From Basic Design to Strength Assessment – CAD-CAE integration

Bård Rasmussen, DNV Software, Høvik/Norway, bard.rasmussen@dnv.com
Arne Christian Damhaug, DNV Software, Høvik/Norway, arne.christian.damhaug@dnv.com

Abstract

We consider integration between computer aided design (CAD) and computer aided engineering (CAE). We define and implement an application protocol interface (API) using XML schema and standard XML services embedded in our Microsoft development platforms. In order to perform strength assessment the design model must be simplified. This is a difficult process which we handle by means of a semi-automated system that we call "the idealization toolbox" (IDTB). The IDTB consists of an API derived from the schema, transformations that define idealizations, and a part library that consists of a set of parameterized structural objects that can be defaults for structural details in the vessel. In the current paper we concentrate on the application protocol, and on the idealization strategy we employ.

1. Introduction

In this paper we consider CAD-CAE integration and describe how we currently export from an early design system – NED – to a CAE system based on engineering concepts– SESAM Genie – for strength assessment. NED is short for Nauticus Early Design and is a CAD tool tailor-made for early design. SESAM Genie, a general design application for plates, curved shells and 3d frame structures, has become a leading design tool in several fields of modelling, including 3D frame structures and stiffened plate/shell structures.

2. CAD-CAE integration

2.1. CAD-CAE integration – the problem

It is assumed that the user creates a model in a CAD tool and then exports the model and the associated properties to a CAE system. There are two main problems in this approach: i) to represent the geometry in a practical way and ii) the required simplification of the model when it is transferred to strength assessment.

In addition, there are several data consistency issues and problems, and exceptions due to system failures, which we do not consider in this paper. Robust and efficient solutions to these are, however, essential for a successful system.

2.2. Application protocols and a CAD-CAE integrator concept

In order to obtain a robust and efficient export of structural data we define and implement an API. The API concept is shown in Fig.1. We use XML schema to define the API.

An important issue in the development of an integrator is the transformation from the internal object model to a model that is convenient for exchange of data between applications and thus may serve as an integrator (i.e. API). We call this new integrator model *the exchange model*. Notice that the exchange model does not contain the complete internal object model, but does only contain data of global interest. That is, we assume that the exchange model describes the structure and the format of the data to be exchanged. This means that we so far have an insufficient model if we aim at a common understanding of data that is exchanged between some collaborating applications. Thus, we need another model that, in addition to what does the exchange model, describes how to interpret the data. We call this model *the reference model*, and by means of this model data is converted to information. Thus, the reference model can be viewed as a specialization of the exchange model. In other words,

the reference model inherits the exchange model and adds semantics to enable interpretation of the data that is exchanged between collaborating applications.

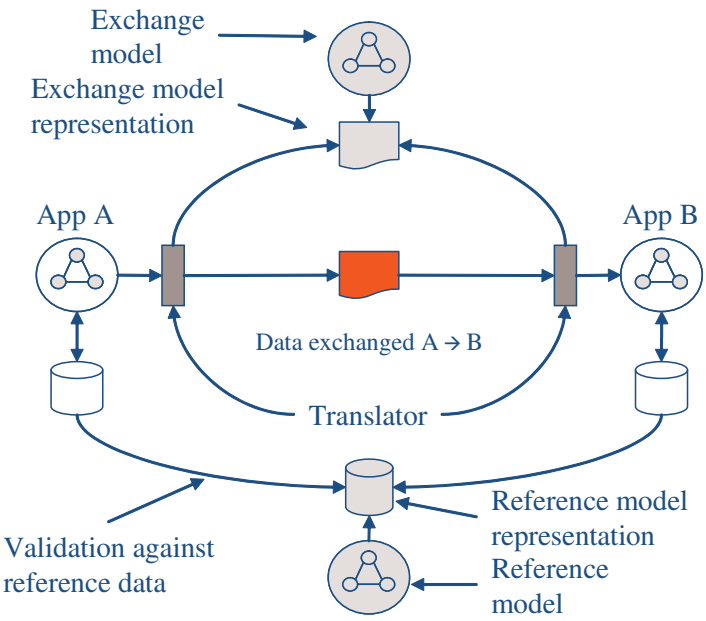


Fig.1: The CAD-CAE integrator concept.

We assume further that all entities in the reference model are specializations of at least one entity in the exchange model, and that all attributes present in the reference model are present in the exchange model. That all attributes must be present in the exchange model is obvious since any data item must be represented in the instance. The reference model in turn inherits all entities and attributes from the exchange model and provides a restricted interpretation compared to the exchange model by enforcing constraints on the behaviour.

The main objective of the API development can now be summarised as: For two applications that support the same reference model we expect that the same semantical understanding is present and that they can collaborate in the same manner as old fashioned point-to-point integration according to the rules described in the reference model. This usage of the two models is shown in Fig.1.

2.3. Geometry representation

After some internal DNV Software studies we have, for the time being, concluded that no general standard for exchange of geometry and topology does satisfy our needs. Therefore, we have decided to embed geometrical data into the XML data sets as CDATA sections that contain ACIS[1] sat representations of the geometry. The embedded geometry data can be referenced, i.e. geometrical entities included in the embedded data may be referenced directly in the application that receives the data.

2.4. Idealization (simplification)

One of the key steps in an engineering analysis is the idealization process. The aim of this process is to transform the intentions, under constraints and rules, to engineering models describing them in the language of structural engineering. The models we use are a structural model and a general constraints, boundary condition and load model which we call a constraints model. The idea is that any business intention, constraint or rule that is known and defined in the design model should be built into the engineering model in terms of structural constraints and requirements. And likewise, that

[1] Copyright 2003-2007 Spatial Corp, see http://www.spatial.com/.

any non structural intention, constraint or rule (means that it is not a part of the structure itself but affects and disturbs the structure) is built into the constraints model. These constraints are typically environmental loads, equipment loads, etc. that define and provide disturbances and constraints to the behaviour of the structure. It is assumed that rule loads, gravitational loads and any other physical constraint that may be derived from the product model by default, is transferred to the constraints model. We note that structural boundary conditions often need to be applied to regularize the problem if the analysis method is not able to handle singular problems.

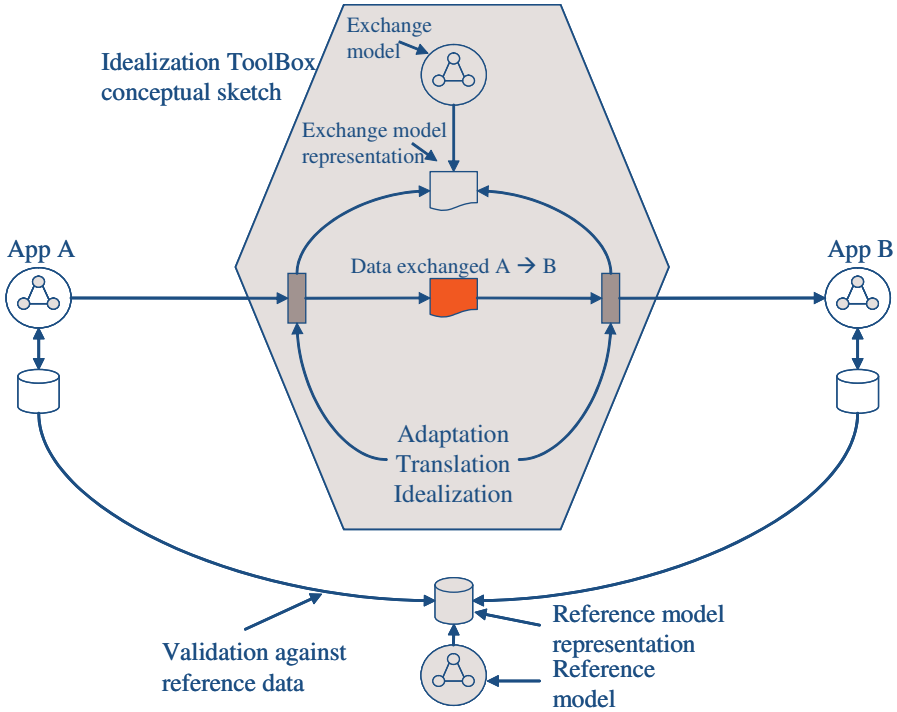


Fig.2: The Idealization ToolBox concept.

What we do is thus an idealization from a design model to an engineering model that has been embedded with engineering knowledge in order to comply with all the intentions, constraints and rules that have been set forth. The design model is assumed to contain the complete domain. We note that we do not foresee that this process will ever be fully automated, but rather be semi-automated with adequate support to ease the logistics for the engineer. The idealization may be viewed as a sequence of transformations where we record each step in order to provide a cycle-back from the engineering model to the design model. Such transformation mechanisms enable us to transfer changes in the design back to the design model in a language understood by the designers.

The Idealization ToolBox is responsible for all adaptations, translations and idealizations that are needed to exchange information between two applications that implement a common API. It is, however, not responsible for the internal interpretation of information as performed in a receiving application. Figure 2 shows a conceptual view of an Idealization ToolBox. The responsibility of the tool box is shaded, while the validation of information and data with respect to a reference model is the responsibility of the applications themselves, i.e. follows the concept shown in Fig.1.

In addition to the Idealization ToolBox, we develop an analysis and modelling framework in order to support the ideas briefly introduced above. In such a framework we need pre-processors, post-processors and general analysis components to do the strength assessment. This leads to the next step in our analysis and modelling framework, the creation of analysis models. In this context an analysis model is a numerical discretization of the engineering problem, for instance by means of finite elements. The creation of numerical models is similar to the process we use to create the engineering model and is done by a sequence of transformations. The main difference is that while the first

sequence is semi-automated, this process is automated and we facilitate means for the engineer to tailor-make the discretization to serve specific purposes. Like the former idealization process we ensure a 'way back' from the numerical model to the engineering model. We use the standard SESAM suite of analysis software to perform the basic tasks required in the analysis process, *SESAM (1996).* Since we have two engineering models we need to do a merge when we create the numerical model. This process is non-trivial but enables us to tailor the discretization to serve specific purposes, see also the short discussion on adaptive finite-element analyses in the next paragraph.

Important methods and concepts in the analysis and modelling framework are model adaptivity and adaptive finite-element analyses. Model adaptivity, *Stein and Ohnimus (1995),* is a technique we use to ensure that the numerical models are capable of representing complex physical behaviour in an adequate way in order to provide good predictions of the response. We may start in a reduced space, say 2D, and enforce kinematical constraints, for instance from Kirchhoff hypothesis. If this reduced space cannot represent what we want to analyse, we transform the model to full 3D theory in the critical areas. Adaptive finite-element analysis is another technique we use to reduce the actual numerical errors in the computations, *Ladevèze and Oden (1998).* The main concern here is to tailor the discretization to the constraint situation and enable an adaptive refinement of the discretization until it provides a response prediction with an error estimate that meets the requirements. The procedure is then a standard implementation of adaptive FE analysis. One problem with this strategy is that it works well only with simple elements. There are considerable problems to develop good error estimators for more complex finite-element formulations.

3. Implementations and examples

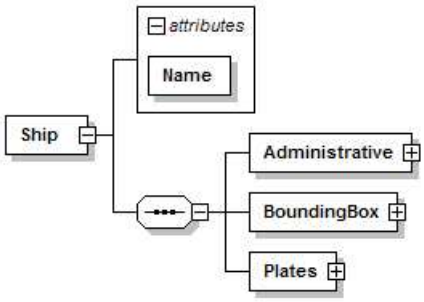
The integrator system and the idealization process discussed in Sec. 2 have been implemented as part of DNV Software's general CAD-CAE solutions. In particular, we have implemented a two way data exchange between Nauticus Early Design (NED) and Genie. The NED software suite is developed as an extension to Intergraph's IntelliShip™ software and is tailor-made for early design.

3.1. API model interface

The model is specified in XML schema and is implemented as a Document Object Model (DOM[2]).

This sub-section gives an overview of the top level entities in the model.

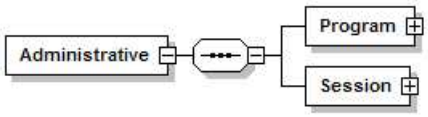
3.1.1. entity Ship



The "Ship" entity is the root of the DOM. It has a "Name" attribute and aggregates three entities.

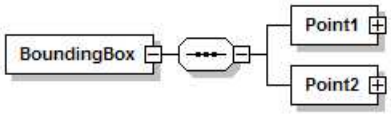
[2] The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page, for further information ref "http://www.w3.org/".

3.1.2. entity Administrative



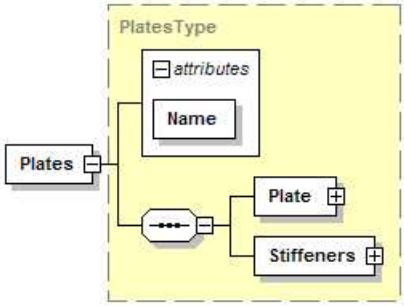
"Program" gives information about the program that created the data set (NED), and "Session" provides information about date and time.

3.1.3. entity BoundingBox



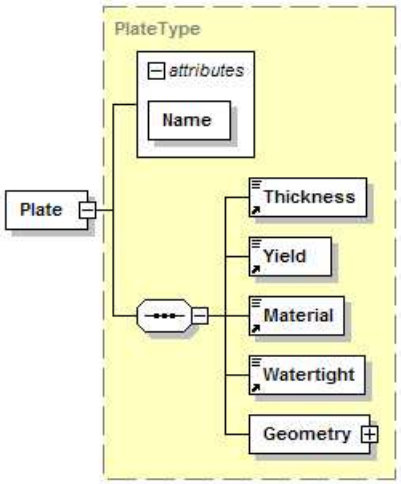
The bounding box defines the selected set of the CAD model in terms of two points. The bounding box may extend to the whole ship.

3.1.4. entity Plates



The "Plates" entity represents a plate group, has a "Name" attribute and aggregates a "Plate" and a "Stiffeners" entity. Each plate can be considered as part of the plate group. The "Stiffeners" entity so collects the set of stiffeners that are contained in the plate group. Note that we do not represent beams that are not functioning as stiffeners on the plate structure in the current version of the model.

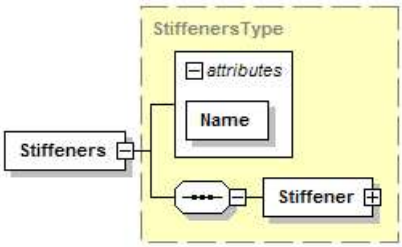
3.1.5. entity Plate



The “Plate” entity has a “Name” attribute and aggregates a set of entities of which “Geometry” impose restrictions to the use of a data set generated according to this model.

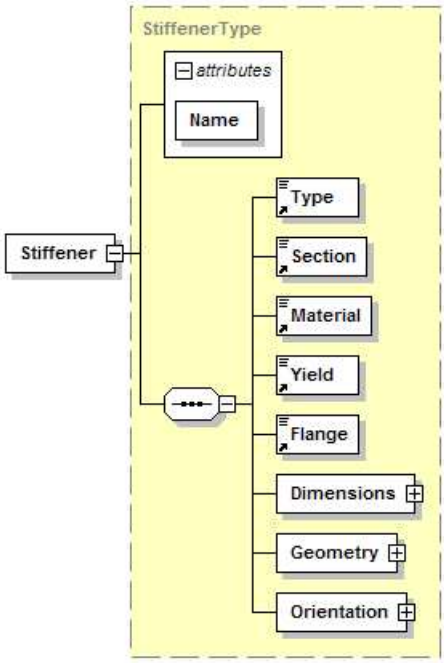
Notice that “Plate” does not nest plates. I.e. no recursion is yet implemented. This is defined in the base model we aim at – but currently the matters are as described here.

3.1.6. entity Stiffeners



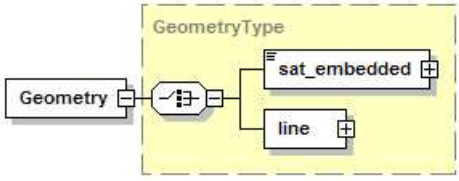
The “Stiffeners” entity has a structure similar to “Plates” and has the same role for “Stiffener” entities. Thus it has a “Name” attribute and aggregates a “Stiffener” entity.

3.1.7. entity Stiffener

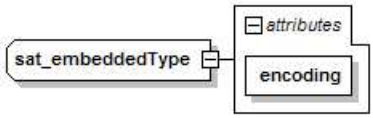


The “Stiffener” entity has a “Name” attribute and aggregates a set of entities of which “Geometry” impose restrictions to the use of a data set generated according to this model.

3.1.8. entity Geometry

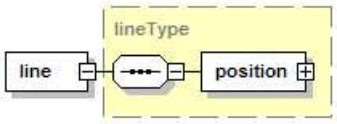


The “Geometry” entity aggregates two entities a “sat_embedded” and a “line” entity.



The “sat_embedded” entity contains a base64 decoded CDATA section that represents the geometry. This implies that the receiving application must be equipped with an ACIS sat file interpreter. A “Geometry” entity evaluates to something like the following in a data set:

```
<Geometry>
   <sat_embedded encoding="base64">
      <![CDATA[MTUwMCAwIDEgMCAgIC ……..]]>
   </sat_embedded >
</Geometry>
```



The “line” entity is an explicit geometry for a straight segment.

3.2. Idealization ToolBox

This section sketches the process flow of the Idealization ToolBox for inner structural parts, Fig.4, and shows some examples that have been processed by the Idealization ToolBox as a step under export from NED to Genie.

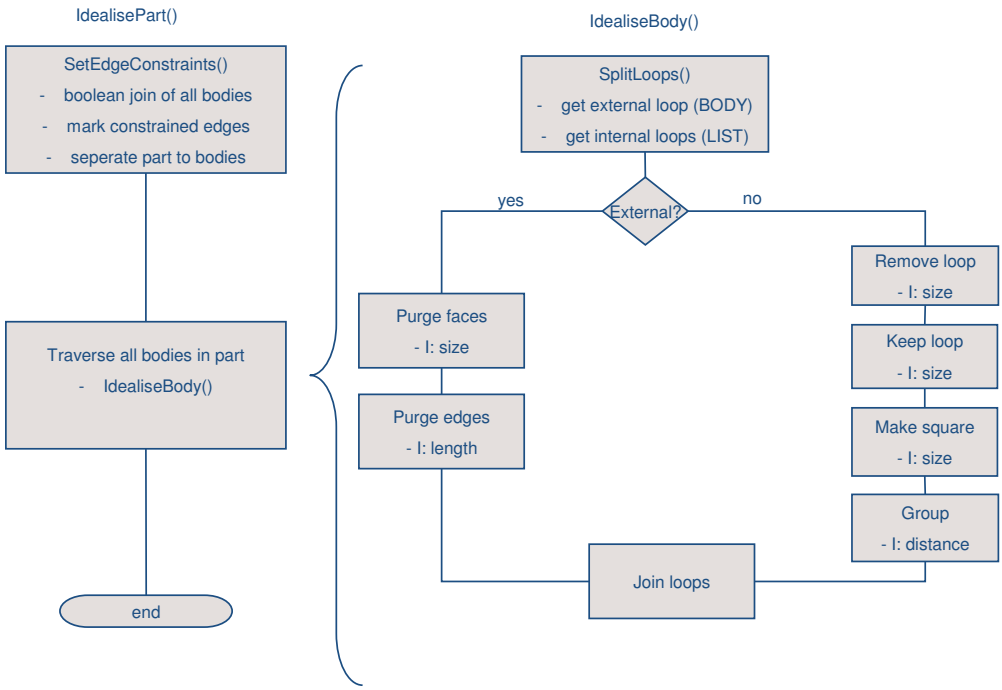


Fig.4: Overview of the part idealization process

Figs.5 and 6 show examples of idealizations to demonstrate the feasibility of the suggested approach. The idealizations include stiffener snapping, hole simplification, offset of structure to eliminate eccentricities, and extrusion line simplification (for corrugated bulkheads).

Fig.5: Mesh quality for production model, detail model and cargo hold model.

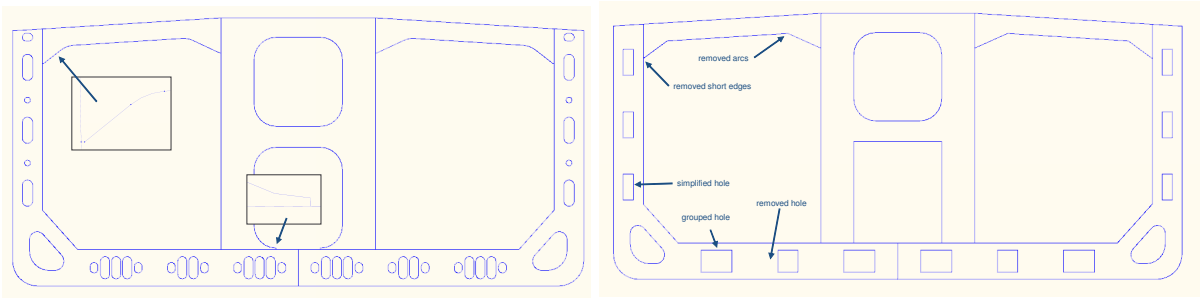


Fig.6: Idealization of web frame. To the left the design model, and to the right the idealized model.

Fig.6 shows what we call “aggressive” idealization of a typical web frame in order to reduce the size of the resulting analysis model as much as possible. Such simplifications are used for global analyses and cargo hold analyses.

3.3. Application of rule loads and code check

For completeness we include a brief section on rule loads and code checks as implemented in Genie. As mentioned in Sec. 2.4., loads are applied to a constraints model which is merged with the structural geometry in order to build a complete analysis model. Genie is based on engineering concepts and thus enables loads to be applied directly on the concepts. In a rule based context this means that loads and boundary conditions are applied to the concepts according to the governing rule sets.

After the finite element solution is available post-processing of the results are performed. In a ship classification session this often means code check, for instance fatigue and plate buckling checks. Since the engineering modeller works on engineering concepts it is possible to implement automatic generation of models for code checks. We call these models capacity models. This is implemented in Genie where the capacity models are derived from the engineering concepts, again according to the rules that govern.

4. Conclusions and further work

We have implemented a CAD-CAE system which is promising. There are issues to be solved, such as rule based idealizations where the user will be provided means to decide how specific details shall be idealized. Another issue is scalability of the solution when the model size increases. Yet, we have not met models that are large enough to challenge the system but we foresee larger and more complex models in the future and careful analysis of the scalability is needed.

An issue we omitted in this study is mesh generation. Mesh generation is hard because classification rule systems pose requirements to the mesh structure. Such requirements must sometimes be solved by inspection and we are in process of improving our mesh generation modules in order to handle the requirements.

References

SESAM (1996), *Sesam Technical Description*, DNVS Report No. 96-7006, Det Norske Veritas AS, Høvik

STEIN, E.; OHNIMUS S. (1995), *Expansion method for the integrated solution and model. Adaptivity within the FE analysis of plates and shells*, in Advances in Finite Element Technology, ed. N.-E. WIBERG, CIMNE, Barcelona

LADEVÈZE, P.; ODEN, J.T. (Eds.)(1998), *Advances in Adaptive Computational Methods in Mechanics*, Elsevier