Knowledge Based Design Patterns for Detailed Ship Structural Design

Michael Zimmermann[1] and Robert Bronsart[1]

ABSTRACT

For detailed ship structural design standardization is one means to ensure a consistent build quality. Cost reductions due to the exploration of series effects can be achieved. With a knowledge based approach standards can be defined precisely and machine interpretable. Support for routine is provided.. Using a modular approach a solution for the semi-automatic design of standardized systems for ship structural design is presented. The design context, i.e. the effect of external design parameters, is defined. Algorithms for the automatic application of knowledge about standards are presented. Solutions for the evaluation of design intent and design context are developed. Using a bottom-up approach individual design assistants are assembled and allow the flexible design of complex structural members like complete floor plate structures. The advantages of a knowledge enhanced product model for design changes are presented. It is shown that automatic validation can be applied to such a product model hence ensuring consistency and supporting quality management as well as reducing the probability of design errors.

KEY WORDS

Detailed design; Knowledge-based engineering, Series effects, Standardization; Rule-based production systems

INTRODUCTION

The shipbuilding industry is facing a highly competitive market that makes high demands regarding product quality, performance and price. For detailed ship structural design standardization is one means to ensure a consistent build quality. Cost reductions due to the exploration of series effects can be achieved.

Today, standards for detailed design are mostly defined and distributed as paper based catalogs; engineers are responsible to understand and adhere to these standards. As a result, for a given design solution design assumptions and decisions are not stated explicitly in the product model, yet are relevant for following design activities.

With a knowledge-based approach standards can be defined precisely and machine interpretable. An automatic application of these standards under the supervision and control of an engineer is made feasible, thus providing support for routine tasks and helping the engineer to concentrate on innovative aspects of the design process.. From a standpoint of configuration design algorithms for the automatic application of knowledge about standards can be applied to formulate design activities relevant for a standardized design solution. Automatic validation can be applied to such a product model hence ensuring consistency and supporting quality management as well as reducing the probability of design errors. Using a modular approach a solution for the semi-automatic design of standardized systems for ship structural design can be developed

DESIGN PROCESS AND THE ROLE OF STANDARDIZATION

As shown in Figure 1 the detailed design phase influences the total costs significantly. To reduce costs the main contractor, mostly the shipyard building the vessel, defines a set of regulations and best practice recommendations, the so called design standards. These standards are highly optimized to increase the productivity during manufacturing and to satisfy all requirements regarding strength, fatigue, etc at the same time.

Focusing on ship structural design, the initial design phase is used to adapt existing standards and to optimize these standards

[1]Department of Mechanical Engineering and Marine, University of Rostock, Germany

to manufacturing capabilities and applicable regulations taking experience from previous projects as well as contractor's requirements into account. During detailed design these standards are then applied by the engineer to design the vessel according to the rules defined in the standards.

Applying design standards the number of different solutions to similar design problems in a vessel or in a series of vessels can be reduced. With these series effects manufacturing costs can be improved. The workers on the shop floor are more familiar with the reduced number of solutions, leading to a decrease in labor time needed and at the same time to an improvement in manufacturing though product quality (Zimmermann et. al 2005).

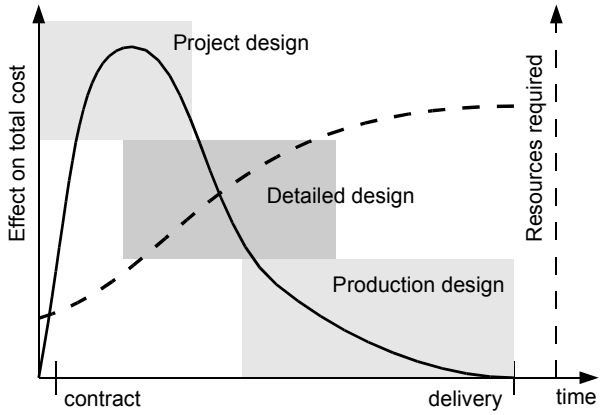

Figure 1: Effect of process and total design time on cost

To reduce design time similar tasks concerning independent aspects of the design of a vessel are deployed to different partners. For example the design of all steel structural parts of the bow section is done by one design agent; the design of the aft section is done by another design agent or at the shipyard. Design offices, either at the shipyard or at design agents, are often working on many projects at the same time. This requires that the individual engineer

- has to work in different design contexts at the same time
- and has to get familiar with the recommendations and requirements of the contractor for each project.

For new projects engineers are required to get familiar and productive with the standards and regulations supplied within a limited timeframe. Working on different projects also requires a constant change of the standards used; the probability of using incorrect regulations or misinterpretation is increased leading to increased costs for necessary changes later on.

Standards offer a means to define solutions to similar problems. These standards are used by engineers at the shipyard and design offices to design a vessel in a way that is best for the manufacturing capabilities at the construction site. As a result manufacturing costs are reduced and the build quality is improved. If common standard parts are delivered or produced in large quantities they do not need to be manufactured individually. The control of material flow can be improved.

STANDARDS – A DEFINITION

For a given problem, even if limited to feasible solutions only, the number of applicable or possible solutions can be large. Based on this domain of solutions standards impose certain constraints with the objective to limit the solution domain. Constraints can

- either be placed on certain aspects,
- can exclude some solutions completely
- or can allow certain solutions only.

Generally speaking, aspects can be anything that concerns the problem at hand and that influences the result. Constraints limit the parameter space for aspects. This limitation is performed using mathematical operations like value assignments, or solving equations. As an example the class rules defining certain requirements regarding stiffener dimensions in relation to plate thickness can be named. Description logic allows the definition of relationships between different aspects hence forming a network of rules for a given standard (Piera 2003).

Here, a distinction between

- constraints that only affect aspects of the standardized object
- and relations that also encompass information from the surroundings of the object, the design context

needs to be made.

As an example in Figure 2, a bracket for a stiffener-stiffener connection is shown. Looking only at direct first-order relationships, the length of the flange l and the thickness of the bracket t are properties of the object alone. The selection of the bracket also requires information of other objects. In this example the section module of the profiles to be connected needs to be known. These can be derived from the type of connected stiffeners and the stiffener dimensions.

In addition to these strict constraints descriptive constraints can be imposed. Descriptive constraints are constraints that are complex to state explicitly in terms of equations or boolean expressions but are provided as e. g. explanatory text. Often, this type of constraints requires complex information about the location in the ship, strength and fatigue levels, manufacturing procedures as well as additional information normally not given in the data model. In detailed structural design the selection of the best solution from a list of permissible solutions is heavily influenced by weak information.

Here, the knowledge of the engineer is required to evaluate the information provided with respect to the problem at hand and to decide if a specific weak constraint is applicable or not. The possibility of misunderstandings is given.

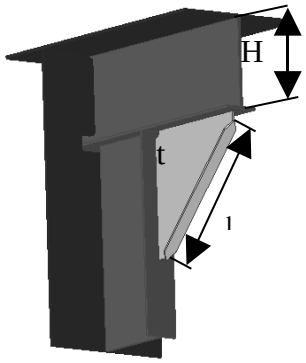

Figure 2: Local and Context Related Aspects

In practice the number of aspects applied for a given standard is limited by issues like maintenance, complexity and the availability of information. Therefore, solutions that are applied often in a typical design scenario or solutions with high error rates and high costs for correction are candidates for standardization. Also, from an economic point of view standardization is only used if the effect of standardization gives substantial benefits regarding cost or quality compared to the effort needed to define, document, distribute and apply these standards. For a standards database the cost involved in defining certain information explicitly within the data model or in a separate application needs to be taken into account, too.

Therefore, in ship design standardization is mostly applied in the area of steel structural design. Here, high numbers of identical solutions like parts or features are used. The identification of all relevant aspects determining the decision for or against a specific solution is feasible yet complex. An opt-out option for uncommon problems with special requirements needs to be provided to allow the engineer to overrule the solution given by a standard (Nieuwenhuis et. al).

DESIGN ACTIVITIES

In the design process different design activities are executed. Each design activity depends on the results of other activities; design activities might be executed sequentially or in parallel. Design activities can be described at different levels of granularity. Following Hubka the generic notion of design activity can be defined as follows (Hubka 1993):

> *Definition: A design activity is a change of the state of information by means of a design method based on knowledge of the designer or any other acting entity and the context of the evolving design.*

From the knowledge level design activities can be described by goals, actions and knowledge as shown in Figure 3. The input and output of the design activity can be expressed as knowledge. The scope and focus of the input knowledge I_K is influenced by the perception of the design context and the general understanding of the problem domain. A design goal G_D is derived from the general requirements, the global objective. Often, conflicting goals needs to be addressed. As a problem formulation is often goal driven the goal influences the selection of an applicable solution algorithm or design activity A_D. The

formulation of the design activity without taking the solution into account is not possible. As output, the design activity creates additional output knowledge O_D unknown before. If no valid or suitable solutions were developed the output knowledge is used to adapt the goal of the design activity. A new internal iteration may be started.

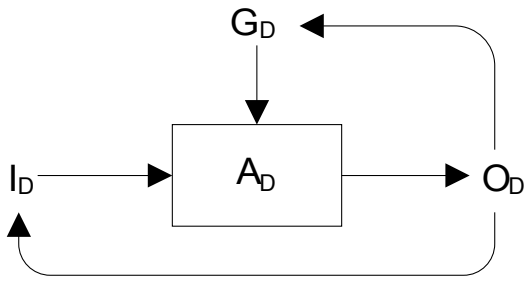

Figure 3: Formalism for a Design Activity

The output may prompt another design activity. The first design activity hereby servers as provider of input knowledge for consequent activities. Constraints are created for further development steps. Often, a design goal may cause several design activities to be executed in sequence or in parallel. The process terminates when the global objective is reached or no more actions can be performed with the knowledge available.

As shown by Sim (Sim 2003) design activities can be classified according to the objective aspired. Three different groups with the activities as shown in Table 1can be identified.

Table 1: Classification of Design Activities

Type	Objective	Action
definition	function to form/structure	sythesising, abstracting, generating, decomposing. associating, comprising, structuring or integrating, detailing, defining, standardizing
evaluation	form/structure to behavior/effects	decision making, evaluating, selecting, analyzing, modeling, simulating, testing or experimenting
management		constraining, exploring, identifying, information, gathering, resolving, searching, decomposing, prioritizing, planning, scheduling

The first category classifies different approaches an engineer might take to reformulate a given design problem so that a potential design solution can be reached. From functional requirements the form or structure of the product is derived. The complexity of the evolving design is managed until a solution has been reached. Once potential (intermediate) design solutions are derived evaluation activities are performed to validate that a solution satisfies the requirements. Also, by discarding infeasible solutions the design solution space is reduced. Two different approaches to this task are possible. On the one hand, a problem focused concentrates initially on the analysis of the problem followed by a systematic concretization and definition process. On the other hand, with a product-focused approach assumptions about possible solutions are made. These are used to gain further insight into the problem and to refine the proposal. Design activities are related to each other. For this purpose management tasks are required. Using strategic considerations the design process, i. e. the coordination of design activities, can be influenced influenced.

The design definition activities can be arranged according to how concrete the model is and the level of detail. The activity of generating is supported by decomposing known solutions, associating ideas or concepts and composing them into design concepts. The detailing level of feasible yet undetailed solutions becomes detailed and concrete through activities like defining, detailing or standardizing. Entities of a domain are configured to construct a reliable structure that satisfies the design requirements. This activity involves search, exploration and discovery of design solutions. These partial solutions are integrated and compositioned into the global model.

CONFIGURATION DESIGN

In the domain of engineering the design process can be defined as goal-oriented search. In most cases, a process of successive design refinements is used. A top-down approach is applied. In each iteration loop, requirements are identified and tested; design decisions are made. The form of components and their fit is evaluated. Functional and behavioral aspects are

considered. The working principle is adhered to. If a pre-defined set of components is given and assembly of selected components is sought that satisfies a set of requirements and adheres to a set of constraints, this search strategy is defined as configuration design (Wielinga 1997).

> *Definition: Configuration design is a goal-oriented problem solving process. The arrangement of a set of known objects based on given functional, geometrical and other requirements is determined.*

A component can be a physical entity, an activity etc. An optimality criterion or cost function can be applied to sort possible results. According to Wielinga, the classification with respect to configuration design as shown in Table 2 can be given. Based on this classification examples of real-world configuration problem types are identified in Table 3.

Table 2: Classification of Configuration Design

Components	Assembly	Requirements and Constraints
Fixed Set	Fixed	Local & directly applicable
Parameterized fixed set	skeleton	incrementally applicable
parameterized set of types	free	functional or global

Table 3: Examples for Configuration Design

Task Category	Components	Assembly	Constraints & Requirement
Assignment	Fixed Set	Skeleton	Incremental
Layout, Scheduling	Fixed Set	Free	-
Parametric Design	Parametric Set	Fixed	-
Skeleton Design	Parametric Types	Skeleton	-

If the set of components is determined entirely in advance, i. e. all properties relevant for the design task are fixed and known, a combinatorial problem needs to be solved. The arrangement of the component needs to be determined to satisfy all requirements and to fulfill all constraints. A parameterized set is given if the parameters and the general shape of each component is known yet the parameter values for one or more parameters can be chosen by the solution. In the field of naval architecture e. g. stiffeners in stock can be described as parameterized set in so far as for each stiffener the contour is defined by the material in stock, yet the length depends on the manufacturing process. Finally, a parameterized set of types is an abstraction of the previous class. For each type of component multi-dimensional parameters are present. If a complete bulkhead e. g. is given as a type, not only the dimensions of the stiffeners applied can be chosen but also the number of the stiffeners itself. Therefore, for a parameterized set of types the final generic shape is not given.

A second source of variant is defined by the relations that define the assembly, i. e. the arrangement of the components. If the arrangement is known complete the configuration design task is reduced to the selection and assignment of parameter values for the components of the arrangement. A less restrictive case is given if the skeleton of the arrangement is given. The specific arrangement has to be determined for the solution. The most flexible and most complex case is a free assembly. The arrangement of the components can be chosen by the design task as long as constraints and requirements are not violated.

Requirements or constraints can be directly applicable, i. e. they restrict or define the value of an individual parameter of a component directly. Incremental constraints are constraints that act on parameters derived from a combination of components. The evaluation of such constraints is only possible if a sufficiently detailed model is already generated. For a functional requirement some relation between an arrangement and the functional characteristic of this arrangement is evaluated. The explicit definition of the relation is required.

KNOWLEDGE-BASED DESIGN ACTIVITIES

As shown above the design process in detailed steel structural design can be classified as configuration design activity. From a set of known components suitable solutions are selected and evaluated. The context for the problem at hand is analyzed. Based on this knowledge an optimal solution is selected from the set of suitable solutions.

Two different usage scenarios can be distinguished:

- Execution of a self-contained design activity. E. g. the definition of a single bracket for a given problem at hand is a self-contained design activity. No further knowledge about previous or following design activities is required.
- Control of the design workflow, i. e. the execution order of a sequence of design activities. E. g, the design of a complete standardized floor plate arrangement can be identified as an ordered execution of subsequent design

activities. Each design activity is responsible for a single design action. The selection of suitable design activities as well as the execution order depends on the problem to solve. The design context plays an important role hereby.

The results presented use a rule based approach. Information about the steel structure to refine is based on a Tribon Steel product data model. A so called production rule system uses rules that work on facts, the so called working memory. Each rule consists of a precondition and the action to take if the precondition is met.

WHEN
    Plate Thickness > 20 mm
THEN
    Use X-Welds

Multiple checks can be present in the precondition. The action part can contain multiple actions. The working memory can be modified by an action. Based on the current state of the working memory, i. e. the facts given, adequate rules are executed. This process continues as long as there are rules left to call.

In the prototype application, each design activity is formulated using a set of rules. These rules capture the design requirements and constraints as preconditions. The design intent is expressed in the action part of the rule. For the working memory two different types of facts are defined, namely:

- the set of standard components,
- the product model for the design problem.

For the representation of the standard components, the modeling approach chosen reuses constructs presented in the ISO 10303 STEP standard. As Tribon provides XML schema definitions for the exchange of steel structural data, constructs defined in the Tribon XML schemas for data exchange are taken up and modified. A class hierarchy as shown in Figure 4 is defined and represents the graph structure of the dependencies. Important parameters – e. g. the cross section of a stiffener type etc. – are defined. For each class of components the set of valid instances is defined. Parameters values are assigned for each instance. Relations between different instances can be modeled. E. g., based on the STEP specification, functional aspects can be expressed.

The standard components are modeled using the web ontology language OWL. Defined by the World Wide Web Consortium, OWL denotes a family of knowledge representation languages for the definition of ontologies. Main considerations for the selection of this language were as follows:

- Expressibility: OWL supports all common modeling constructs used for data model design. Type hierarchies can be defined. Inheritance is supported. Data type properties allow assigning specific values for certain aspects of a components. With object properties links between different component items can be defined.
- Reasoning: As OWL is based on first order description logics, support for automatic inference is available. This allows for semi-automated consistency checks of the data model developed. Inconsistencies can be detected. For the instance data constraints to fulfill can be defined and validated.
- Tool and Community Support: With open-source libraries like JENA for the programmatic access to the data, the integration of OWL data models into applications is easy. An extensive community and multiple vendors working with OWL provide a wealth of information.
- Editing: Powerful editing environments for the definition and management of class structures. In contrast to e. g. relational databases or STEP databases direct editing of instance data is supported. The definition of relations between instance objects, i. e. a definition of a graph-oriented network, can be performed comfortably.

The system uses a task specific data model representing the steel structural model. This model is based on the Tribon M3 XML model for steel structural design A task specific and hence simplified data model was chosen, as a full featured data model like STEP AP218 or the Enterprise Reference Model from Atlantec would lead to a significant increase of the complexity for the rule formulations used.

The data provided by the XML data model is imported into OWL and hence simplified. E. g. detailed definitions of the geometry of the boundary are removed. Geometric operations are used to determine the topology of the steel structure. Intersections of structural members are derived. For this purpose multiple transformation steps are defined using the SPARQL query and data transformation language. As a result a custom topological model with only the information required is created.

A direct interaction with the product data model present in Tribon M3 is currently developed. Based on the Data Extraction Utilities and the Vitesse Framework provided by the Tribon system a custom API is created. This API allows the transparent interaction with the relevant geometrical, topological and functional aspects of the model. For each object in the model functional aspects can be retrieved using a top-down approach. Using Component Object Model Techniques (COM) the API can be accessed from any application and can hence be used as data provider for the rule system.

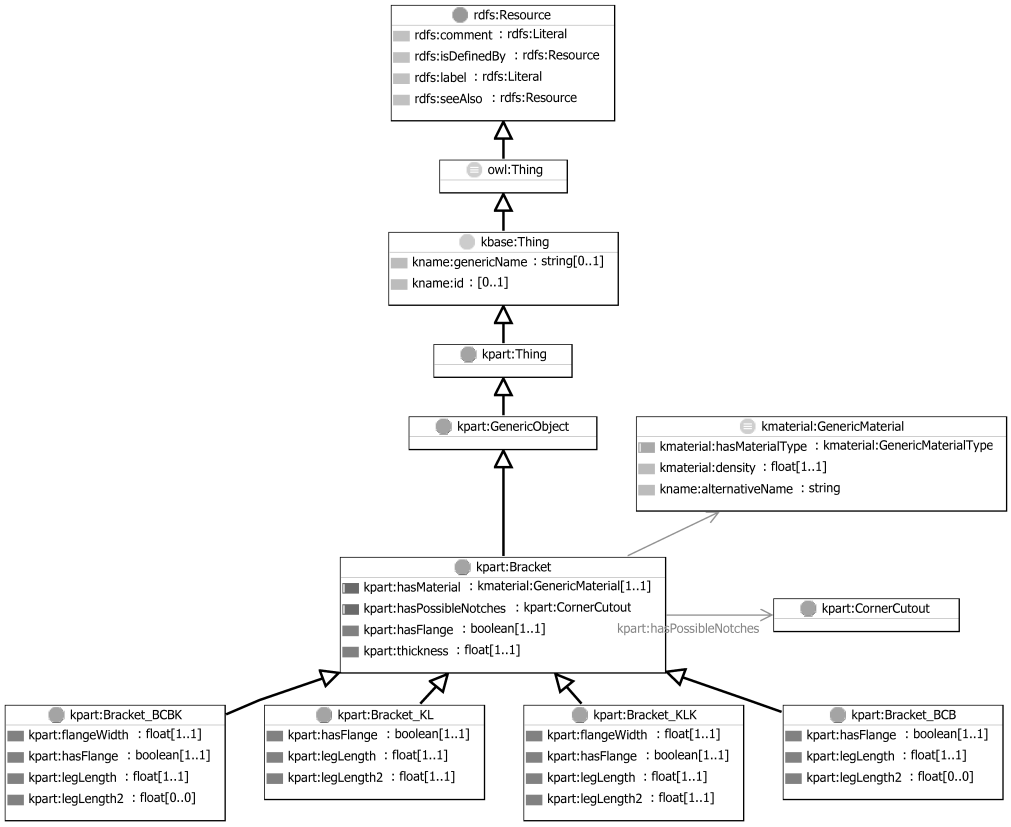

Figure 4: Class hierarchy for the Bracket Definition

The system is based on the open-source rule based system Drools. Drools is a high-performance production rule system written in Java that supports forward chaining i. e. input driven reasoning. Advanced features like custom rule execution ordering and the specific activation of selected groups of rules are supported. Based on the selected design task the corresponding rule definitions are activated, see Figure 5. The working memory is populated. The component database and the steel structural model are loaded. For this purpose the information available in OWL syntax is automatically transformed into JAVA objects and stored into an object oriented database. As database DB4o is used. An object oriented database was selected as this provides means for a transparent access to the objects stored in the database. Other means for data storage like hibernate etc. could be implemented.

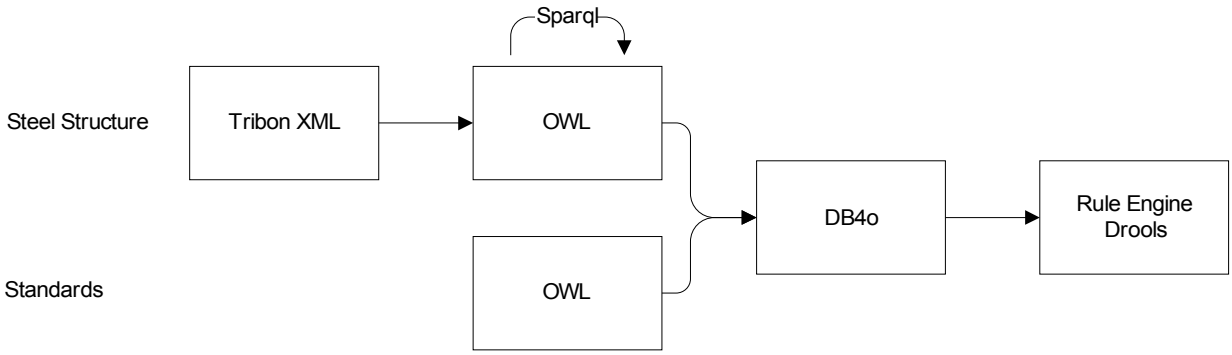

Figure 5: Data Preparation for the Rule Engine´

STANDARD COMPLIANT DESIGN ACTIVITIES

As an example for the rule based implementation of an atomic design activity the standards compliant and context sensitive selection of a bracket is shown. Here, the bracket acts as connection element between a plate and an intersecting stiffener.

The list of standardized bracket types and instances are taken from a recent project of a major German shipyard. The context, i. e. the determination of external parameters that have an influence on the design result, were identified in close cooperation with engineers on the shipyard. These are:

- The orientation of the profile
- The dimensions of the profile and it's cross section
- The overall load and strength requirements
- Space constraints

An unambiguous assignment of valid bracket sizes to the size of the profile is given in the standards database

For selected sections of the vessel the final steel model was analyzed. A statistical breakdown of the bracket types used in these sections is shown in Table 4. Three different bracket types – namely KL, BCW and BCB – are mainly used. For the bracket type KL flanged (KLK) and unflanged (KL) brackets are used.

Table 4: Statistical Breakdown of Bracket Types

Type	KL	KL (flanged)	BCW	BCB	Other (6 Types)
Number	95	60	51	37	56

From a standpoint of standardization only the bracket types KL, KL with flange, BCW and BCB are relevant. The other types are used sparsely. The implementation of a design solution for the latter types is neither economically sensible, see above, nor technical feasible. The later instances are only used for special cases.

Based on an initialized working memory the profile-plate-intersection for which a bracket should be defined, is selected by the user. For this problem the context is determined and suitable brackets are selected from the standards database. A scoring algorithm is applied to find a solution with optimal performance. In case that multiple candidates found, a manual interaction is required to select the best solution. Finally, the bracket is defined for the location selected . The working memory is reset i. e. all temporary variables are set to an initial state. The next design problem can now be identified and solved

Auxiliary Parameters in the Data Model

For the selection of suitable i. e. valid brackets additional parameters are introduced, namely:

- Activity (Steel Model): The activity parameter defines whether an object in the structural model is active, involved or inactive. For the example of a bracket-definition, the stiffener and the plate for which the bracket should be defined are active. If a part influences the design process developed for the active parts this part is marked as involved by the rule system. E. g. neighboring stiffeners could have an effect on the result of a design activity. By default all objects in the working memory are set to inactive.
- Performance Score (Standards Components): The performance score is a compound parameter. Mandatory and optional performance parameters are distinguished. If a mandatory performance parameter, e. g. with respect to strength requirements, is not met, this standard component is ignored for the current design problem. Optional performance parameters are used to evaluate the performance of a design solution with respect to soft constraints. Manufacturing costs, ease of installation or the exploration of series effects can be named. Different weight factors can be assigned to each optional performance parameter.

Bracket Selection & Evaluation

As an unambiguous assignment of bracket instances to profile types and dimensions exists, all non fitting brackets are sorted out:

```
WHEN
        $profile :          Profile (activity == active)
        $bracket:           Bracket(validFor not $profile)
THEN
        $bracket.setValidity(invalid)
```

In the conditional clause the active profile is selected from the data model. A temporary variable $profile with rule scope is created. In the second line it is tested whether the standards definition is not applicable for the active profile. If both tests are true, the consequence is executed. The bracket instance is marked as invalid and is not used in other rules.

For a single profile type and it's dimensions in most cases flanged and normal brackets can be used. If space restrictions are present, the flanged solution is preferable. Otherwise, due to lower manufacturing costs, the unflanged bracket type is used. Here, SpaceManager is a knowledge component where space requirements, the functional aspects of certain rooms etc. are defined. Currently, this is defined as an external component where all relevant information is entered manually. In the future an integration into the product data model of the CAD system used can ease data management and facilitate the use of this knowledge.

```
WHEN
        $profile:               Profile(activity == active)
        $plate:                 Plate(activity == active)
        $bracket:               Bracket(validity not invalid; hasFlange == false)
        SpaceManager.isRestrictedStrict($profile, $plate)
THEN
        $bracket.setValidity(invalid)
```

If a strict space requirement for the combination of plate, profile is given, all unflanged brackets are marked as invalid, see the first rule below. In contrast, a soft constraint defines a preference for a flanged solution. The scoring of the flanged proposed solutions is increased. The score for unflanged candidates is decreased.

```
WHEN
        $profile:               Profile(activity == active)
        $plate:                 Plate(activity == active)
        $bracket:               Bracket(validity not invalid; hasFlange == true)
        SpaceManager.isRestrictedSoft ($profile, $plate)
THEN
        $bracket.updateScore(space, 1)

WHEN
        $profile:               Profile(activity == active)
        $plate:                 Plate(activity == active)
        $bracket:               Bracket(validity not invalid; hasFlange == false)
        SpaceManager.isRestrictedSoft ($profile, $plate)
THEN
        $bracket..updateScore(space, 0)
```

Based on these and other rules not shown a context sensitive configuration of appropriate brackets can be identified. The selection of a solution with optimal performance is performed as follows:

```
WHEN
        $bracketA:              Bracket(validity == valid)
        $bracketB:              Bracket(validity == valid)
        $bracketA.getScore() > $bracketB.getScore()
THEN
        $bracketA.setSelected(true)
        $bracketB.setSelected(false)
```

If a valid bracket has a higher score than another valid bracket candidate from the working memory, this bracket is marked as selected. The bracket with a lower scoring level is marked as not selected. This rule is called repeatedly until all possible configurations are tested and hence the optimal solution is found. In case of multiple solutions with identical performance scores the user is asked for a selection. Using similar approach different rating criteria could be implemented.

Bracket Definition

The definition, i. e. the placement of the suitable bracket found is currently performed in the knowledge model only. An

instance of the bracket solution is created for the current problem domain and added to the working memory. The type of the solution is passed as argument.

```
WHEN
        $bracket:               Bracket(selected == true)
        $profile:               Profile(activity == active)
        $plate:                 Plate(activity == active)
THEN
        $bracketInstance($bracket)
        $profile.connectedTo($bracketInstance)
        $plate.connectedTo($bracketInstance)
```

Depending on the angle between plate and profile either an inline placement or an attached placement is used, see Figure 6. It is tested whether the bracket supports the placement preferred.

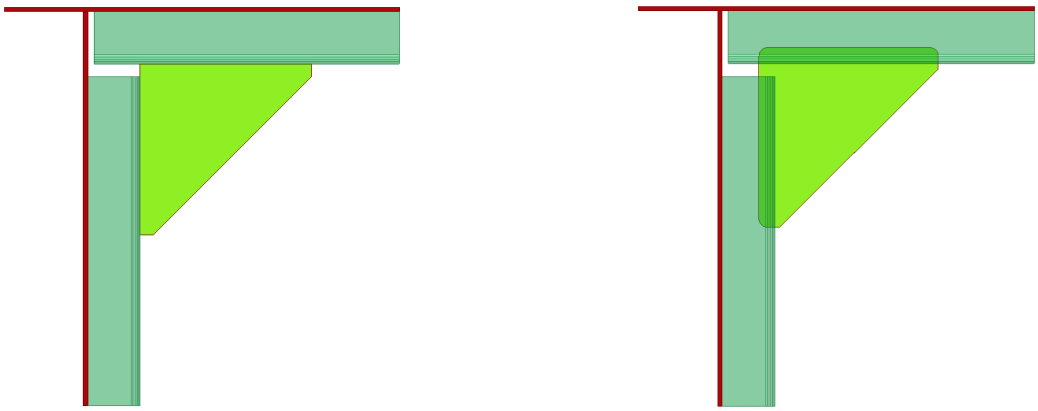

Figure 6: Inline or attached placement of Brackets

The following rule shows the treatment of a bracket that supports attached placements.

```
WHEN
        $profile:               Profile(activity == active)
        $plate:                 Plate(activity == active)
        $bracketInstance:       BracketInstance()
        $angle:                 GeoTool.getAngle($plate, $bracket)
        $angle < 90
        $bracket.supportsPlacement(attached)
THEN
        $bracket.setPlacement(attached)
```

As the scope of the research is placed on the evaluation of rule based methods for the determination of solutions, no direct interaction with the Tribon CAD system is given. A manual transfer of the solution achieved is required. In the future it is planned to extend the API mentioned above. With this approach a fully automatic definition of brackets etc. using a rule based approach can be achieved.

Reset

Finally, the complete working memory is reset. All temporary value like activity and selection parameters are set to the initial state. The scoring compound parameter is cleared.

```
WHEN
        $bracket:               Bracket()
THEN
        $bracket.setSelected(false)
        $bracket.SetActivity(inactive)
        $bracket.resetScoring()
```

DESIGN PATTERN

Design patterns are focused on the consequent execution of multiple design sequences. Two different types of design patterns can be distinguished, namely:

- the multiple execution of a single design pattern
- the execution of a sequence of different design patterns.

Multiple Execution

For the first case, the bracket definition as shown in the previous chapter is extended. The problem to work on, i. e. the profile and plate to evaluate, are selected automatically. An iterative approach allows for the fully automatic definition of all brackets within the knowledge model, see Figure 7.

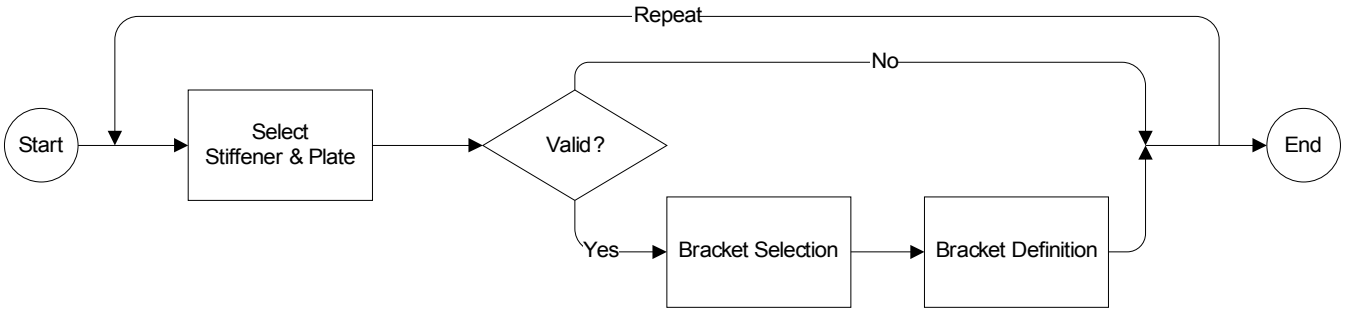

Figure 7: Automatic Definition of Multiple Brackets

In a first step possible locations for bracket placements are determined. Two intersection profile and plate elements are identified and activated.

```
WHEN
        $bracket:               BracketInstance()
        $profile:               Profile()
        $plate:                 Plate()
        $plate.intersectsWith($profile)
THEN
        $profile.setActivity(active)
        $plate.setActivity(active)
```

This configuration is evaluated. Yard standards – e. g. every second stiffener is connected to a plate with a bracket – are adhered. In the example below strength considerations, if applicable, are evaluated using a rule based sub module called via the StrengthManager component. In this component a subset of the GL class rules is modeled. A global stress and load model of the ship structure can be accessed if present. If no action is required the next configuration is tested.

```
WHEN
        $profile:       Profile(Activity = active)
        $plate:         Plate(Activity = active)
        Not StrengthManager.needsBracket($profile, $plate)
THEN
        Reset & continue

WHEN
        $profile:       Profile(Activity = active)
        $plate:         Plate(Activity = active)
        StrengthManager.needsBracket($profile, $plate)
THEN
        Continue with Bracket Selection
```

With this approach the atomic design entity “bracket selection and definition” is performed repeatedly. A consecutive

placement of all brackets in an activated domain is performed.

Complex Design Patterns

Using the design activities shown exemplary above, complex design patterns connect the execution of different design patterns. E. g. in Figure 8 the definition of details for a floor plate structure is shown. Based on the basic steel structure, information about strength and stress requirements as well as knowledge about the function of the compartments separated by the floor plates, different design activities are selected and executed.

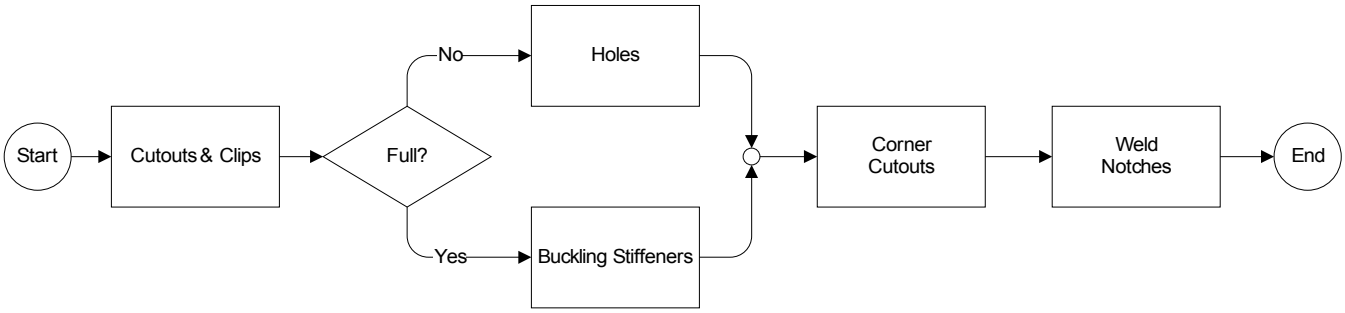

Figure 8: Complex Design Pattern for a Floor Plate

Depending on the room configuration either a full watertight floor plate structure is defined. Also, yard and class society regulations are taken into account. E. g. the regulation that every third floor plate should be designed as full floor plate structure is regarded. For each floor plate, cutouts are defined for intersection stiffeners. For watertight tank walls adequate clips are selected. The connection of the stiffeners to the plate structure is defined so that an adequate cross section for load transfer is given. For lightweight floor plate structures holes are defined according to the yard standards. The distance from the edge of the hole to the edge of the plate is controlled. The corner radius is determined. For full floor plates buckling stiffeners are defined on the plate. Corner cutouts as well as cutouts over weld seams are defined.

As a result for a standardized context the detailed design of a floor plate structure is performed automatically. In case of ambiguities the expertise of the user is asked for.

CONCLUSIONS

A knowledge-based approach to detailed steel structural design can be used for quality controlled semi automatic design. The workload of the engineer is eased. More time can be spent on innovative work. Series effects can be explored. The use of solutions with optimal performance can be achieved. As shown in the previous sections the layered, bottom-up approach allows the reuse of individual design activities as different levels of complexity. Depending on design problem given a suitable level of automation can be chosen.

Today, integration with existing CAD systems is time intensive and requires a good understanding of programming techniques. Often, different data sources like a global strength model, compartmentation plans etc. are not connected. Data integration needs to be performed manually. A more integrative approach would help to reuse the knowledge inherent in these information sources throughout the complete design process. Changes could automatically be propagated throughout the product model.

Knowledge acquisition and knowledge explication are difficult. Often, existing yard standards or class regulations are not formulated clearly. Dependencies on the context are not always given. A formulation of a standard that is suitable for the implementation of a rule based system can be difficult to obtain as the naval engineer lack the relevant understanding of such systems. The unambiguous definition of the design activities has proven to be a problem. An adaption of existing knowledge based engineering standards to changing requirements on the shipyard is therefore cost and time intensive.

REFERENCES

Hubka, V. & Eder, W. *Design Science*, Springer, 1993

Nieuwenhuis, J. and Nienhuis, U. *Knowledge and Data Reuse in Ship System Design,* International Conference on Cimputer Applications in Shipbuilding, 2004

Pierra, G. *Context-Explicitation in Conceptual Ontologies: PLIB Ontologies and their Use for Industria,l* Data Journal of Advanced Manufacturing Systems, 2006, 5, 243-254

Sim, S. K. & Duffy, A. H. B .*Towards an Ontology of Generic Design Activities,* Research in Engineering Design, 2003, 14, 200-223

Wielinga, B. J. & Schreiber, A. T. *Configuration design problem solving,,* Special issue on AI and design, 1997, 49-56

Zimmermann, M. et. Al, *Knowledge based engineering methods for ship structural design,* International Conference on Computer Applications in Shipbuilding, 2005