An Application for the Management of Standardized Parts in Steel Structural Design

R. Bronsart, A. Geitmann, M. Zimmermann

March 22, 2006

Abstract

Today, the design process in the maritime industries is characterized by a complex interaction of many partners. Often, identical or similar design tasks are performed by different partners working in parallel. A significant amount of time is needed for information exchange and for the coordination of design activities. Within the collaborative research project Context Sensitive Structural Components (KonSenS) research is carried out with the objective to develop IT-based methods for the management and application of standards for steel structural design. As a result improved standardization which leads to the exploration of series effects is achieved; manufacturing costs are reduced. In this paper a flexible solution for the handling of standards is presented.

Access to an electronic catalog is provided to all partners concerned via network connections. Following a single-source approach only one standards database is needed for many different design projects. Configuration on a per-project basis allows for the tailoring of the standards to the problem at hand. In addition a subset of the catalog information is selected for certain design tasks or design contexts. These selection methods restrict the options available to the user; the probability of incorrect design decisions is reduced. For common design tasks workflows for the selection of optimal solutions are defined. Taking up ideas defined in the standardized WfMC formats these workflows incorporate strength, manufacturing and cost aspects. If required a computation module is used to evaluate the performance of a solution. Using the standards database as a result designers working on identical problems will find identical results; standardization is improved.

1 Motivation

As part of the ongoing trend towards globalization many engineering processes are outsourced to specialized partners. This behavior can also be seen in the design process in the maritime industry. Here, in a complex outsourcing scenario every partner involved has its

- own knowledge base,

- own design methods,
- own workflows.

A close and successful cooperation requires the use of evolved communication strategies. In ship design information about design procedures, best practice recommendations and specifications about the problem at hand need to be communicated. For this purpose standardization is used as a tool to define guidelines for all engineers involved. Though, even with elaborate documentation about standards etc. the interpretation of the mostly paper-based information is left to the engineer. For identical problems different solutions might be developed based on the background, knowledge and interpretation of a person. A reduction of series effect and thus an increase of cost can be registered. For approaches to a collaborative design process and their application in the industry see [1]. With the "KonSenS Electronic Catalog", in the following called KonSenS, an application server for the management of standardized parts based on [2, 3] is developed. An IT-based solution allows the direct retrieval of up-to-date information about applicable solutions by an engineer. With interfaces to CAD-systems solutions are directly linked to the standards database in the application server KonSenS. Methods for the application of workflow based design wizards and for the handling of related documentation or other information are also provided.

2 The System KonSenS

Not only is the system KonSenS a platform for the management of objects, it combines parts management, documentation management and a rule based computation system as well as unified workflows for assistance based on all information stored. In the following chapters a detailed overview over the architecture and the approach to data modelling used for data storage is given.

2.1 Architecture

The main objective of the research project is the centralized storage of all information for access by all partners involved in a certain project. For this purpose a client–server–architecture was chosen. Thin clients are used to access information objects on the server, i. e. the information is stored on the server. Using the SOAP–protocol for communication clients access the application logic like design wizards or rules via web services, see section 2.4. By using a standardized communication protocol clients can be implemented in many programming languages. The integrability into third–party systems like CAD–systems is eased. For a sample implementation see section 3.2.

2.2 Data Model

An efficient data model is a fundamental requirement for the fast access to information objects and for the communication between server and client.

Also, flexibility is a key requirement for an efficient storage of all types of information objects. Therefore, the data model uses three layers of abstraction namely

1. the Abstract Model,
2. the Meta Model and
3. the Instance Model.

2.2.1 The Abstract Model

On a generic level the Abstract Model is used to define associations between any type of information object stored in the application server. The structure of the Abstract Model is shown in figure 1 in UML notation. The classes Entity and EntityGroup are super classes of all other classes

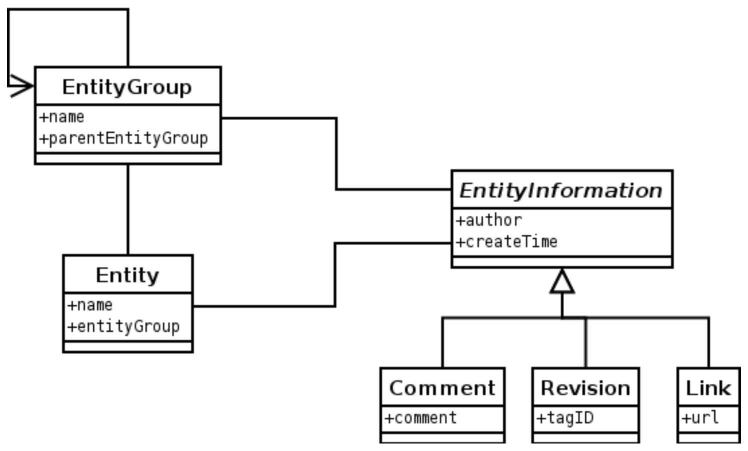


Figure 1: The Abstract Model

used in the data model.

Entity: This class is parent class for all classes used to represent data objects in the more specific lower layers of the data model, i. e. it acts as a very abstract container class for the storage of information about e. g. brackets. Entities have a name, a level unique identifier and one ore more associated EntityGroups. Associations to EntityInformation and derived objects can also be stored.

EntityGroup: Entities are grouped by properties, e. g. referring profiles or yard names, using EntityGroup objects. EntityGroups have a name, a level unique identifier and a parent group, which also is an EntityGroup. Tree–like structures for the organization of standardized solutions into a hierarchy can be defined using EntityGroups. For each attribute that should be sortable like e. g. the thickness of a plate an EntityGroup–based tree is defined.

EntityInformation: EntityInformation is the abstract superclass of all objects used to store additional information for a standard object like comments, revision tags, links etc. EntityInformation objects have an author and a time stamp depicting the creation date.

Using these general classes a structure for search in the standards database can be defined. Grouping of standard objects based on certain attributes is possible, too. Any kind of object can be stored and indexed. Associations to arbitrary pieces of information can be defined.

2.2.2 The Meta Model

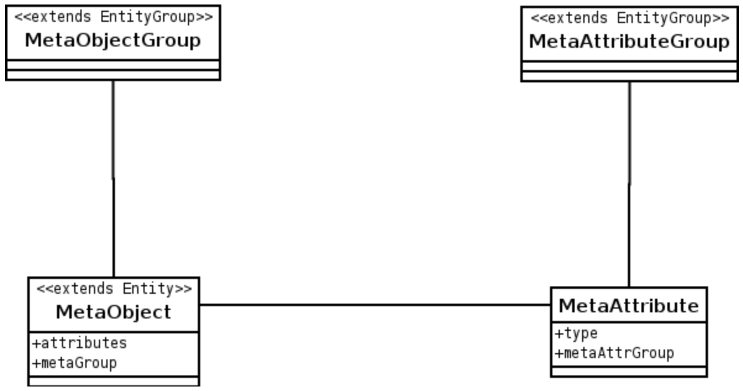


Figure 2: The Meta Model

Based on the Abstract Model the Meta Model, as shown in figure 2, is used to describe the structure of all standardized objects stored in the database. Comparable to an object oriented programming language objects and attributes can be defined. With such an approach types of standardized parts are defined. For the Meta Model the following classes are used:

MetaObjectGroup: This class is derived from the class EntityGroup. It groups the MetaObjects which are associated with a MetaObject-Group object. This structure is the base for browsing the database, see chapter 2.4.

MetaObject: The class MetaObject, derived from Entity, represents the description of a real world object like a bracket, a bulkhead or a stiffener. Attributes are associated with it. Grouping is done using MetaObjectGroup objects.

MetaAttributeGroup: Instances of this class are used for subscription of attributes, i. e. to link certain attributes like e. g. the thickness of a plate to a group named e. g. ”geometric attributes”. It is also derived from EntityGroup.

MetaAttribute: MetaAttributes define the attributes for the Meta Model. Therefore, a data type is given to define the type and range of attribute values. Values for any attribute are stored at the Instance Model level. MetaAttributes are associated to MetaObjects, so a MetaObject "BCB" has a MetaAttribute "MAT", which describes the thickness of the bracket "BCB".

E. g. for brackets a shipyard defines, amongst others, the bracket types BCB, BLK and BKB as standard types with a certain shape, material, thickness and further information. For this purpose in the meta model three different MetaObject instances are created; relevant attributes are defined by MetaAttribute instances.

2.2.3 The Instance Model

While the Meta Model defines attributes etc. for a type of a standardized object, at the level of the Instance Model values are assigned to each object. So, instances have attribute values that are either calculated by the rule based computing system or saved as static values in the database. Figure 3 shows the associations of all classes at the Instance

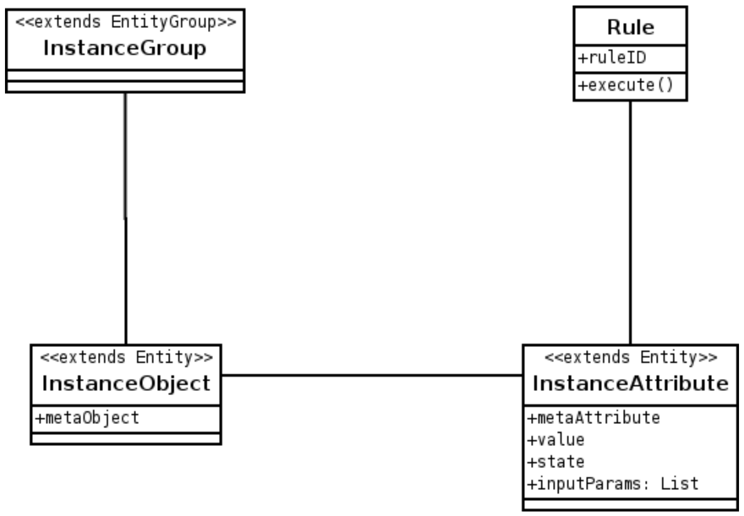


Figure 3: The Instance Model

Model. Classes are:

InstanceObject: An InstanceObject represents a standardized object, e. g. a bracket "BCB 120" of bracket type BCB as shown in figure 4. Subclassed from Entity associations to a MetaObject and one or more InstanceGroups and InstanceAttributes can be defined. This means, structural information and attributes defined in a MetaObject instance are known. The InstanceGroup instances are used for grouping of the object. The InstanceAttributes are holding values for the attributes.

InstanceGroup: InstanceGroups are used for grouping of InstanceObjects by properties defined similar to MetaObjectGroups. Subclassed from EntityGroup tree–like structures can be defined. Compared to MetaObjectGroups InstanceGroups allow the definition of more than one association to InstanceObjects. That means, for each sorting criteria, e.g. the name of the yard which uses the bracket or the profile the bracket can be mounted on, there are two separate InstanceGroup instances both associated to the InstanceObject "BCB 120". InstanceGroups are primary used for searching, see section 2.4.

InstanceAttribute: Attribute values are stored in InstanceAttribute objects. Based on the different InstanceObject instances an associated attribute (MetaAttribute) can have different values. As seen on figure 4 the MetaAttribute "MAT", describing the thickness of a bracket, has two associated InstanceAttribute instances "MAT = 8" and "MAT = 12". These values are either static (as in the example) - and thus stored in the database - or dynamic, i. e. they are calculated by the rule module that is part of the application server.

Rule: This class is used as connector to the rule based computation system. It is needed for the calculation of values for dynamic attributes.

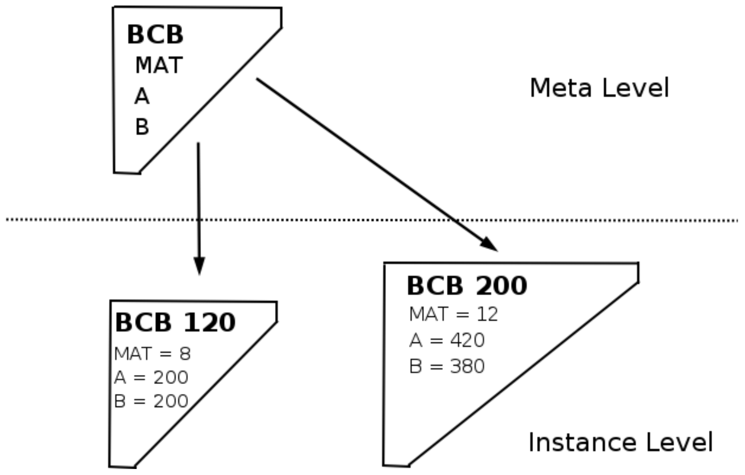


Figure 4: Example of bracket objects

Figure 4 shows two ObjectInstances for a MetaObject named "BCB", a bracket type. This bracket "BCB" has three MetaAttributes:

MAT: the thickness of the bracket,

A: the leg length of one side and

B: the leg length of the other side.

Also, two InstanceObject instances "BCB 120" and "BCB 140", representing the real brackets with their geometric parameters, are shown.

Instances of class InstanceAttribute are associated with "BCB120" and "BCB140" for the storage of attribute values.

2.3 Navigation in the Data Model

For navigation within the data model two different methods are implemented. Firstly, navigation in the structure defined by the group–type–instance structure of the standards, i. e. browsing, is available. The second alternative uses search to extract the relevant information and its structure. Combining both methods is also possible.

2.3.1 Browsing

Browsing is the navigation through the Meta Model with associated instances. In figure 5 it is shown on the left part. First, root nodes are shown. In this example only one root node "part" is presented. By selecting one of the root nodes, the related child nodes "bracket" and "cutout" are retrieved from the standards database. These nodes are instances of the MetaObjectGroup class. These objects also have child nodes, either MetaObjectGroup or MetaObject instances. MetaObject instances, here "BCB" and "KL", are leaf nodes in the Meta Model tree, but they have child nodes on the Instance Model level (BCB 120, BCB 140, KL 140) that are of type InstanceObject. On this level, the end of the tree is reached, that means, instances from InstanceObject are always leaf of the browsing tree. Once the browsing tree leafs are reached, all associated attributes with their values (A, B, MAT, NOTCH, ...) and presentations can be read from the database. The information is presented by the client, as shown in figure 6.

2.3.2 Searching

With browsing the hierarchical structure of the data model is used to retrieve a structured representation of the standards database. Often, for certain problems the engineer on the shipyard or in the design office needs to retrieve only a subset of a group of instances. Also, for certain design wizards the retrieval of standardized objects with certain values for some parameters is required.

For this purpose search for InstanceObjects is based on the grouping defined by instances of type InstanceGroup. Using the instances tree-structured graphs are created where each tree represents a property of searchable InstanceObject objects. The root nodes of a search tree are called systematics. As an example on the right of figure 5 examples for search trees are shown. E .g. using the systematics "Yard" all standard objects available for a given yard can be found. For a flexible definition of queries for information retrieval a query language is supported. Omitting basic definitions for comparators and parameter values the characteristics of the query language defined in a simplified EBNF is as follows:

```
Query          = SearchPath ['#' Filter]
SearchPath     = '(' {ViewDefs} | ViewDef ')'
ViewDefs       = '('[BindOperation]{ViewDefs} ')'
```

```
BindOperation = '|' | '&' | '!'
ViewDef       = '('Systematic {',' Group } ')'
Systematic    = 'S=' String
Group         = 'G=' String
Filter        = '(' {FilterDefs} | FilterDef ')'
FilterDefs    = '('[BindOperation]{FilterDefs} ')'
FilterDef     = Attribute Comparator Value
```

Using constructs based on this language flexible queries can be defined.

As a simple example the query

```
(&(S=part, G=bracket)(S=design, G=refprofile, G=FP))
                     #(mat < 12.0))
```

retrieves all standard objects of type (S=part, G=bracket), i. e. all brackets, that can be assembled on flat bars (S=design, G=refprofile, G=FP, G=200) and that have a plate thickness less than 12 mm (mat < 12.0). The definition of more complex queries can be performed using logical operators like AND, OR and NOT.

2.4 Communication

As mentioned above, the communication between server and clients is based on the SOAP[5] protocol. SOAP is a lightweight XML-based messaging protocol used to encode the information in Web service-Request and -Response messages before sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols, including SMTP, MIME and HTTP. A SOAP message is modelled as Head–Body–Pattern. A SOAP envelope acts like a container in so far as that it stores a message header and a message body. The header has information for authorization, routing etc. while the body contains the user data for the web service.

Because firewalls are normally used to protect the internal network of shipyards etc. the more comfortable or more powerful communication protocols [4] for client–server–applications like Java RMI, CORBA or DCOM are not used in the scenario described. These protocol, while offering more advanced features like remote method invocation and passing of complex objects require complex configuration if used across firewalls or proxies.

For browsing, search and more advanced features web services are published using the Web service Description Language (WSDL) [6] and are used by client applications to access the data and application logic of the application server KonSenS. With such a thin–client based approach enhancements to the capabilities of the system or bug fixes to the application logic can be deployed often without the complex and time-consuming need to update all clients.

3 Management of Standardized Parts

In the previous chapter a highly configurable system for the management of standardized parts was introduced. For each field of application the data model to use must be defined, i. e. types with the corresponding attributes and related instances must be given.

3.1 The KonSenS Model

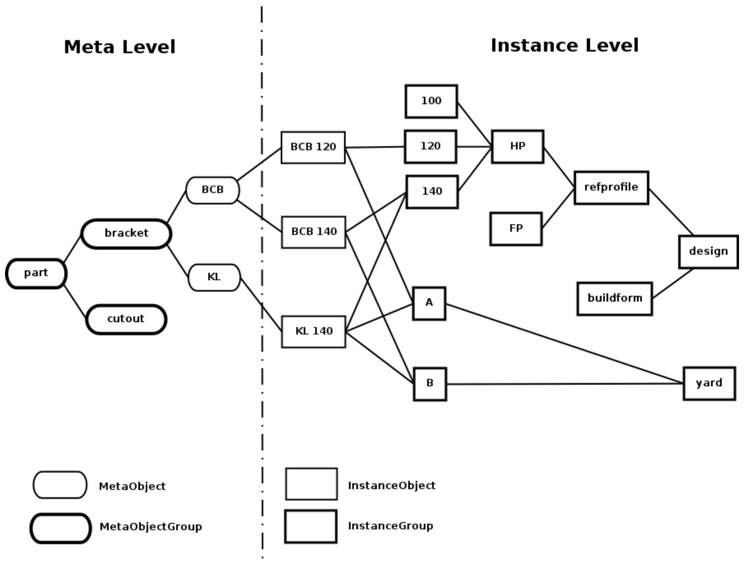


Figure 5: Sample from the KonSenS Data Model

Figure 5 shows a part of the data model used for the application in the steel structural design process. Shown are three ObjectInstances "BCB 120", "BCB 140" and "KL 140". To the left of these objects the corresponding Meta Model is shown. On the right side systematics used for searching are presented. Here, the objects can be grouped by:

- the yards, which use the brackets and
- the referring profiles, in this case bulb profiles with 120 and 140 mm height.

3.1.1 The KonSenS Meta Model

As seen in the chapter above, the Meta Model defines the main syntax and semantics of all objects used. For the usage in steel structural design the following MetaObjects and -Groups are defined:

MetaGroups: These objects are used for browsing the tree to InstanceObjects, i. e. they define a very general structure for all standardized parts. The following groups are defined:

Part: Parts are main structures used in steel structural design. These are solid structures. Currently, the following parts are implemented as child groups:

- brackets,
- clips.

Feature: Features are geometric details of parts like holes or cutouts, i. e. these are objects that modify the geometry of existing parts. As an example cutouts are currently implemented as children of features.

MetaObjects: For steel structural design MetaObjects are used to define parts used. These are stored as child nodes in the Meta Model tree. In figure 5 the nodes "BCB" and "KL" represent MetaObjects.

The KonSenS Meta Model is defined by a XML–file. As an example the following excerpt of a definition file shows the definition of a MetaObject called "BCB", a certain type of bracket.

```
<metastructure>
         ...
  <attributegroups>
   <attributegroup name="geometry">
    <attribute name="A" type="INT"/>
         ...
    <attribute name="OFF" type="FLOAT"/>
   </attributegroup>
   <attributegroup name="common">
    <attribute name="name" type="STRING"/>
   </attributegroup>
         ...
  </attributegroups>
  <objectgroups>
   <objectgroup name="part">
    <objectgroup name="bracket">
     <object id="bcb">
      <ref_presentations>
       <presentation_ref type="TRIBON"/>
       <presentation_ref type="IMAGE"/>
      </ref_presentations>
      <ref_attributes>
       <attribute_ref name="name"/>
       <attribute_ref name="A"/>
       <attribute_ref name="MAT"/>
       <attribute_ref name="NOT"/>
       <attribute_ref name="NOA"/>
       <attribute_ref name="OFF"/>
      </ref_attributes>
    </object>
   </objectgroup>
  </objectgroup>
  <objectgroup name="feature"/>
         ...
  </objectgroups>
         ...
```

```
</metastructure>
```

The MetaObject shown has several attributes with names based on the Tribon syntax. As seen, the MetaObject "BCB" is a child node from the MetaGroup "bracket", which will be a child of "part".

In the example two presentations are defined. One is used for the integration in Tribon, the other one is an associated image. With multiple representations a single source approach can be used. I. e. one common data set is used where for each target system - this can be a CAD-system, for visualization purposes etc. - only the relevant attributes are shown.

At present 35 MetaObjects, i. e. 35 different types of standardized objects, are stored in the database.

3.1.2 The KonSenS Instance Model

The Instance Model stores all InstanceObject instances of the formerly described MetaObject instances. Within the collaborative research project KonSenS information about standardized parts is delivered by the project partners, three major shipyards in Germany. In table 1 an overview over the number of different parts managed is given.

MetaObject Names	**Number of Instances**
brackets	626
clips	177
cutouts	341

Table 1: Number of standardized parts in KonSenS

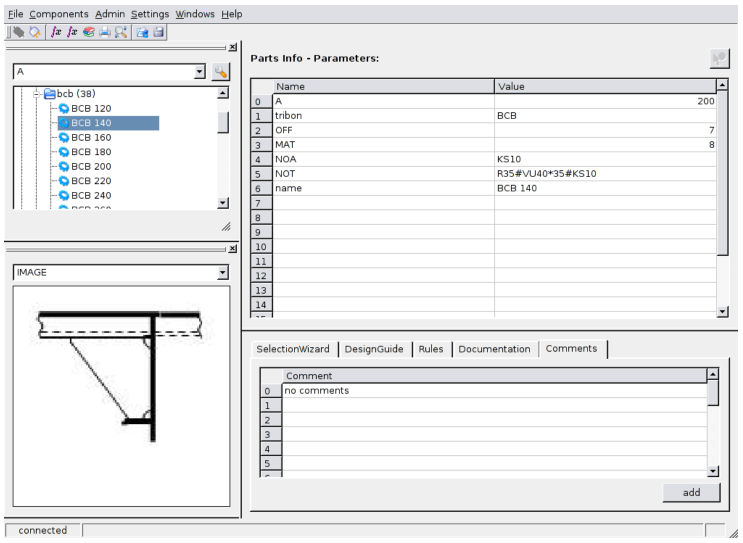
Similar to the Meta Model instances are also defined in an XML–File. Administrators is given the option to automatically generate Spreadsheet-Files from this description that can then be distributed to the engineers for data entry. Completed spreadsheets are imported into the system. Changes to the data model are possible, i. e. round-tripping is supported.

3.2 Clients

Within the research project two clients are implemented as prototypes. These are used for demonstration purposes. Both are implemented in Python; the pyQT framework is used for visualization and to generate the user interface.

Standalone Client: This client is used as named, i. e. it provides an interface to the standardized parts and related information stored in the standards database. By browsing the MetaGroup tree all parts can be accessed. The client is also used as front end for the document management system.

Tribon integrated client: For the usage in steel structural design this client is integrated in the CAD–system Tribon using the Vitesse



Parts Info - Parameters:

	Name	Value
0	A	200
1	tribon	BCB
2	OFF	7
3	MAT	8
4	NOA	KS10
5	NOT	R35#VU40*35#KS10
6	name	BCB 140
7		
8		
9		
10		
11		
12		
13		
14		

	Comment
0	no comments
1	
2	
3	
4	
5	

Figure 6: Screenshot of the Standalone Client

application programming framework. Basic workflows like the standards conformant construction of brackets and cutouts are implemented. For this purpose the structural model of the CAD-system is evaluated; possible solutions for a chosen problem are retrieved using search and systematics. For a design task like the definition of a bracket attribute values are entered automatically as far as possible thus reducing the number of user inputs required.

Due to the open design and the consolidation of the complete application logic into the server the development of additional clients that integrate CAD–systems or further applications into the system is possible.

4 Summary and Outlook

A central repository for standardized parts allows for instant up-to-date access for all partners working on a certain project. A single source approach reduces the time needed to manage and maintain multiple standards concurrently. With the integration of the standards database into CAD–systems the time needed for certain design tasks and the probability of design errors can be reduced.

Currently, the system KonSenS is a first prototype focused on storage and retrieval of standardized parts. It is a highly configurable and flexible system. By using the data model described, engineers have a unique data source for the steel structural design process. The access to additional information using EntityInformation derived classes allows for a further consolidation and a tighter integration of all relevant pieces of information.

Further work is currently carried out on the closer integration of a rule–based computation system for the dynamic evaluation of parameters or conditions. Advanced workflows for a guided wizard-based design are developed.

5 Acknowledgement

The work presented in this paper is supported by the German Federal Ministry of Education, Science, Research and Technology under grant 03SX163D. The authors acknowledge the valuable discussions with the partners of the project Konstruktionsstandards fr schiffbauliche Strukturen zum Einsatz in CAD-Systemen, KonSenS.

References

[1] Hyeong–cheol Kim, et al., Introduction to DSM Hull Modelling System "COSMOS" based on TRIBON M3, ICASS 2005

[2] M. Zimmermann, R.Bronsart, K. Stenzel, Knowledge Based Engineering Methods for Ship Structural Design, ICASS 2005

[3] R. Bronsart, M. Zimmermann, Knowledge Modelling in Ship Design using Semantic Web Techniques, Compit05

[4] Young–Soon Yang, A Study on the Web–based Distributed Design Application in the Preliminary Ship Design, ICASS 2005

[5] SOAP Spec, http://www.w3.org/TR/SOAP

[6] Web Services, http://www.w3.org/2002/ws