies
• Reinforcement Orientations
Material Specifications & Properties
• Composites
• Homogeneous (metallics)

Fig. 5. Data which is supported by AP209 [17].

The **product** entity has certain mandatory attributes and related entities. For example, a **product** entity must have a version, a context and a category classifying the **product**. The detailed data entities that make up an instance of an analysis or CAD model, relate to a **product_definition** entity. In turn, this **product_definition** relate to a **product_definition_formation** which provides versioning for an instance of a **product**. These high level product entities also provide links to the **application_context** and **application_protocol_definition** that identify which STEP schema the data conforms to. Lastly, a variety of optional information about people, organizations, dates and times and other related metadata can be linked to these top level product entities.

This structure ensures that an application importing or accessing a **product** (such as an SDM tool) can understand what is being imported before handling the complete model. These high level entities serve to organize multiple STEP populations within the same system. Multiple STEP data-sets residing within a database removes the constraint that they be considered *files*. The constituents of each model or data set, are then identified by their relationship to the high level product entities. In this fashion, complex interrelated data-sets can be constructed which reduces data duplication.

An extract of a STEP P21 file (ASCII) showing some of these high level entities can be seen in Fig. 6. As shown, each instance of an entity has an identifier followed by the entity name. The attributes are enclosed by parentheses and comma separated. When an entity is an attribute of another entity, it is referenced by this identifier. Throughout the paper, graphical instantiation of this structure will be used (not to be confused with EXPRESS-G which is the standardized graphical representation of the EXPRESS language defined in Part 11). Instances are represented by boxes with the entity name in capital letters. Arrows show the referencing between instances. A string beside an arrow specifies the name of the attribute. In some cases STEP entity structures can be quite complex. If an entity box has its text in italic, it represents a simplification of a more complex structure, or a shortening of the entity name. Bold text beside an entity box is an additional description for the reader to better relate the graphics to the context.

```
(...)
#42= PRODUCT('1234','winglet analysis','',(#44));
#53= PRODUCT_DEFINITION_FORMATION('v.2','',#42);
#59= PRODUCT_DEFINITION('winglet analysis fine mesh',$,#53,#60);
#70= PRODUCT_RELATED_PRODUCT_CATEGORY('linear_static_analysis',$,(#42));
#44= PRODUCT_CONTEXT('design_context',#1,'design_context');
(...)
```

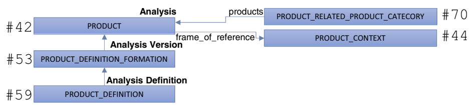


Fig. 6. Left: Extract of a STEP P21 file. Right: Graphical representation used in this paper.

It is also important to understand that in addition to these high level entities, many of the low level entities such as nodes, elements and loads can hold addtional meta-data such as names, labels and descriptions. STEP post- and pre-processors can implement these, to describe intentions and comments regarding the creation, review and modification of the model.

3.2. The analysis model

The data structure of an Analysis STEP AP209 data set is well described in the Recommended Practices for AP209 [20].

A few details of the data structure will be discussed here, focusing on the parts that will have a relationship to the structural testing data.

In AP209 the analysis is represented by a **product** entity, which was described in the previous section. This analysis **product** has a version and a definition. The entity **product_definition_shape** represents the shape of the product used for the analysis. The **product_definition_shape** can include the idealization of the CAD model (abstraction), node sets, and more importantly, from the analysis perspective, the **fea_model_definition**. The **fea_model_definition** is the link to the

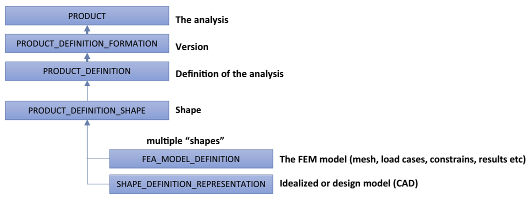

PRODUCT
The analysis
PRODUCT_DEFINITION_FORMATION
Version
PRODUCT_DEFINITION
Definition of the analysis
PRODUCT_DEFINITION_SHAPE
Shape
multiple "shapes"
FEA_MODEL_DEFINITION
The FEM model (mesh, load cases, constrains, results etc)
SHAPE_DEFINITION_REPRESENTATION
Idealized or design model (CAD)

Fig. 7. High level entities in the Analysis Model.

nodes and elements making up the mesh **shape** and additional FEA related information. The whole analysis definition is then built up of entities linked to one another to give structure and meaning to the data.

Analysis load cases in AP209 are represented by **control_linear_static_analysis_step** entities that relates different **states**. Each **state** is a collector of loads, constraints or other nested **states**. Section 6.3 shows how the model relates the states to the actual test cases.

3.3. *The structural test model*

Fig. 7shows the high level structure of the FEA model. Fig. 8 introduces a similar structure representing the object that is being tested. The **product** in this case is the tested part which also has a version, definition and shape. The two versions are linked via relationship entities. The shape may be linked to the same Nominal Design data set, which is already related to the Analysis forming a consisten data set. The product being tested could be related to its own unique design version of needed.

Another **product** represents all the result data from all tests that relates to load cases in the Analysis Model. This product has a version as well, and multiple definitions with each representing the results from individual structural tests.

The sensors and the tests are also represented by STEP entities. The sensors are related to the tested part **product**, while the tests relate the sensors and the test results. Sensors and test representations are further discussed in Sections 4 and 5 respectively.

4. Sensors

There exists a wide variety of sensors such as strain gages, accelerometers, vibration sensors, displacements sensors and more. Many of these are assemblies of multiple sensors, for example a triaxial gage is just three sensors assembled together with specified angles between them.

To generically cover all types of sensors we represent each sensor as an assembly of multiple sensor components. Each sensor assembly and each component has its own **product** with a definition holding properties.

To avoid repetitive information, we introduce a **product** representing the type of sensors used. As an example, the specification of a tri-axial strain gage of a specific type, brand and model would be represented by one sensor type **product**. For each sensor of this type, mounted on the tested part, there exists a sensor assembly **product** having three individual sensor component **products**.

Each of the representations, sensor, sensor component and sensor type, are able to hold properties. Properties that are related to the sensor assembly:

1. **Position:** the position based on the coordinate system of the FE model
2. **Orientation:** the orientation of the sensor in the FE model
3. **Reference Element:** the element (or a set of elements) in the analysis model on which the sensor is placed
4. **Element Face ID:** an ID (or a set of IDs) representing the face of the elements on which the sensor is placed

Properties that are related to the sensor components:

1. **Direction:** the direction of the sensor component in the FE model
2. **ID:** An ID to number the sensor component

The definition of the complete set of properties for the sensor type is still ongoing. However suggested properties are:

1. **Sensor Type:** Strain gage / Accelerometer / Displacement Sensor
2. **Sensor Description:** Further description of the sensor type
3. **Manufacturer:** The name of the manufacturer
4. **Model name:** The model name of the sensor type
5. **Number of sensor components:** a number specifying the number of sensor components
6. **Angles:** For strain gages, a set of angles defining the angles between each sensor components

All these properties typically originate from different input sources, but are now contained within the same AP209 model and this facilitates the storing, organizing and sharing of the complete data set. Additional properties are planned to be added in future work to hold a comprehensive description of the sensors.

Properties that relates to the sensors, but are test case dependent are defined differently. For example filtering techniques performed on the data by the DAQ System (Data Acquisition system) are not necessarily the same for every usage of the sensor. These properties are related directly to the result data which we cover in the next section.

An example of how the sensor data structure can look in a STEP model is shown in Fig. 9. Note that the *reference element* property of the sensor assembly is a direct link to the actual element in the FE model, providing traceability between analysis and testing in the same model.

5. Structural tests

In STEP the generic entity **action** will be used to represent *the action of performing a structural test*. The items used in the test are assigned to this entity by a **applied_action_assignment**, which in turn assigns each item a role of *input* or *output* to the **action**. The *input* items to the test are the sensors and the tested part, while the *output* is the sensor result data for that particular test.

The **action_method** is the link to the description of how the test was performed. This could be in the form of a reference to a certain external document, or in a more structured form with STEP entities. The work

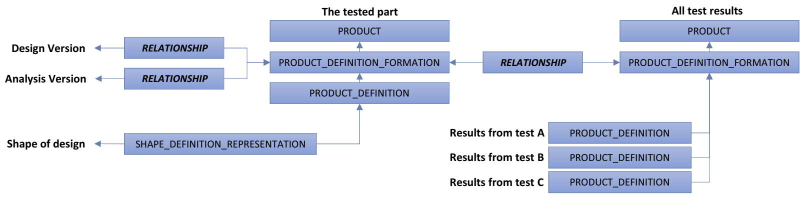


Fig. 8. Relations between FEA, Design, Tested Part and Test Results. In this case, there are three individual tests.

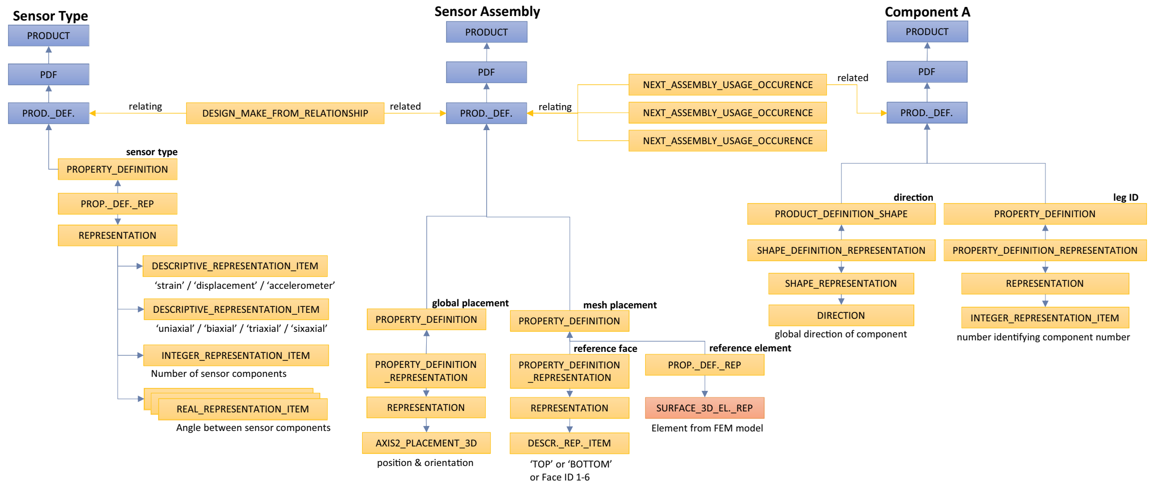

Sensor Type
PRODUCT
PDF
PROD._DEF.
relating
DESIGN_MAKE_FROM_RELATIONSHIP
related
Sensor Assembly
PRODUCT
PDF
PROD._DEF.
relating
NEXT_ASSEMBLY_USAGE_OCCURENCE
NEXT_ASSEMBLY_USAGE_OCCURENCE
NEXT_ASSEMBLY_USAGE_OCCURENCE
related
Component A
PRODUCT
PDF
PROD._DEF.
sensor type
PROPERTY_DEFINITION
PROP._DEF._REP
REPRESENTATION
DESCRIPTIVE_REPRESENTATION_ITEM
'strain' / 'displacement' / 'accelerometer'
DESCRIPTIVE_REPRESENTATION_ITEM
'uniaxial' / 'biaxial' / 'triaxial' / 'sixaxial'
INTEGER_REPRESENTATION_ITEM
Number of sensor components
REAL_REPRESENTATION_ITEM
Angle between sensor components
global placement
PROPERTY_DEFINITION
PROPERTY_DEFINITION _REPRESENTATION
REPRESENTATION
AXIS2_PLACEMENT_3D
position & orientation
mesh placement
PROPERTY_DEFINITION
reference face
PROPERTY_DEFINITION _REPRESENTATION
REPRESENTATION
DESCR._REP._ITEM
'TOP' or 'BOTTOM' or Face ID 1-6
reference element
PROP._DEF._REP
SURFACE_3D_EL._REP
Element from FEM model
direction
PRODUCT_DEFINITION_SHAPE
SHAPE_DEFINITION_REPRESENTATION
SHAPE_REPRESENTATION
DIRECTION
global direction of component
leg ID
PROPERTY_DEFINITION
PROPERTY_DEFINITION_REPRESENTATION
REPRESENTATION
INTEGER_REPRESENTATION_ITEM
number identifying component number

Fig. 9. Example of data structure for sensor with three sensor components (only one is shown).

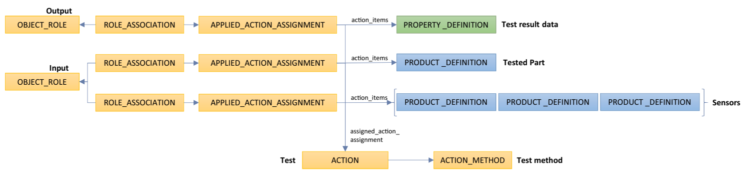


Fig. 10. Data structure for a test performed on a part with three sensors, resulting in a certain test result.

related to this is ongoing and is not presented in this paper.

6. Structural Test data

6.1. Structural test results

The original test result data coming from the test equipment software will typically be in the form of Excel files or other proprietary formats. The data can be extremely large, and it is generally expected that it has been filtered before being converted or imported to this STEP model.

The storing of test data in STEP is based on Part 50 *Mathematical constructs* [21]. The entity **listed_real_data** holds the values, but the complexity of this portion of the STEP standard requires multiple other entities to define what kind of data is held within it. The details of this data model are outside the scope of this paper, but readers are encouraged to review the Part 50 documentation. For simplicity we will define the entity ***data_array*** to represent an array of values. The information within this *entity* is an array of result data corresponding to the data output from *one* sensor component for *one* test case, the type of data (i.e. strain or displacement) and the size of the data. The**data_array** relates to a **property_definition** allowing us to use the result data as a property to other entities.

As seen in Fig. 11, relationships are used to group the **property_definition** and ***data_array*** results from each sensor component to **property_definition**s corresponding to the whole test case. The test case **property_definition**s reference the corresponding **product_definition** of the output data. These **property_definition**s are the same as those labeled as output of the **action** in Fig. 10.

Fig. 12is a combination of both Figs. 10 and 11 showing the overall relationship between the individual sensor results and the test actions.

6.2. Structural test result properties

In addition to the sensor properties presented in Section 4, we will now look at properties that are related to sensors, but that may vary for each test case. They are typically properties originating from the DAQ equipment and software used for retrieving test data.

This applies to properties such as:

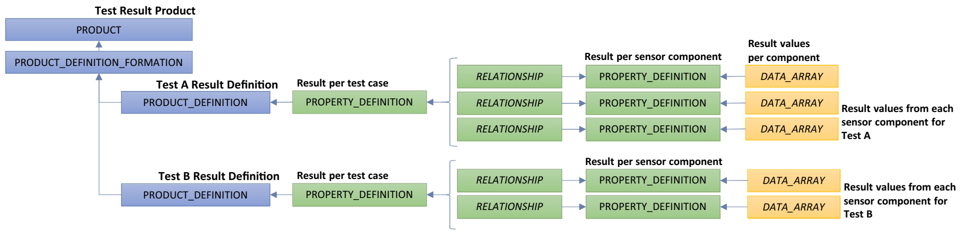


Fig. 11. Example of data structure showing the relation between the sensor result values and the output result data **product**.

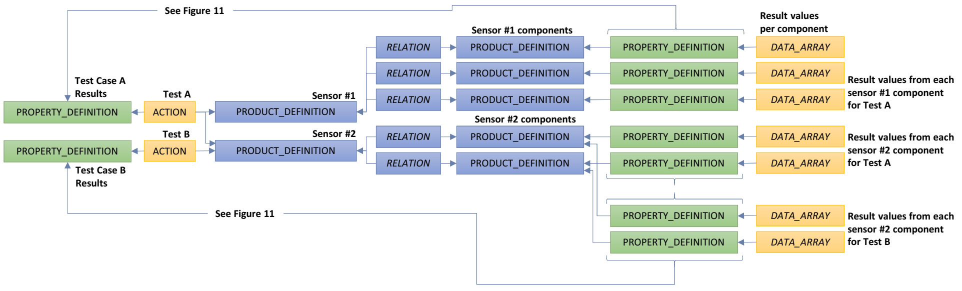


Fig. 12. Data structure showing relation between sensor results, sensors and tests. Here we have two tests and 2 sensors. One of the sensors (sensor 2) is used in both tests.

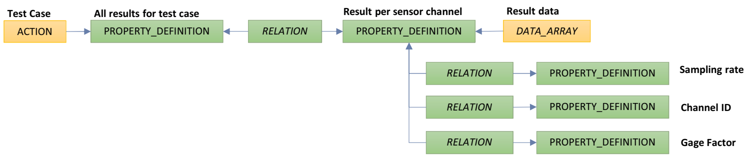


Fig. 13. The data values array references the result property of sensor a component, which is also referenced by properties that are unique for this sensor channel and test case.

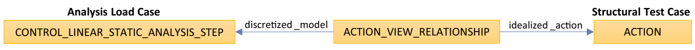


Fig. 14. Load cases from the Analysis data set are related to the Test Case of the Structural Test data set via the action_-view_relationship.

1. The channel ID from the test Equipment
2. Filtering Techniques
3. Sample Rate
4. Scaling
5. Gage Factor

The use of relationships between **property_definition**s is again used to include these test case dependent properties. This is illustrated in Fig. 13.

6.3. *Structural test relation to analysis load case*

In Section 3.2 we presented how load cases are defined in an Analysis Model. The test cases are related to load cases via the **action_-view_relationship** entity. This entity relates a *discretized model* (the analysis load case) to an *idealized action* (the test **action** that is being idealized) (Fig. 14).

7. Model development methodology

As AP209 is primarily meant to store and share CAD and simulation data, structural testing was not part of its original design. The first question when investigating the use of this standard for another domain such as structural testing, was if the standard itself required an extension: Does additional entities and types need to be added to the AP209 schema?

To answer the above, a careful examination of the AP209 schema was performed to get a detailed overview of which type of data the data model *can* represent. A good understanding of the whole schema was acquired during the development of a converter to translate FEM analyses in Nastran format to AP209.

The next step was to define which type of data from the structural testing domain needed to be included in the data model. These data types were then mapped to AP209 elements (entities, attributes, data types etc.). Careful attention was given to how to relate this domain to the analysis elements.

As noted previously, many of the STEP elements are generic, and can be used to represent a wide variety of data. However, the pre- and post-processors need to know how to interpret these generic constructs. An example is the entity **action**, a generic item, but with certain attributes to specify what the *action* represents (here, used to define the test case). This is where the *Application Reference Model* (as briefly discussed in 2.2) and documents such as Recommended Practices are required to specify the data model semantics.

The standard itself contains the formal description of every STEP element, while the Recommended Practices describe *how* it is intended to be used and implemented in applications. Such an implementors' guide is currently being developed for the testing domain to formally describe all the details presented here. In addition, to properly and formally introduce the results of this paper to the STEP standard, and make it available to the structural testing community, the AAMs and ARMs and their mapping to the AIM need to be developed. This would possibly also involve the introduction of new entity subtypes specifically for the domain of structural testing. The ARM shall include the concepts that are specific to the structural testing domain; they shall be mapped to the STEP resources as described in this paper. But for the purpose of this initial study, no extension of the standard is required.

After the mapping was defined from the test data to AP209 entities, another converter was created. This converter uses the results from structural tests in .csv format as well as files defining the sensors and test cases as input. The converter directly creates STEP data in an AP209 database (using Jotne's tools EDMS [23] and EDMopenSimDM [24]). The analysis related to the test case already resides within the database, allowing the converter to access it and create direct links between the new structural test data population and the analysis model.

The Simulation Data Management use case discussed in the introduction would utilize this converter function to construct a complete view of the product. A prototype is being performed to validate the usage of the model. An airplane winglet has been designed, simulated, manufactured and tested to imitate the different phases of product development. The data of each phase has been either exported or converted to STEP AP209 and imported to the EDMopenSimDM

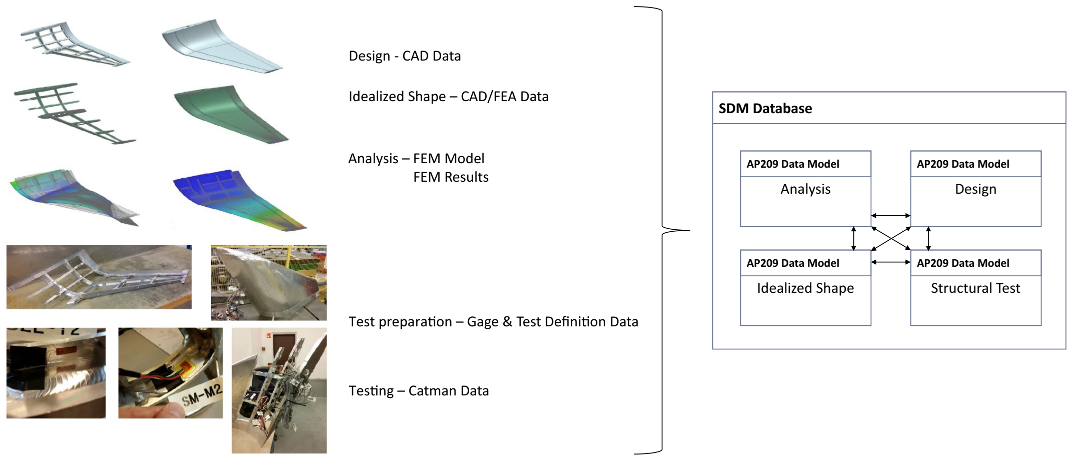


Fig. 15. Different AP209 data sets imported to the SDM tool.

application (Figure 15). This application uses AP209 as its database schema. The tool is being further developed to let the user access and manage the federated data.

8. Conclusion

We have now shown how the structural test related data can be represented in the AP209 data model, and how the relevant pieces of data can be connected to an engineering analysis and its results.

The purpose of SDM software is to manage and provide an overview of all the information related to simulations and give quick access to specific data. Having all the different aspects of the product in a consistent schema in a single database enables exactly this. If implemented properly, it enables the enterprise to utilize this data without having to open files in their original software.

Accessing information can easily be done by executing simple query functions on the single consistent and comprehensive data set. Examples of queries could be, retrieving the type of sensor used, the location of it on the mesh, getting the result data from a particular sensor for a particular test, and comparing it to the corresponding analysis results. Different views on the AP209 population can be implemented, such as an overview of all sensors that were used in a specific test, and their result values in both analysis and testing.

Besides representing contents data of analyses and structural tests, AP209 also provides the resources for data management. This includes defining who created a model, who accepted it, deadlines, tasks to be performed etc. These specifications can be directly linked to the entities that describes the analysis and structural test contents within the data sets.

The complete data set can then be exported to ASCII or binary STEP files that are compliant with the LOTAR (Long Term Archiving and Retrieval) specifications [22]. The resulting information can be shared with other systems conforming to the these standards, which enables project information to be archived or retrieved with all data still being traceable. To make the results of this study available to the structural testing community, AP209 should be updated and published as a new edition. The STEP resources seem to be sufficient to capture the information requirements discussed in this paper, but the AAM, the ARM and the mapping specification will need to be updated.

References

[1] 1994. ISO 10303-1:1994 Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles. Geneva (Switzerland): International Organization for Standardization (ISO).
[2] ISO 10303-209:2014 industrial automation systems and integration – product data representation and exchange – part 209: application protocol: multidisciplinary analysis and design. geneva (switzerland): international organization for standardization (ISO). 2014a.
[3] FAA-21-48 - using electronic modeling systems as primary type design data, u.s. department of transportation, federal aviation administration. 2010.
[4] ISO 10303-11:2004 industrial automation systems and integration – product data representation and exchange – part 11: description methods: the EXPRESS language reference manual. geneva (switzerland): international organization for standardization (ISO). 2004.
[5] ISO 10303-21:2016 industrial automation systems and integration – product data representation and exchange – part 21: implementation methods: clear text encoding of the exchange structure. geneva (switzerland): international organization for standardization (ISO). 2016.
[6] ISO 10303-22:1998 industrial automation systems and integration – product data representation and exchange – part 22: implementation methods: standard data access interface. geneva (switzerland): international organization for standardization (ISO). 1998.
[7] ISO 10303-23:2000 industrial automation systems and integration – product data representation and exchange – part 23: implementation methods: C++ language binding to the standard data access interface. geneva (switzerland): international organization for standardization (ISO). 2000a.
[8] Kramer T, Xu X. STEP In a nutshell, in: advanced design and manufacturing based on STEP. London: Springer - Verlag London Ltd.; 2009.
[9] Anonymous. STEP application handbook ISO 10303 version 3, SCRA, north charleston, south carolina, USA. 2006.
[10] ISO 10303-203:2011 industrial automation systems and integration – product data representation and exchange – part 203: application protocol: configuration controlled 3d designs of mechanical parts and assemblies. geneva (switzerland): international organization for standardization (ISO). 2011a.
[11] ISO 10303-2010:2001 industrial automation systems and integration – product data representation and exchange – part 214: application protocol: Core data for automotive mechanical design processes. geneva (switzerland): international organization for standardization (ISO). 2010.
[12] ISO 10303-43:2011 industrial automation systems and integration – product data representation and exchange – part 43: Integrated generic resource: representation structures. geneva (switzerland): international organization for standardization (ISO). 2011b.
[13] ISO 10303-42:2014 industrial automation systems and integration – product data representation and exchange – part 42: integrated generic resource: Geometric and topological representation. geneva (switzerland): international organization for standardization (ISO). 2014b.
[14] ISO 10303-104:2000 industrial automation systems and integration – product data representation and exchange – part 104: integrated application resource: Finite element analysis. geneva (switzerland): international organization for standardization (ISO). 2000b.
[15] ISO 10303-242:2014 industrial automation systems and integration – product data representation and exchange – part 242: application protocol: managed model-based 3d engineering. geneva (switzerland): international organization for standardization (ISO). 2014c.
[16] Hunten KA. CAD/FEA Integration with STEP AP209 technology and implementation. MSC aerospace users conference proceedings. 1997. http://web.mscsoftware.com/support/library/conf/auc97/p01297.pdf[accessed 18 january 2018]
[17] PDES Inc., ISO 10303-209 "Multidisciplinary Analysis and Design", http://www.ap209.org/introduction (accessed on 16 January 2018).
[18] Hunten KA, Feeney AB, Srinivasan V. Recent advances in sharing standardized STEP

composite structure design and manufacturing information. Comput.-Aided Des. 2013;45:1215–21.
[19] LOTAR (long term archiving and retrieval). LOTAR EAS: engingeering analysis & simulation workgroup. http://www.lotar-international.org/lotar-workgroups/engineering-analysis-simulation/scope-activities.html(accessed on 19 January 2018).
[20] Hunten KA. Recommended practices for AP209ed2 10303-209:2014. CAx implementor forum. 2016.
[21] ISO 10303-50:2002 industrial automation systems and integration – product data representation and exchange – part 50: integrated generic resource: mathematical constructs. geneva (switzerland): international organization for standardization (ISO). 2002.
[22] NAS9300-001, long term archiving and retrieval of digital technical product documentation such as 3d, CAD and PDM data : part 101: structure, aerospace industries association of america inc.2017.
[23] EDMS (version 2.100.15) [STEP Data Manager software], Jotne EPM Technology AS.
[24] EDMopenSimDM (version 14.0) [Simulation Data Manager software], Jotne EPM Technology AS.

Remi Lanza completed his in M.Sc. in Mechanical Engineering in 2015 within the field of finite element analysis. Remi later acquired an interest for computer science, and joined Jotne EPM Technology in 2016 where he started his industrial Ph.D., which is a collaboration project between the Norwegian University of Science and Technology (NTNU) and Jotne. While pursuing his Ph.D., Remi has been exposed to several of the STEP ISO 10303 standards, especially STEP AP209. His initial research involves the usage of AP209 to facilitate data management and retention of simulation data and structural test data.

Jochen Haenisch leads the Aeronautics, Defence and Space business area in Jotne EPM Technology. He has contributed to and managed many implementations of the Jotne data interoperability tool EXPRESS Data Manager. These applied various STEP standards including ISO 10303-209 (Multidisciplinary analysis and design), ISO 10303-214 (automotive), ISO 10303-239 (product lifecycle support, PLCS) and ISO 15926 (oil&gas). In 1990 he entered into the ISO Subcommittee for Industrial Data, ISO/TC 184/SC 4, for many known as STEP. He regularly attends their plenary meetings as head of delegation for Norway. Currently he is deputy convenor of WG12, Common Resources.

Mr. Kjell Bengtsson, is a Vice President at Jotne, has a Mechanical Engineering background and a diploma in Marketing. He started out at Volvo Car and General Electric doing CAD/DB applications and later management positions, and is now VP at Jotne EPM Technology. Kjell has been exposed to STEP, PLCS and other related standards for the last 25 years and is actively involved in neutral database implementation projects in the most complex defense and aerospace sector projects. Kjell is a Member of the Board of PDES, Inc and supports other industry organizations like AIA/ASD, NIAG (NATO), FSI and more.

Prof. Rølvåg was born in Mo I Rana, 16/10-1963. Rølvåg holds a M.Sc. and a Ph.D. within finite element dynamics of elastic mechanisms and control from NTH. His publications are mainly within non-linear finite element dynamics and active damping of elastic mechanisms. He has been central in developing FEDEM, a finite element based modeling and simulation tool with multidisciplinary capabilities (see www.fedem.com). He has also established several engineering companies and optimized products for the automotive, offshore and aerospace industries. Prof. Rølvås research interests cover computer science applied for engineering applications focusing on simulation of behavior and strength of electromechanical products.

A.2 Paper 2 - ISO 10303 AP209 - Why and how to embed nonlinear FEA

This article is awaiting submission and is therefore not included.

A.3 Paper 3 - Extending STEP AP209 for Nonlinear Finite Element Analysis

This article is awaiting submission and is therefore not included.

Appendix B

Secondary publication

B.1 Paper 4 - Open Simulation Data Management and Testing - The CRYSTAL Project

This paper is not included due to copyright restrictions.

Appendix C

Contact and Nonlinear Materials in STEP

The following presents a topic which was originally intended as a paper, but is instead included in this theses as its own appendix.

Paper 3, which implements some of the proposed AP209 extensions recommended from Paper 2, covers nonlinear analysis and analysis parameters. This appendix, further recommends extensions to the AP209 standard, for covering FEM contact interactions, and certain nonlinear material models.

Section C.1 presents a set of finite element test cases using contact interaction and nonlinear material models. The converter mentioned in 3.1.1 was extended to cover the AP209 extensions, and is used to convert the test cases to other formats. The original models were created in Abaqus, then converted to AP209, and translated to Nastran and Ansys formats. All models were solved in their respective solvers. A detailed discussion on the results of these analyses is not included, however, visualizations of the results are presented.

Section C.2, and section C.3, covers some details on the contact and nonlinear material implementations.

C.1 Test Cases

Test Case A

The first test case is based on a NAFEMS benchmark test case, *Benchmark 2: 3D Punch (Rounded Edges)* described in [56].

- **Geometry**
  - Top cylinder diameter: 100mm
  - Top cylinder height: 100mm
  - Top cylinder fillet radius: 10mm
  - Bottom cylinder diameter: 200mm
  - Bottom cylinder height: 200mm
- **Mesh**
  - Using symmetry, only a quarter is meshed
  - Top cylinder: 1960 elements
  - Bottom cylinder: 16100 elements
  - All elements are of type linear hexahedral (Abaqus: `C3D8R`)
- **Time step incrementation**
  - Automatic time stepping
  - Max incr.: 100
  - Initial incr. size: 0.01
  - Min. incr. size: $1.0E-5$
  - Max. incr. size: 0.01

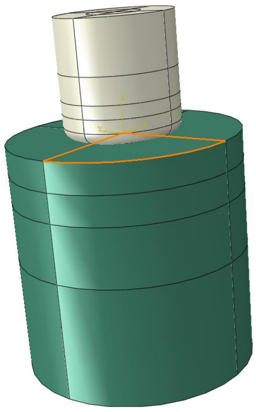


Figure C.1: Test case A geometry

- **Material**
  - Top cylinder: $E = 210GPa$ and $\nu = 0.3$
  - Bottom cylinder: $E = 70GPa$ and $\nu = 0.3$
- **Contact interaction**
  - Contact defined on element surfaces, between bottom face of top cylinder, and top face of bottom cylinder.
  - Penalty friction formulation method with friction coefficient $= 0.1$ and elastic slip $= 0.005$
- **Boundary conditions and load**
  - Bottom surface of bottom cylinder has fixed translation.
  - Constraints to simulate symmetry.
  - Distributed pressure load on the top surface of the top cylinder with 100MPa.

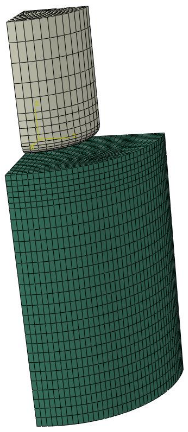


Figure C.2: Test case A mesh

Figure C.3 shows the resulting deformation for the different solvers at the last time step. The graphs in Figure C.4 shows the Z-displacement on the nodes along a path going from the center of the top surface of the bottom cylinder, to the outer edge of the cylinder.

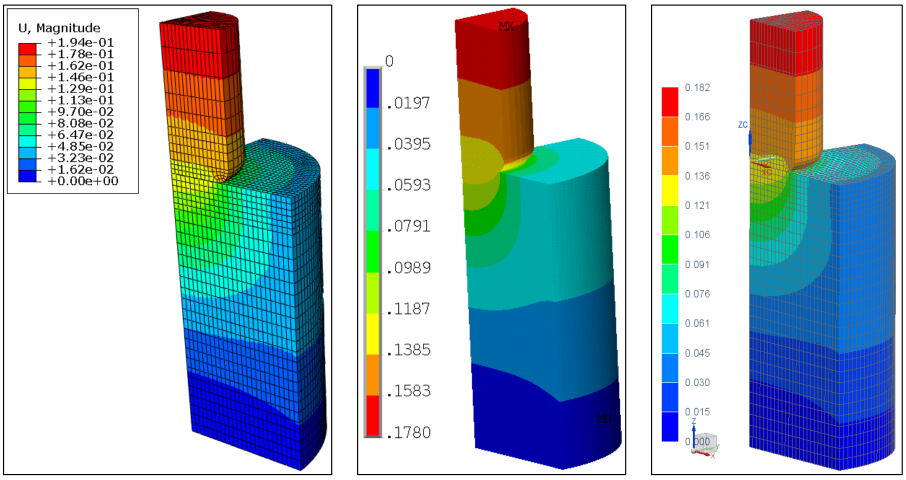


Figure C.3: Test case A displacements visualization (Abaqus / Ansys / Nastran)

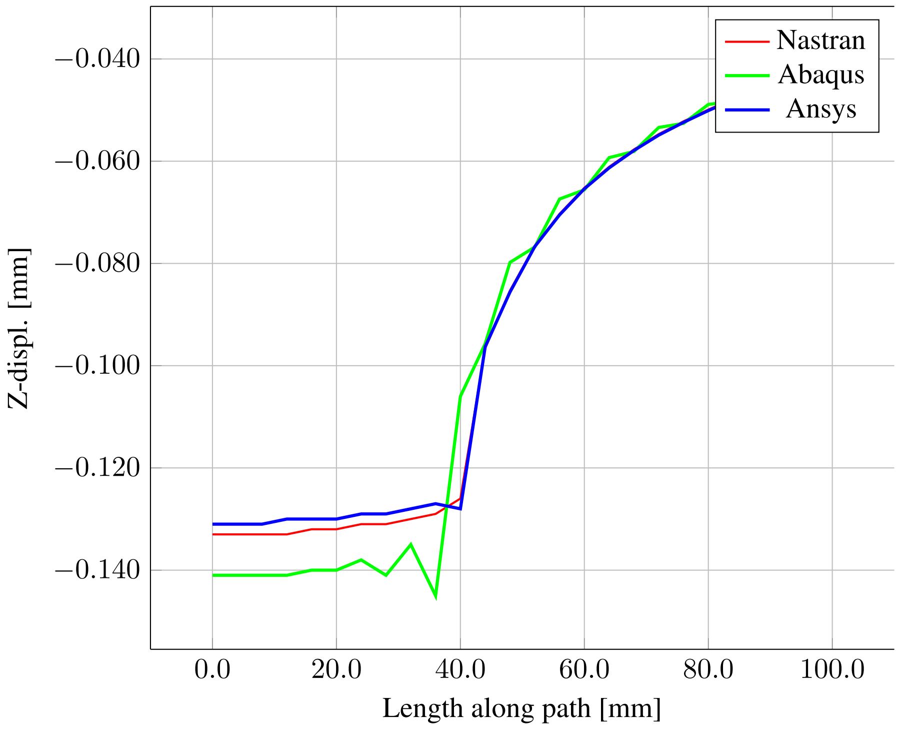


Figure C.4: Z-displacement, along path on top of bottom cylinder, from center to edge

Test Case B

This test case is also based on a NAFEMS benchmark test case, *the three bar problem* described in [57].

- **Geometry**
  - Three bars with dimensions 10x10x100 (mm)
  - Center to center spacing = $30mm$
- **Mesh**
  - Each bar composed of 80 elements (2x2x20)
  - All elements are of type linear hexahedral (Abaqus: `C3D8R`)
- **Time step incrementation**
  - Automatic time stepping
  - Max incr.: 100
  - Initial incr. size: 0.01
  - Min. incr. size: $1.0E-5$
  - Max. incr. size: 0.01
- **Material**
  - All bars: $E = 210GPa$ and $\nu = 0.3$
  - Bar left: $\sigma_{yield} = 200MPa$
  - Bar center: $\sigma_{yield} = 300MPa$
  - Bar right: $\sigma_{yield} = 400MPa$
- **Boundary conditions and load**
  - Nodes on top faces of bars are fixed in all translation degrees of freedom.
  - Nodes on bottom faces of bars connected to reference point with linear equations.
  - Forced displacement downwards on reference point from 0 to $0.275mm$

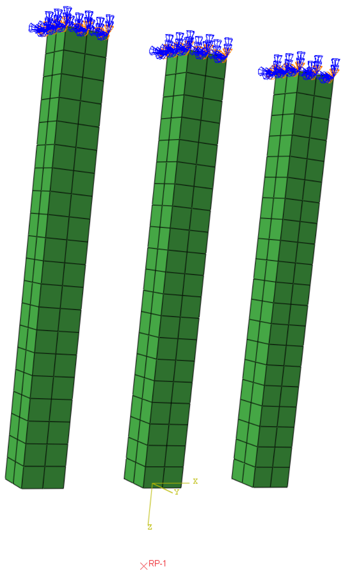


Figure C.5: Test case B

Solving the analyses, results in the first bar reaching yield at 200MPa, second at 300MPa, and last at 400MPa. All three solvers agree on these results.

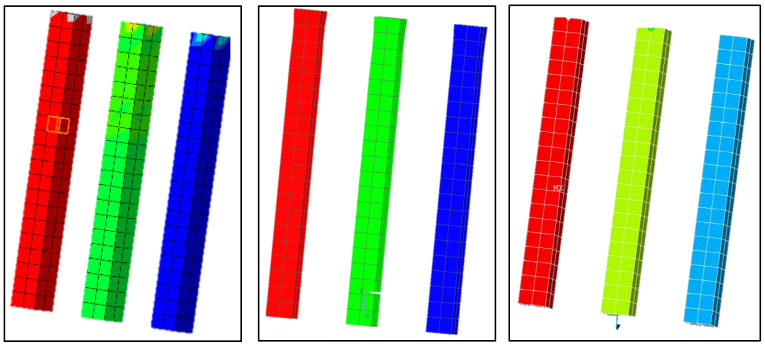


Figure C.6: Test case B Von-Misses visualization, red = 400MPa, green = 300MPa, blue = 200MPa (Nastran / Abaqus / Ansys)

Test Case C

- **Mesh**
  1000 shell elements of type S4R with thickness 0.175mm.
- **Material**
  - $E = 50GPa$ and $\nu = 0.3$
  - Multilinear plasticity model, as seen in Figure C.8

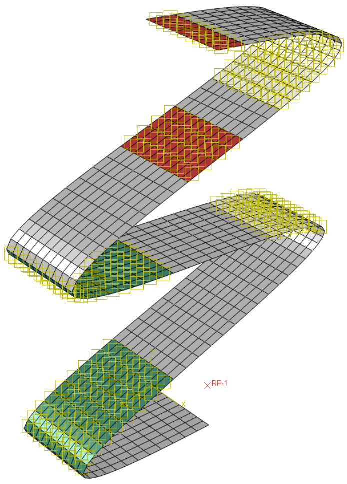


Figure C.7: Test case C

- **Time step incrementation**
  - Automatic time stepping
  - Max incr.: 400
  - Initial incr. size: 0.01
  - Min. incr. size: 10^{-5}
  - Max. incr. size: 0.1
- **Boundary conditions and load**
  - Nodes on one long side is fixed in X.
  - Nodes on one short side is fixed all dofs.
  - Forced displacement along one short side in Y; from $0.0mm$ to $-17.5mm$ to $17.5mm$ to $-17.5mm$ and back to $0.0mm$.
- **Contact interactions**
  - Three surface regions (as seen with colors on Figure C.7) have self contact defined.
  - Penalty friction formulation method with friction coefficient = 0.1 and elastic slip = 0.005

Figure C.9 shows a visualization of the resulting deformations in the three solvers.

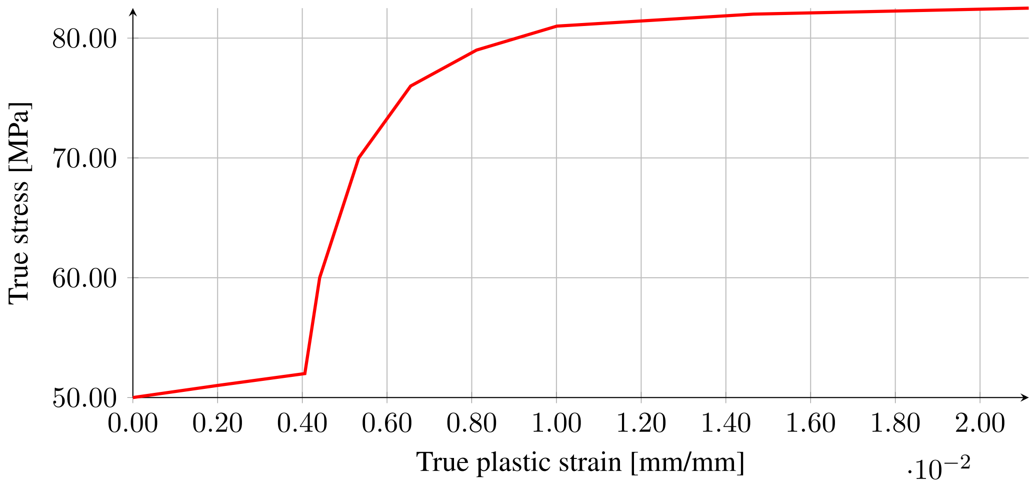


Figure C.8: Test case C, multilinear material model

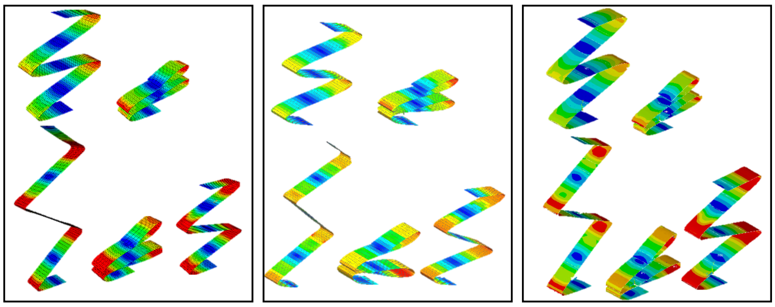


Figure C.9: Test case C Von-Misses visualization (Abaqus / Nastran / Ansys)

Figure C.10 shows the von Mises stress and the plastic strain, on a path defined by the center nodes along the long edge of the mesh, at the last time step of the analysis. The results are following the same path, with the exceptions of some peaks. A detailed discussions of these differences is not included.

C.2 Contact Interaction

The first concept required by AP209 to support contact, is the definition of surface and edge regions. In this chapter we focus on surface contact region, yet the same concepts are applicable for edge regions.

AP209 supports the definitions of element groups, however, it does not relate the group to any surface or edge identifier. We extended the entity **element_group**, by adding the subtype **element_aspect_group**. The already existing data type ***element_aspect***, can represent the aspect of an element; edge, face, and volume. This new entity has an attribute *aspect* which can hold an ***element_aspect*** value. With this, a surface region can

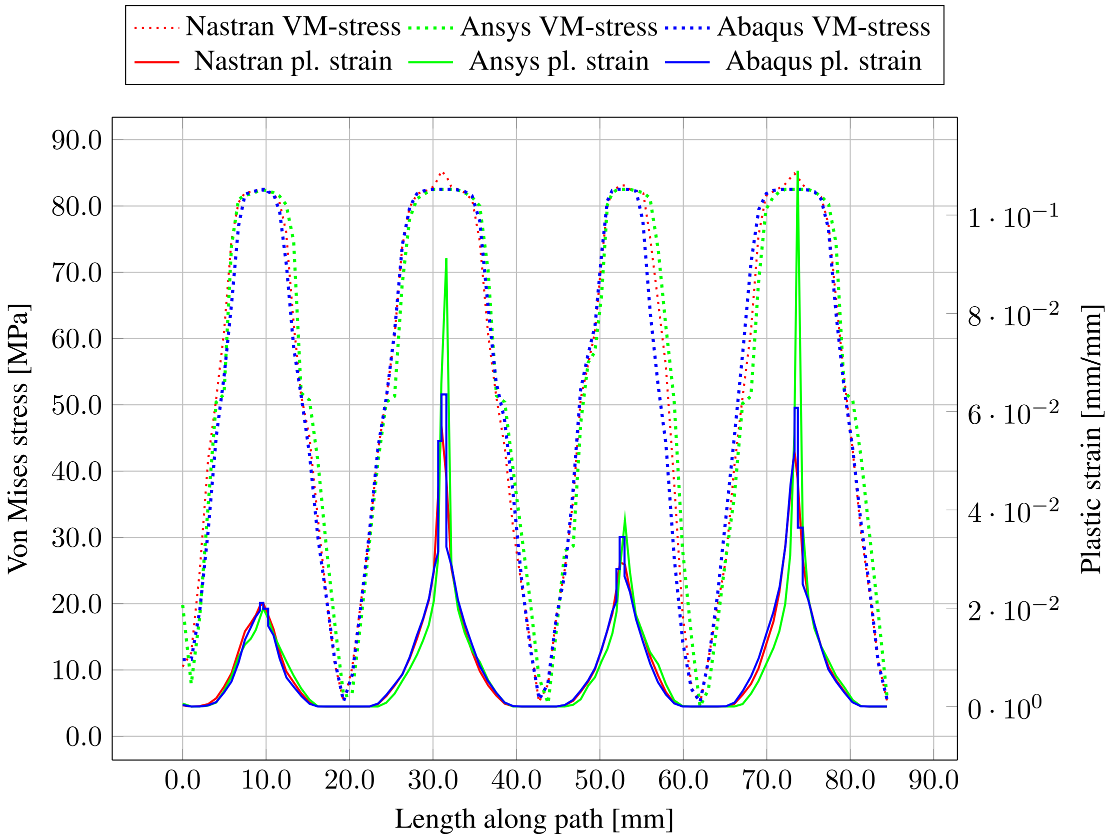


Figure C.10: Von Mises stress and plastic strain

be represented by referencing a set of elements, and the face identification of the elements belonging to the surface, for surface and volume elements.

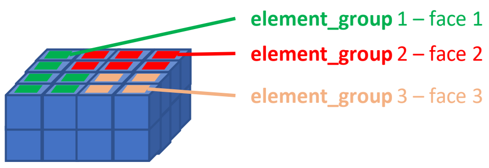


Figure C.11: Element groups and faces

The EXPRESS code in D.1, shows the definition of **element_aspect_group.** Note `WR1`, a *where rule*, which upon model validation, will check that there is a consistency between the element type and the selected aspect. For example, a face is not allowed on a 1D element. The code of the function used in this *where rule* is left out for simplicity.

```

ENTITY element_aspect_group
   ABSTRACT SUPERTYPE OF(ONEOF(
      volume_3d_element_aspect_group,
      volume_2d_element_aspect_group,
          surface_3d_element_aspect_group,
          surface_2d_element_aspect_group
   ))
   SUBTYPE OF (element_group);
   aspect : element_aspect;
WHERE
   WR1 : consistent_elements_aspect(aspect, elements);
END_ENTITY;
```

Listing C.1: EXPRESS code of **element_group**

element_group is already a supertype of element specific groups; volume 3D, volume 2D, surface 3D, etc. These entities hold *where rules* that restricts the element types allowed in the group. Additional entities were added, which inherits from both existing element type groups, and the abstract supertype **element_aspect_group**. An *abstract* entity, is an entity that is only allowed to be instantiated through one of its non-abstract subtypes. Figure C.12 shows all new implemented entities.

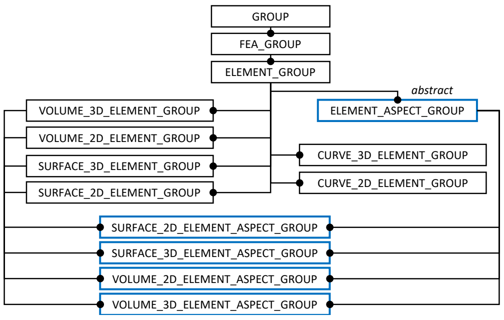


Figure C.12: Element group entities (In blue; new entities)

Now, by using the element specific aspect groups entities, validation rules, if checked, will ensure that the appropriate element types and element aspects are used.
To allow for a region with different face IDs across its belonging elements, multiple groups can be defined, and related with the already existing entity **element_group_relationship**.
Further, parameters for the contact interactions are required to be represented. In AP209, the entity **state_definition** is used for everything that is load and boundary condition related through a set of different subtypes. A natural place to add support for contact interactions is as new subtypes of **state_definition**. Multiple subtypes were added to not only support contact interaction, but also *glue* interaction, and potential other types of

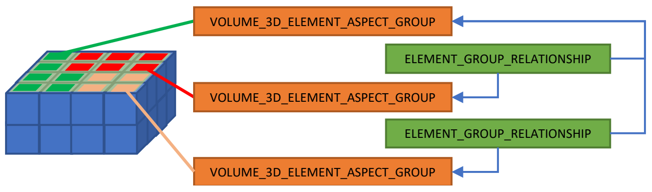

VOLUME_3D_ELEMENT_ASPECT_GROUP
ELEMENT_GROUP_RELATIONSHIP
VOLUME_3D_ELEMENT_ASPECT_GROUP
ELEMENT_GROUP_RELATIONSHIP
VOLUME_3D_ELEMENT_ASPECT_GROUP

Figure C.13: Instance diagram; Relating multiple sub-regions with different face IDs.

interactions between mesh regions. An entity **fea_interaction** was added as a subtype of **state_definition**. Furthermore, a hierachy of new entities where added as subtypes of **fea_interaction**, as seen Figure C.14.

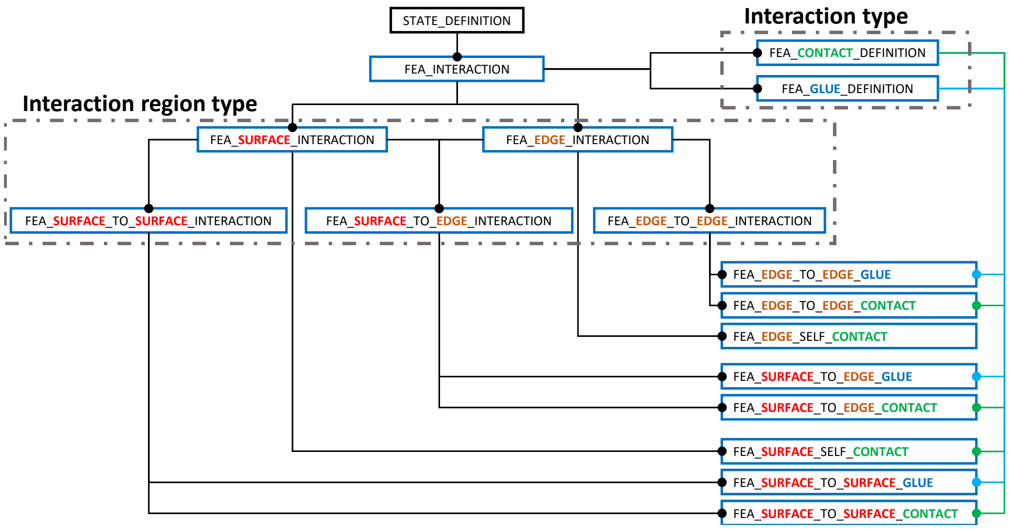


Figure C.14: Hierachy of new FEA interaction entities. Text colors are added for clarity

The **fea_contact_definition** and **fea_glue_definition**, are used to define an interaction as contact or gluing. Possibly other interaction types could eventually be implemented in the future, and follow the same structure. They hold no attributes, but are used to collect the specific parameters describing the interaction properties used in the solver.
For each different type of region interaction; single surface, surface to surface, single edge, edge to edge, and surface to edge, there exists an entity inheriting from **fea_interaction**, allowing for both self interaction, and interaction across regions. **fea_surface_interaction** and **fea_edge_interaction** has one attribute referencing one region (**fea_group**) as a *master* region. The *surface to surface*, *edge to edge*, and *surface to edge*, adds a second attribute, referencing a *slave region*. Each of these hold *where rules*

which restrict the type of **fea_group** that may be referenced, and which element aspect the groups have. For example, **surface_interaction** enforces a *where rule* which specifies that the referenced **fea_group**, has to either be a subtype of **element_aspect_group** with aspect set to a face, or a **node_group** (a surface can in certain solvers be defined by a set of nodes).
The express code for the new entities can be found in Appendix D.

C.3 Nonlinear Materials

Support for nonlinear materials in AP209, is not documented, however, materials in STEP are supported in a very generic way. There are specific entities and types for linear material properties such as E-modulus, poisson ratio, density, and a few others. These are specializations of generic material properties.
In this study, we didn't attempt to cover all nonlinear material concepts, but focused on two simple models; perfectly plastic, and multilinear material model.
Figure C.15 shows an example of instances specifying a FEA material. In this case, only the E-modulus and poisson ratio are specified for the linear analysis.

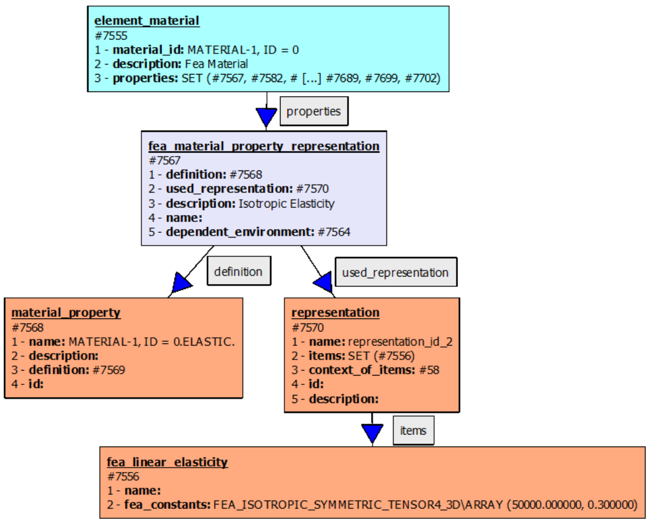

element_material
#7555
1 - material_id: MATERIAL-1, ID = 0
2 - description: Fea Material
3 - properties: SET (#7567, #7582, # [...] #7689, #7699, #7702)
properties
fea_material_property_representation
#7567
1 - definition: #7568
2 - used_representation: #7570
3 - description: Isotropic Elasticity
4 - name:
5 - dependent_environment: #7564
definition
used_representation
material_property
#7568
1 - name: MATERIAL-1, ID = 0.ELASTIC.
2 - description:
3 - definition: #7569
4 - id:
representation
#7570
1 - name: representation_id_2
2 - items: SET (#7556)
3 - context_of_items: #58
4 - id:
5 - description:
items
fea_linear_elasticity
#7556
1 - name:
2 - fea_constants: FEA_ISOTROPIC_SYMMETRIC_TENSOR4_3D\ARRAY (50000.000000, 0.300000)

Figure C.15: Instance structure specifying a linear material with E-modulus and poisson ratio

A FEA material in STEP is identified by the entity **element_material**. It collects one or more material properties instance structures, which together defines the material properties. For each property, a structure of **fea_material_property_representation**, **material_property**, **representation**, and **representation_item** is needed. In the example, the **representation_item** is a **fea_linear_elasticity**. For nonlinear material properties, such specialized entities do not exist, and the use of more generic entities are used. For a simple perfectly plastic material model, we only need an additional property and value, specifying the yield strength. We recommend that this is done by a similar structure as the above example, but set the value of the yield point in a **measure_representation_item**, as seen in Figure C.16. The value is a stress value, and the *name* can be used to specify this is a *true stress* value, as opposed to *engineering stress*, or other types of stress. To identify the property as a yield point, the *description* attribute of the **fea_material_property_representation**, or the *name* attribute can be used and set to the string identifier `Yield point`.

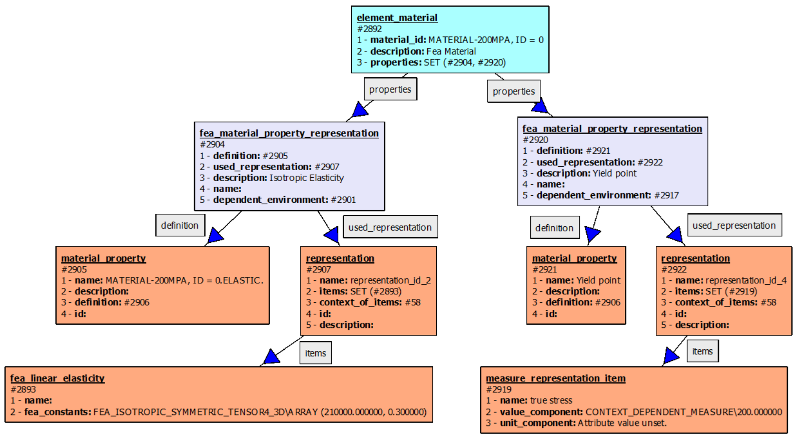

element_material
#2892
1 - material_id: MATERIAL-200MPA, ID = 0
2 - description: Fea Material
3 - properties: SET (#2904, #2920)
properties
properties
fea_material_property_representation
#2904
1 - definition: #2905
2 - used_representation: #2907
3 - description: Isotropic Elasticity
4 - name:
5 - dependent_environment: #2901
fea_material_property_representation
#2920
1 - definition: #2921
2 - used_representation: #2922
3 - description: Yield point
4 - name:
5 - dependent_environment: #2917
definition
used_representation
definition
used_representation
material_property
#2905
1 - name: MATERIAL-200MPA, ID = 0.ELASTIC.
2 - description:
3 - definition: #2906
4 - id:
representation
#2907
1 - name: representation_id_2
2 - items: SET (#2893)
3 - context_of_items: #58
4 - id:
5 - description:
material_property
#2921
1 - name: Yield point
2 - description:
3 - definition: #2906
4 - id:
representation
#2922
1 - name: representation_id_4
2 - items: SET (#2919)
3 - context_of_items: #58
4 - id:
5 - description:
items
items
fea_linear_elasticity
#2893
1 - name:
2 - fea_constants: FEA_ISOTROPIC_SYMMETRIC_TENSOR4_3D\ARRAY (210000.000000, 0.300000)
measure_representation_item
#2919
1 - name: true stress
2 - value_component: CONTEXT_DEPENDENT_MEASURE\200.000000
3 - unit_component: Attribute value unset.

Figure C.16: Instance structure specifying a yield point on a material.

For a multilinear model, which in FEA is commonly specified in a tabular form, still needs to be specified as individual properties of the material. Multilinear models can be specified in different forms, but in this case we use data points of true stress - true plastic strain values. Each value is treated as properties to the associated material as seen in Figure C.17

The values are identified by the same attribute as for the plasticity model, with the string identifier `Plasticity`. This identifier is used, instead of specifying it as a `multilinear plasticity material model`, because in the context of the STEP data model, these values are only properties of the material, which *can be used* to specify a multilinear material model by a system processing the data.

To identify the pairs of values, the entity **property_definition_relationship** is used.

These constructs for specifying data, which in FEA are relatively simple, become rather complex in STEP. This is because STEP data models are intended for relating to multiple domains, and additional data can be associated to most entities. For example, each **fea_material_property_representation**, has a *dependent_environment* attribute, which can be used to specify the environment conditions under which the material values were determined in. The representation of a physical test performed, which was used to establish the material model, could also be related to the values. In addition, documentation,

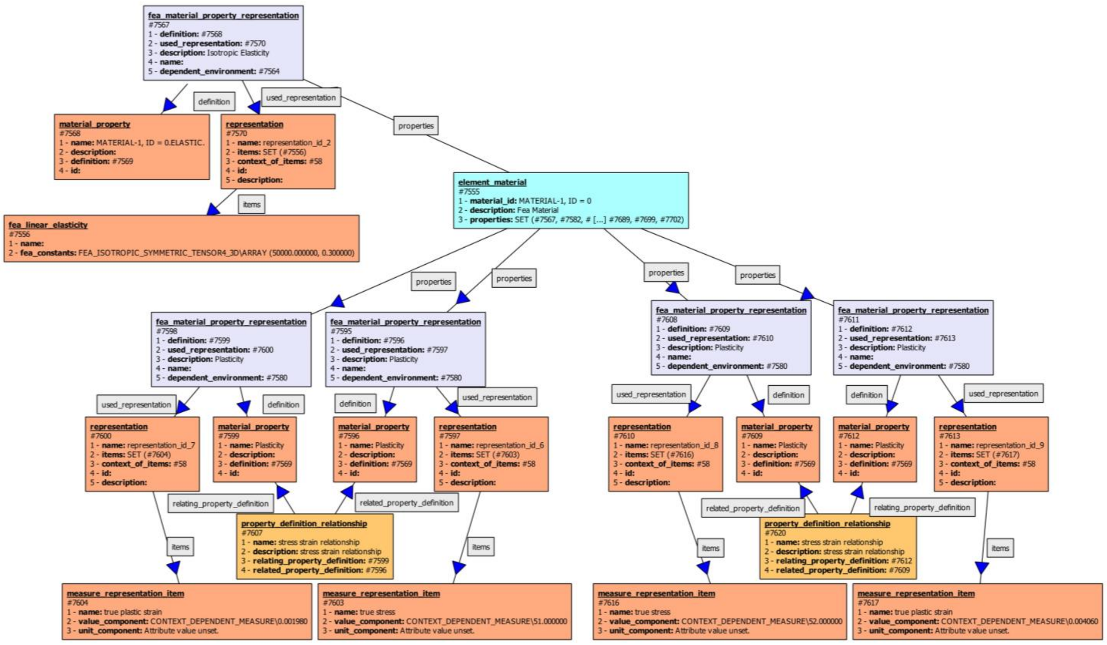

fea_material_property_representation
#7567
1 - definition: #7568
2 - used_representation: #7570
3 - description: Isotropic Elasticity
4 - name:
5 - dependent_environment: #7564
definition
used_representation
properties
material_property
#7568
1 - name: MATERIAL-1, ID = 0.ELASTIC.
2 - description:
3 - definition: #7569
4 - id:
representation
#7570
1 - name: representation_id_2
2 - items: SET (#7556)
3 - context_of_items: #58
4 - id:
5 - description:
items
fea_linear_elasticity
#7556
1 - name:
2 - fea_constants: FEA_ISOTROPIC_SYMMETRIC_TENSOR4_3D\ARRAY (50000.000000, 0.300000)
element_material
#7555
1 - material_id: MATERIAL-1, ID = 0
2 - description: Fea Material
3 - properties: SET (#7567, #7582, # [...] #7689, #7699, #7702)
properties
properties
properties
properties
fea_material_property_representation
#7598
1 - definition: #7599
2 - used_representation: #7600
3 - description: Plasticity
4 - name:
5 - dependent_environment: #7580
fea_material_property_representation
#7595
1 - definition: #7596
2 - used_representation: #7597
3 - description: Plasticity
4 - name:
5 - dependent_environment: #7580
fea_material_property_representation
#7608
1 - definition: #7609
2 - used_representation: #7610
3 - description: Plasticity
4 - name:
5 - dependent_environment: #7580
fea_material_property_representation
#7611
1 - definition: #7612
2 - used_representation: #7613
3 - description: Plasticity
4 - name:
5 - dependent_environment: #7580
used_representation
definition
definition
used_representation
used_representation
definition
definition
used_representation
representation
#7600
1 - name: representation_id_7
2 - items: SET (#7604)
3 - context_of_items: #58
4 - id:
5 - description:
material_property
#7599
1 - name: Plasticity
2 - description:
3 - definition: #7569
4 - id:
material_property
#7596
1 - name: Plasticity
2 - description:
3 - definition: #7569
4 - id:
representation
#7597
1 - name: representation_id_6
2 - items: SET (#7603)
3 - context_of_items: #58
4 - id:
5 - description:
representation
#7610
1 - name: representation_id_8
2 - items: SET (#7616)
3 - context_of_items: #58
4 - id:
5 - description:
material_property
#7609
1 - name: Plasticity
2 - description:
3 - definition: #7569
4 - id:
material_property
#7612
1 - name: Plasticity
2 - description:
3 - definition: #7569
4 - id:
representation
#7613
1 - name: representation_id_9
2 - items: SET (#7617)
3 - context_of_items: #58
4 - id:
5 - description:
relating_property_definition
related_property_definition
related_property_definition
relating_property_definition
property_definition_relationship
#7607
1 - name: stress strain relationship
2 - description: stress strain relationship
3 - relating_property_definition: #7599
4 - related_property_definition: #7596
property_definition_relationship
#7620
1 - name: stress strain relationship
2 - description: stress strain relationship
3 - relating_property_definition: #7612
4 - related_property_definition: #7609
items
items
items
items
measure_representation_item
#7604
1 - name: true plastic strain
2 - value_component: CONTEXT_DEPENDENT_MEASURE\0.001980
3 - unit_component: Attribute value unset.
measure_representation_item
#7603
1 - name: true stress
2 - value_component: CONTEXT_DEPENDENT_MEASURE\51.000000
3 - unit_component: Attribute value unset.
measure_representation_item
#7616
1 - name: true stress
2 - value_component: CONTEXT_DEPENDENT_MEASURE\52.000000
3 - unit_component: Attribute value unset.
measure_representation_item
#7617
1 - name: true plastic strain
2 - value_component: CONTEXT_DEPENDENT_MEASURE\0.004060
3 - unit_component: Attribute value unset.

Figure C.17: Instance structure specifying a multilinear material model.

pictures, and other metadata, could be attached using STEP data.
When AP209 is used for a simple translation of a single FEM analysis, all of this additional information is not required. However, in the context PLM and SDM, and storing the data for long-term archiving, these features becomes crucial.

Appendix D

fea_interaction EXPRESS code

```

ENTITY fea_interaction
   SUBTYPE OF (state_definition);
   name           : label;
   description    : OPTIONAL text;
END_ENTITY;

ENTITY fea_surface_interaction
   SUBTYPE OF (fea_interaction);
   master_face_region : fea_group;
(*WHERE:
    Master region is ELEMENT_ASPECT_GROUP with
    face type aspect or NODE_GROUP*)
END_ENTITY;

ENTITY fea_edge_interaction
   SUBTYPE OF (fea_interaction);
   master_edge_region : fea_group;
(*WHERE:
Master region is ELEMENT_ASPECT_GROUP with
edge type aspect or CURVE_3D_ELEMENT_GROUP or
CURVE_2D_ELEMENT_GROUP or NODE_GROUP*)
END_ENTITY;

ENTITY fea_surface_to_edge_interaction
   SUBTYPE OF (fea_surface_interaction, fea_edge_interaction);
   edge_region_is_master : BOOLEAN;
END_ENTITY;

ENTITY fea_surface_to_surface_interaction
   SUBTYPE OF (fea_surface_interaction);
   slave_face_region : fea_group;
(*WHERE:
```

```
Slave region is ELEMENT_ASPECT_GROUP with
face type aspect or NODE_GROUP*)
END_ENTITY;

ENTITY fea_edge_to_edge_interaction
   SUBTYPE OF (fea_edge_interaction);
   slave_edge_region : fea_group;

(*WHERE:
Slave region is ELEMENT_ASPECT_GROUP with
edge type aspect or CURVE_3D_ELEMENT_GROUP or
CURVE_2D_ELEMENT_GROUP or NODE_GROUP*)
END_ENTITY;

ENTITY fea_contact_definition
        ABSTRACT SUPERTYPE
        SUBTYPE OF (fea_interaction);
--WHERE
-- WR1: restrict properties to contact properties;
END_ENTITY;

ENTITY fea_glue_definition
        ABSTRACT SUPERTYPE
        SUBTYPE OF (fea_interaction);

--WHERE
-- WR1: restrict properties to glue properties;
END_ENTITY;


ENTITY fea_surface_self_contact
   SUBTYPE OF (fea_contact_definition, fea_surface_interaction);
END_ENTITY;

ENTITY fea_surface_to_surface_contact
   SUBTYPE OF (fea_contact_definition, fea_surface_to_surface_interaction);
END_ENTITY;

ENTITY fea_surface_to_edge_contact
   SUBTYPE OF (fea_contact_definition, fea_surface_to_edge_interaction);
END_ENTITY;

ENTITY fea_edge_to_edge_contact
   SUBTYPE OF (fea_contact_definition, fea_edge_to_edge_interaction);
END_ENTITY;

ENTITY fea_surface_to_surface_gluing
   SUBTYPE OF (fea_glue_definition, fea_surface_to_surface_interaction);
END_ENTITY;
```

```

ENTITY fea_surface_to_edge_gluing
   SUBTYPE OF (fea_glue_definition, fea_surface_to_edge_interaction);
END_ENTITY;

ENTITY fea_edge_to_edge_gluing
   SUBTYPE OF (fea_glue_definition, fea_edge_to_edge_interaction);
END_ENTITY;
```

Listing D.1: EXPRESS code for **fea_interaction** and its subtypes

ISBN 978-82-471-9399-0 (printed ver.)
ISBN 978-82-471-9676-2 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)