Validation and Quality Control of Design and Production Information – Applying Rule Based Data Mining and Business Intelligence Concepts to Engineering

Thomas Koch, Atlantec Enterprise Solutions GmbH, Hamburg/Germany,
thomas.koch@atlantec-es.com

Abstract

Design and production preparation tasks for ships and offshore structures generate large sets of data providing every detail needed for planning, procurement, and production. Many different participants and systems are involved in this very dynamic process with high modification rates. There is a constant risk of errors being introduced.

We describe how a combination of state-of-the-art data retrieval and rule-based analysis techniques to validate engineering design and production data has been employed to reduce the risk of costly errors often uncovered very late in the production process. A rule based approach helps to protect customer know-how and provides a high degree of flexibility and sophistication.

1. Introduction

The design and production of ships and offshore structures requires creation of large amounts of data which are used to define the engineering design and eventually help generate production data. It is a well known phenomenon that the cost of recovering from errors introduced early in the design process grows non-linear as production progresses. It is therefore quite useful to try to detect such errors as early as possible.

The widespread use of software applications in design clearly has helped to reduce the frequency and mitigate the effects of traditional "human" error introduced during design. Nevertheless, some categories of errors still remain, and, unavoidably, additional possible sources of errors have been added. These are commonly encountered typical errors:

- software induced errors (e.g. failure to update values after changes, failure to save modifications)
- data transfer errors
- numerical precision issues
- incomplete data
- non-compliant design decisions (intentional or unintentional)
- wrong input or configuration
- missing or late information

An analysis of commonly encountered errors in different shipbuilding environments has shown that a major part of all errors remains undiscovered until production, tests, or even sea trials mainly because of the un-validated transition of data through departments, sub-contractors and other partners involved. Explicit validation however is very time-consuming and usually quite costly. Therefore, validation is only performed when absolutely required or when quality issues are already known.

The research work described in this paper is based on the assumption that it should be possible to reduce the frequency of errors due to constantly applied quality validation procedures and by shortening the feedback distance (between discovery and origin of an error). To reduce the cost and time needed, these validation procedures should be automated as much as possible. At the same time it should be practicable to apply validation after any process step as required.

2. Sample Scenarios

The typical product development process for ships can be depicted in a simplified form as shown in figure 1. It shows the principal flow of information between stages like early design and detailed design. In addition to the pure data flow perspective there is the increased partitioning of the design tasks as the process moves from global design activities to more and more refined tasks. So, as a side effect, more and more parallel "threads" of activities are created, which are performed in a quasi-concurrent mode (i.e. a mixture of sequential and parallel tasks). This also increases the potential for change requests being exchanged between those tasks, which always bear the risk of some part of the data becoming de-synchronized.

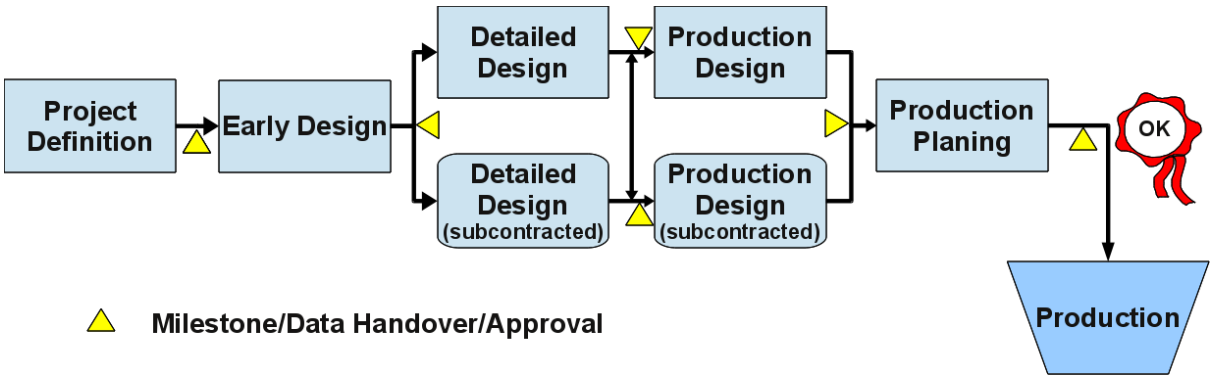

Fig.1: Common design/production data flow in shipbuilding projects

The easiest way to create a quality control structure is to install hand-over milestones. This is what traditionally is being done in the form of some internal/external approval process, e.g. by signing of a set of drawings or by receiving some sort of more or less formalized statement of approval, e.g. from an owner or a classification society.

It seems plausible to use exactly these milestones as initial QC checkpoints. Therefore the goal is to inject a computerized QC step just ahead of the regular approval processes as well as to inject additional QC steps around every data hand-over. In the case of subcontracted design, the QC step may occur before delivery (preformed by the subcontractor) as well as after receipt (performed by the customer).

In the longer run, the execution of QC steps should also be possible at any point during the execution of a design task up to the point where a designer may use this as a regular checking tool. Again, by detecting errors at the earliest point in time leads to reduced rework within a single design task. Compared to system-embedded QC functions, this external review procedure will provide the possibility of checking against other external data e.g. to verify correct interfaces.

Based on investigation of some real-world problems at shipyards, we have selected two scenarios that exemplify the quality control problem quite well.

2.1. Design data transfer from subcontractor to shipyard

Subcontracting of engineering work has been in widespread use in shipbuilding for many years. Subcontracted work is a clearly identifiable point of data handover. It often results in CAD models, CAM data, drawings, part lists etc. Subcontractors will usually be required to perform some pre-delivery checking. Similarly, shipyard customers will perform some sort of checking after receiving the data from subcontractors. Due to the potentially complex mixture of data, this checking process can be very costly and may only be based on random samples.

2.2. Release of production data to production

This scenario is part of the mainstream shipyard data flow. Once design data (including production specific information) has been released to production, any further change or correction will be potentially very costly. This is another critical data handover as the next stages of processing will produce large amount of derived information, creating a lot of dependencies on the original design data.

3. Validation Processes

All design processes include some sort of validation activity. Regularly, the current state of detail is communicated to project participants like owners, authorities, suppliers etc. In engineering projects validation will occur in many ways, but most of these activities are traditionally driven by high level considerations like acceptance of product layout, features, and performance, compliance with rules and regulations, or to provide technical conditions and constraints to suppliers. In addition there has, however, always been a fairly substantial degree of process-internal checking, e.g. like reviewing, signing off, and publishing of drawings for design units like steel blocks, piping systems etc.

To check the correctness and validate design or production data, it is necessary to have full access to the detailed data resulting from the design activities. Usually early and detailed design data is produced in CAD systems. It is quite possible that a single design involves multiple design software systems, either to cover different phases of the design activities (like during early vs. detailed design) or due to organizational constraints, e.g. different systems being used by different departments or subcontractors. For production preparation tasks, a wide range of CAM systems may be applied covering tasks like NC data generation, detailed scheduling, and work content estimation. Closely linked to this phase are production planning activities ranging from pure scheduling and resource planning to detailed production control tasks.

Usually, the execution of the validation process will occur by means of manual/interactive checking done by design staff/management or reviewers, either using the originating design system or working with derived information like drawings, lists, spreadsheets, and the like. This way of checking addresses high level aspects like fulfilment of the requirements and specifications, compliance with approval, corporate, or project-specific rules, or application of best practice. At the same time, reviews will be used to detect lower levels like modelling or drawing errors, but based mostly on random samples, experience, or by coincidence.

Increasingly, some of the more sophisticated applications like CAD systems are offering additional modules or functions to perform validation tasks within these systems. This helps to detect system-specific design errors, once validation constraints or rules have been defined. It makes it possible to perform regular QC validation tasks in the current design context. However, the application of these validation tasks is limited to the specific system. This results in major limitations both regarding the scope of validation (no possibility to check data combinations from different systems) as well as detecting data handover problems of any type.

Hence it is desirable to utilize automated methods to provide a better coverage of checking. For this type of automated validation it is a requirement to have access to all the data needed (which would in most cases originate from different data sources) and to find a means to formulate validation conditions/rules on a broad scale.

Automated validation may occur on many different levels:

- data correctness and consistency
  - data formatting/validity
  - data sufficiency or completeness
  - data consistency/inconsistency resolution

- design rules
  - fundamental design rules (to ensure compliance with fundamental principles, e.g. no free-floating parts, no collisions/intersections of rigid bodies)
  - classification and statutory design rules (to avoid trivial/obvious deviation from authoritative regulations)
  - corporate design rules
  - project-specific design rules

Data correctness and consistency checks are used to detect data corruption, unintended or unauthorized manipulation, or low level software errors.

Checking design rules encompasses the major bulk of validation tasks. Such rules require higher semantic expressivity than data correctness conditions.

In such a QC setting it should also be possible to work with different versions of data. This will help identify revision issues, allows discovering unexpected changes, and will provide historic information to track down the cause of such issues. As the automated QC process can be rerun at any time, complete validation can be (re-)done many times, therefore guaranteeing complete checking against all defined rules.

4. A System Architecture for Automated Quality Control

To describe the system architecture of our QC system, we first consider the QC process before discussing the system components and their relationships to the system environment. The process can be structured into 4 major steps, as shown in Figure 2.

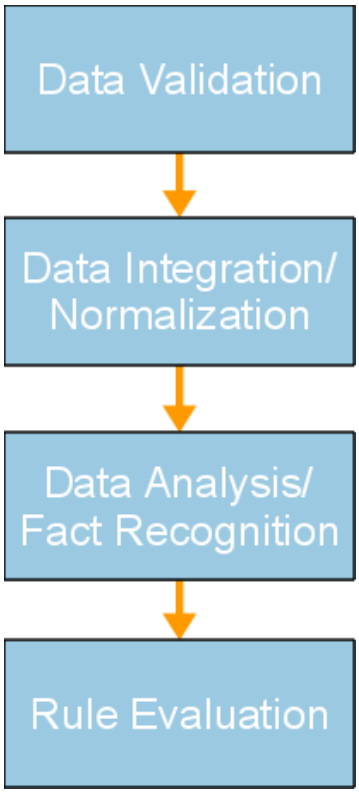

Fig.2: QC Process Flow

The first step is ***data validation:*** to perform quality control of engineering data, it is key to work with data taken straight from the current revision of data intended to be used in further processing. For example, before handing over design data to production planning, a QC check would prevent errors from migrating into production or planning data. To accomplish this, close integration is needed with those systems that are used to generate and/or maintain such design data. Integration in this context means that a most complete collection of the data should be accessible during the checking process.

In the majority of cases it will be necessary to collect data from different data sources which can use quite diverse data structures to represent partially similar or overlapping information, creating a data structure set combination as depicted in figure 3.

The data validation process step will include actions to remove duplicate information, to consolidate overlapping

information from different sources, and elimination of obsolete or invalid data. Data sanity checking at this level may lead to a premature finalization of the whole quality control process, e.g. if some data file has been substantially corrupted.

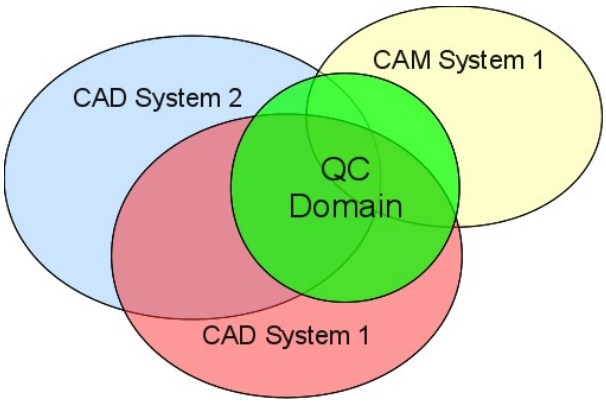

Fig.3: Overlapping data from different systems vs. interesting subset

The key task for the next step ***data integration*** is the merging of different data sets into a single normalized data representation. Most importantly, it will also normalize the data structures such that following steps will be able to work with a single set of meta data, i.e. a single data model.

Once the data has been integrated into a single data representation, the ***data analysis*** activities can be applied as the next step. These activities will primarily aim at (re-)constructing a ***fact base*** from the data available. Constructing a fact base can mean simple actions like ***selecting*** objects of interest from the integrated data set. For example, if a CAD structure with assemblies and parts has been provided, depending on the objective of the checking activity, only parts may be considered for checking. More complex activities would try to ***derive*** additional facts from existing data. A simple example would be to calculate the weight of a part based on the geometry and material properties provided. However, far more complex operations may be performed at this stage. This can include the creation of new individual data items representing their own set of properties. For example, welding information may be inferred from part connectivity and edge preparation details. Derivation may actually be deferred in some cases, where the need to have this information available cannot be determined in advance and the computational effort is high.

Once the data preparation activity has been completed, a ***rule evaluation*** activity will be performed, to perform the full-scope quality validation. Rule based systems like *JBOSS (2009)* or *FRIEDMAN-HILL (2003)* provide a number of capabilities that seem an excellent fit for the quality checking problem: they use declarative rule formulations which do not require the user to pre-determine some flow of execution. Instead the user can focus on describing which conditions need to be matched in order to fulfil a certain rule. Rule definitions can be expanded over time by end users as needed, thereby allowing large systems to be built and maintained over time. This effectively means that a quality knowledge base can be established in an organization at a convenient pace. The result of the rule evaluations will be collected and processed to provide a report about the checking process result.

This process has been implemented in a system as shown in figure 4. The sample CAD and CAM applications are representing an existing design and planning environment. These systems are used without any modification.

To execute a QC process, only a few parameters need to be defined by the user:

- the set of original data sources and optional sub-selection filters (e.g. the steel block(s) or a outfitting module(s) to be checked), and

- a collection of rules and/or rule sets to be used for checking needs to be selected to define the extent of checking (called a rule/rule set configuration).

The data validation and data integration steps are performed by different ***publisher*** modules, which provide the capability to retrieve data from the specific target system (CAD 1, CAD2, CAM …). All these modules are implemented using the Topgallant® Adapter Framework , and convert the incoming data according to a single normalized data model, the Topgallant® Enterprise Reference Model (ERM), *AES (2009)*. This data model is built on international and industry standards and forms a superset of the typical data models found in CAD, CAM, and planning systems, thereby capturing all available information, *ISO 214 (2004), ISO 215 (2004), ISO 216 (2004), ISO 218 (2004), ISO 227 (2005)*. All further processing in the system will occur using this data model.

The normalized data representation is next processed in the data analysis step. To generate additional fact data, this step is directed by the requirements of the selected rule set(s). Every rule defines dependencies on specific types of facts exclusively via its condition clause (see sec. 7). It is therefore possible to activate any matching fact-finding algorithms (i.e. algorithms which will potentially produce facts of the required type) based on these requirements. For example, a rule to check the compatibility of edge preparations on parts to be welded together will look for specific edge feature objects in the fact base. These edge feature objects may be constructed from geometrical properties of a part's geometry.

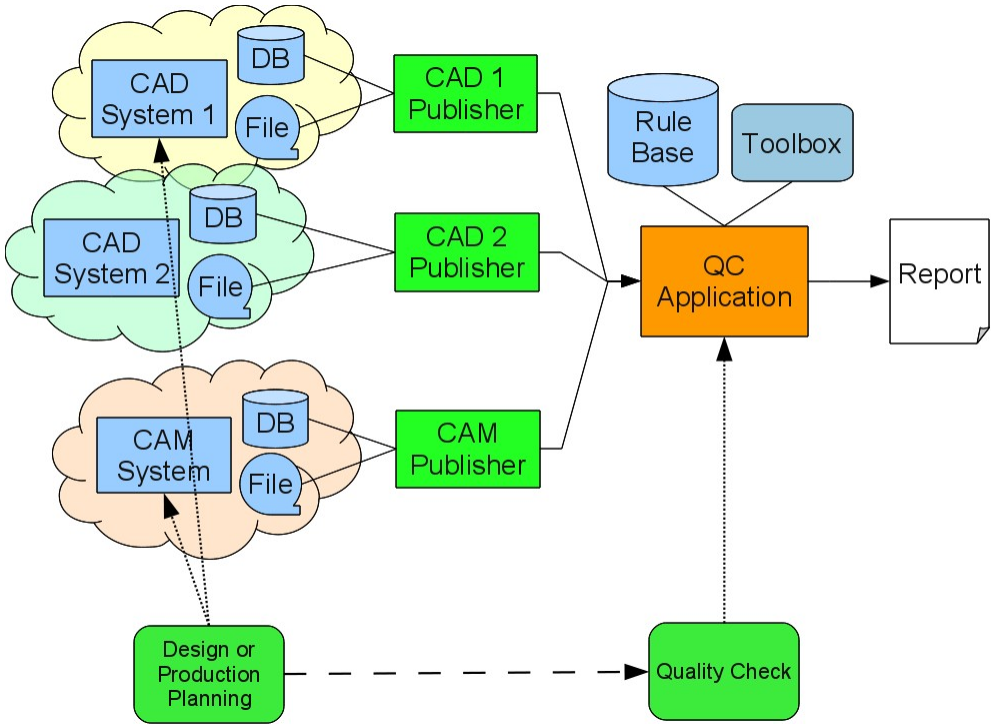

Fig.4: System Architecture

Once the fact base has been established, the main rule evaluation will take place. It needs to be supported by a library of methods and algorithms, some of which may be computationally complex. These methods are collected in a toolbox library, which makes them available to rule definitions. For example, geometry algorithms like area or volume calculation, transformations, centre-of-gravity, shape comparison etc. are provided in a geometry toolbox. The toolbox mechanism needs to be extensible to add more methods as needed. This provides the flexibility necessary to cover more and more topics with rules.

A central foundation of the QC application will be a rule base, which can be created and/or expanded by the user/organization. Is essentially a database of rules organized into a hierarchy of rule sets.

5. Data Integration and Data Fusion

During the data integration phase, a data warehouse approach is used to combine data from different sources into a single, normalized representation, thereby eliminating duplication and identifying conflicting data. This is accomplished by an ETL process (extract, transform, load) provided by the Topgallant® adapter framework. The integration step includes data quality checks and can therefore already lead to QC error reports, which will help to identify inconsistent data items that are required to be in sync.

The storage mechanism used provides full support for versions of data items. It helps to identify data modification during all stages of the QC process and to include history records in the fact base and thus in the rule evaluation strategy. For example, unchanged, approved data may be exempted from some rules during future checking stages. The details depend, of course, on the actual dependencies of the rules.

The specific challenge during the data integration is the requirement to deal with "fuzzy" data. This effect can occur on all levels, the simpler examples being numerical round-off deviations. Therefore, for example, geometric analysis methods have been equipped with numerical tolerance parameters that can be modified dynamically during execution. This way it is possible to take into account the reduced numerical resolution of some data sources, without generating a large number of error reports due to more or less trivial numerical differences, which are generally acceptable. This approach can be further extended to deal with unavoidably deviating data items by employing data fusion techniques like probabilistic indicators. In the system discussed here, resolution policies similar to the approach described in *ANOKHIN and MOHTO (2006)* have been implemented. All data flagged as conflicting is sent through a fusion pipeline, in which a series of ***elimination*** steps is followed by a ***fusion*** step. The elimination steps will be used to remove duplicates or contradictory values based on a selected resolution policy like selection of attribute properties like version, availability, or minimum, maximum, average value. Since this will potentially still leave multiple values to select from (particularly if multiple attributes of a single data item are conflicting) the fusion step will then pick the final resolution value based on distribution parameters or selection patterns.

6. Data Mining Techniques

Data mining techniques have been developed over many years, primarily to "discover" implicit facts in large, fragmented data collections. The main application domain has been for systems used for administration, finance, or sales and marketing, *HAN and KAMBER (2006)*. Data mining is used to discover relationships and structures between data items that are not known in advance and are not expressed explicitly in the data model(s) being used in the target data store(s). Data mining thus has a close relationship to knowledge discovery or machine learning. It seems useful to apply techniques developed in this field to other areas like engineering data analysis.

In the context of a QC application, data mining is particularly useful to support the attempt to ***re-establish*** links and relations in data sets that are expected to exist (e.g. due to technical requirements) but are not explicitly modelled anywhere in the available data. For example, a designer will have various design intent aspects in mind when defining a steel structure. While creating a CAD model of such a structure, a lot of that high level information will be lost and may not be documented in the stored data set but result in data elements like the actual geometric shape of parts that reflect a rather implicit representation of the original intent. Therefore data mining techniques may help to re-establish or derive facts from existing data that can be critical to validate the expected quality conditions.

Data Mining involves a wide range of analysis methods with different degree of complexity. The key method areas commonly used are various statistical analysis algorithms and pattern matching. Pattern matching is a very powerful mechanism particularly for discovery of associations between data items. For example, based on a certain feature tree linked to a part, it is possible to assess facts about the connectivity between different parts. Similarly, geometry relationships will allow deriving spatial or topological associations, e.g. a part being located in a compartment, being attached to a wall etc. In the QC application we have also implemented a number of classification algorithms to determine specific assembly types like `T-beam` or `Micro-panel` based on associations. The system is therefore able to discover specific configurations which can then trigger suitable rule evaluations. Along the same lines, failure to discover such associations may also be a relevant input to rule evaluations.

Another interesting result of data analysis is the identification of ***outliers***, which represent untypical data items. Outliers can be submitted to additional checking, as their occurrence frequency is expected to be low. Therefore the likelihood to detect a design error will be higher but it may be that the special properties are fully justifiable, which can be proven by additional rule checks.

Outliers detection is a result of cluster analysis applied to the data set. Discovery of clusters in data attribute distribution can provide a lot of additional insights – from the quite obvious geometrical interpretation (e.g. all parts of a bulkhead form a geometric cluster) to the more advanced uses like welding dimension control, where material thickness would be related to welding dimensions by a weighting function.

7. Rule Based Processing

Rule based systems are another technology commonly used for knowledge processing. The traditional architecture of a rule processing system involved the ***fact base*** (also called working memory), the ***rule base*** (also called production memory), and an ***inference engine*** (also called rule engine). The system is configured by defining rules in the rule base, e.g. by using a rule editor application. To process concrete data the fact base needs to be initialized by inserting data items e.g. from external systems like databases or modelling systems. While the engine is executing, new facts may be derived and inserted into the working memory.

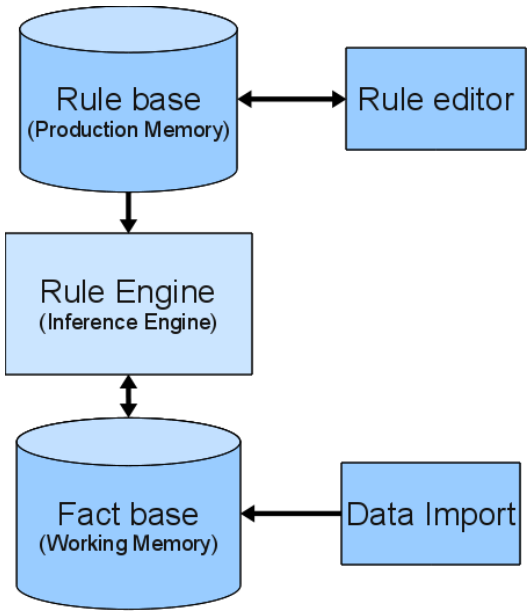

Fig.5: Rule Engine components

Rule processing systems differ from conventional programming logic as they are working based on declarative rules. This means that there is no fixed flow of processing (like: check this first, then do this, then do …). Instead, rules define conditions that are evaluated based on a scheduling algorithm implemented in the inference engine, which will act upon fulfilment of conditions.

A ***rule*** is a relatively simple construct to describe a set of ***actions*** that should be applied if a specified set of ***conditions*** is matched by the facts found in the working memory. The basic structure of a rule looks like this:

```
rule "title" [<options>]
      when <conditions>
      then <actions>
end
```

The **when** clause states conditions that need to be satisfied in order to ***fire*** the rule. Firing a rule will result in its actions list described in the **then** clause to be carried out. Actions can be anything, but typically they will either establish new facts or revoke invalided facts, or record some state, or send some message. As part of actions, any kind of computation may be performed.

Rules can be grouped in ***rule sets***, which will provide a logical collection of rules. For example, some rules might be used to assert certain facts (thereby inserting newly constructed data items as derived facts in the working memory), which will then in turn be subject to further rule evaluation. Rule set can again be included in other rule sets, which subsequently build up complete rule set libraries (see figure 6).

It is also possible that rules within a rule set share common resources like parameter settings among all rules in a rule set. Rule sets also provide a convenient means of grouping or classifying rules by topic. Rule sets can be nested to any depth, which provides a flexible way of organizing a large collection of rules.

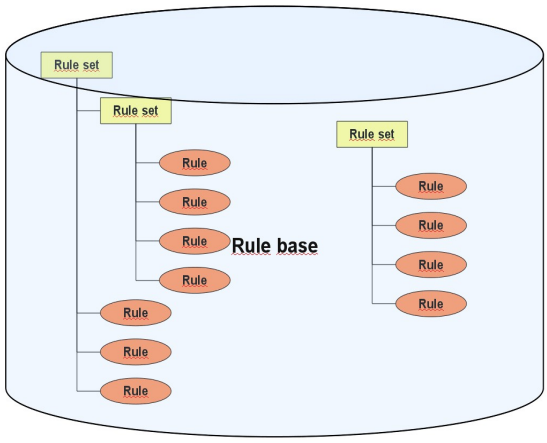

Fig.6: Rule base structure

Having stated that rule systems are declarative, there are still situations where rule evaluation should follow some sort of sequence pattern. For example, it may be desirable to evaluate light-weight checks first to improve overall performance. Or, it may be useful to define several processing stages that should not necessary overlap, as each stage may produce new findings (ending up as new facts in the fact base), which in turn would trigger re-evaluation of other rules, that may have been evaluated before. The solution to this is to assign a rule priority, which tells the system to prioritize rules with a higher setting. It should be noted, though, that the priority mechanism is particularly useful for optimization, rather than bringing in procedural flow logic "through the back-door".

8. Prototype Implementation

The QC application has been implemented as an interactive system that provides all functions to establish a rule base, load a fact base from different connected data sources like CAD or CAM systems, and to perform data analysis and evaluation of a selected rule configuration.

As described above, quality control rules fall into different categories. Whilst the lower level conditions are concerned with data integrity and transmission fault detection, and are therefore somewhat static, the higher level design rules involve a great deal of engineering knowledge. Following the architecture described above a rule base needs to be built that reflects the different types engineering rules and constraints to be checked. Most important is the fact that these rules often must take into account corporate or project-specific rules that must be enforced. Therefore, a QC environment has to provide users with the possibility to establish (i.e. define/write) such rules.

The figure 7 shows the rule editing view of the application. The tree display shows the structure of the rule set hierarchy and any rules contained in it. The form on the right hand side shows a sample rule as it is displayed during rule definition.

The editor component is required only when creating new rules or modifying existing ones. It can be disabled for workplaces that require quality checking but no rule development.

The data retrieval sub-system consists of dynamically pluggable adapters. For the test environment, publishers for the Tribon® M3 Hull system, the Unigraphics® NX system, and AutoCAD® DXF have been used. In addition, output from the Alma act/cut automatic nesting system has been connected as a sample CAM application data source.

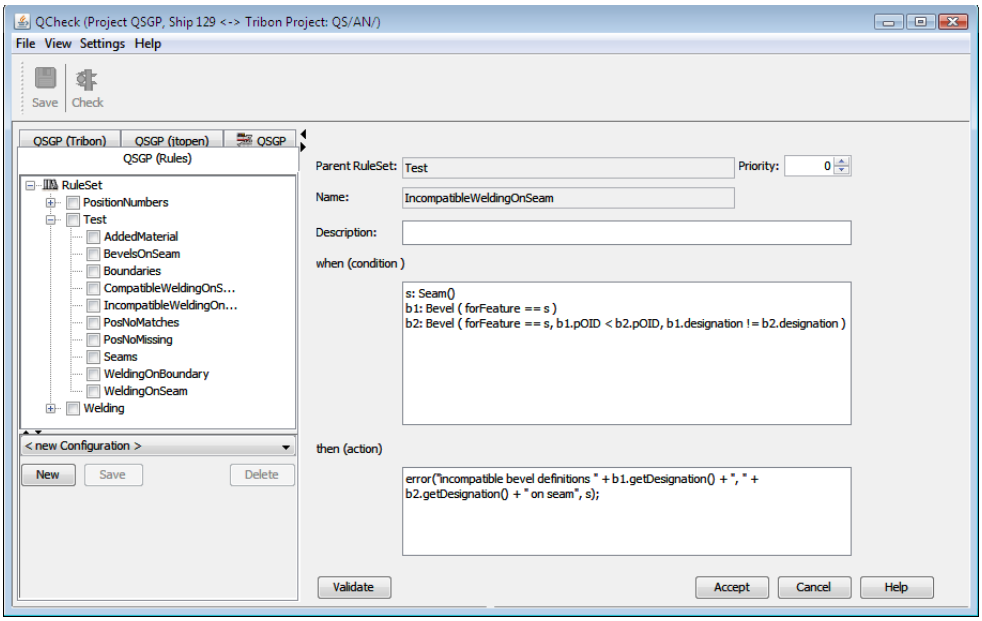

Fig.7: Rule Editor

To access CAD data from a sample project, the system specific publisher view will provide access to the CAD data store, providing navigation through the design structure. By selecting parts of the structure (e.g. a block), all CAD data items in that structure will be included as input data.

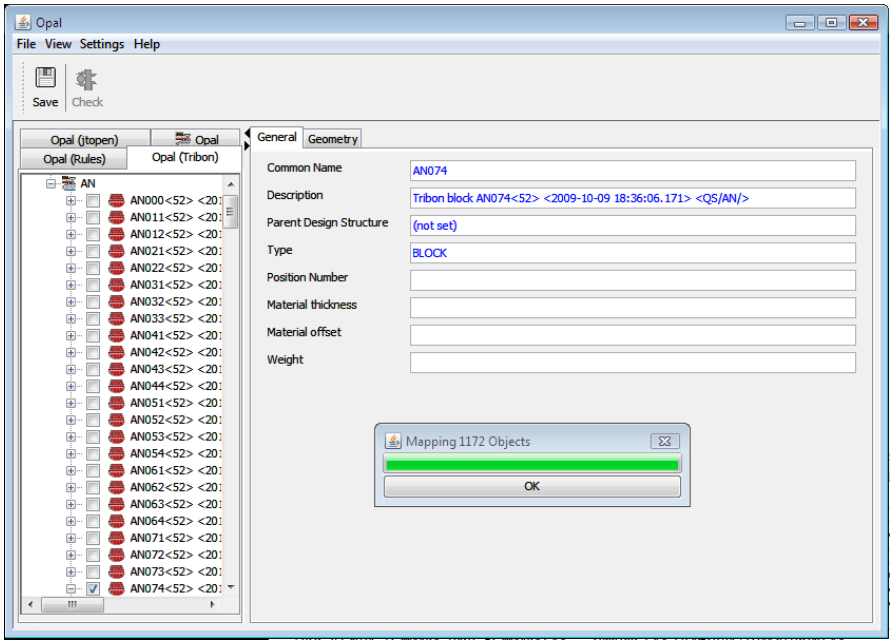

Fig.8: Loading fact base from CAD data

Once the fact base has been loaded, rule configurations can be selected and the checking procedure is performed using the “Check” function. Validation errors or warnings will be indicated in the tree, a summary of detected problems can be searched, and a more detailed report can be saved as a PDF or spreadsheet document.

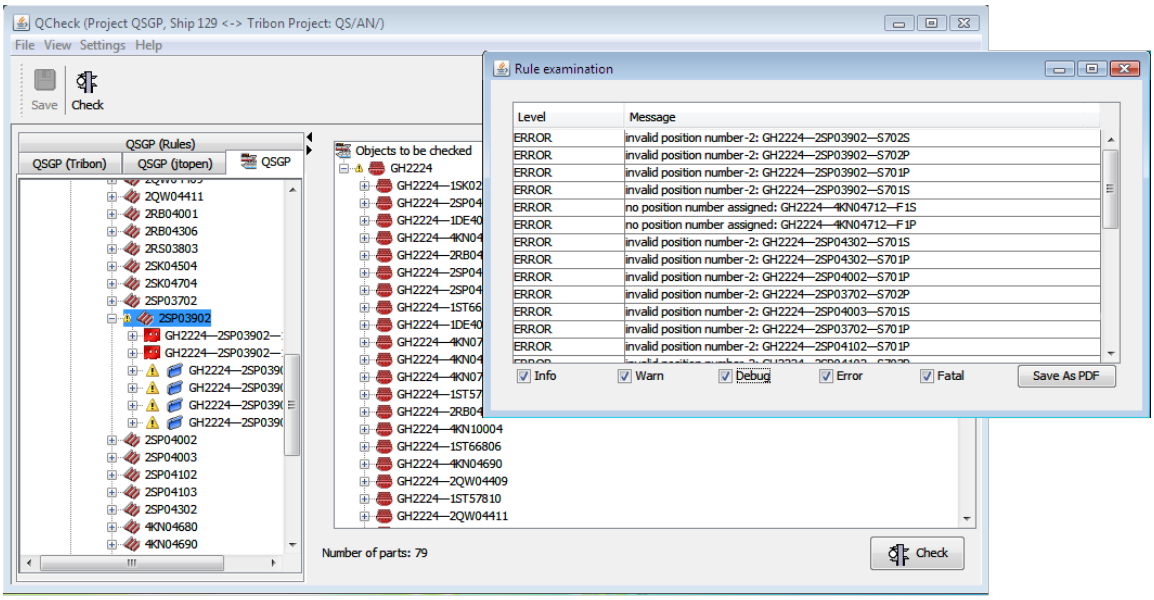

Fig.9: Rule Evaluation

9. Summary and Conclusion

The approach to validation and quality control for engineering and production data described in this paper has proven to be effective and useful. The goals defined for this development have been reached. The mechanism chosen for formulating rules for engineering data validation has a large potential. The brevity and conciseness of rule formulations is quite promising and can be even further developed to be highly domain oriented. The combination of computer science tools such as geometry processing or numerical analysis techniques applied to engineering with combined with data mining and business analytics tools on the business side will provide a platform for powerful validation tools.

The essential prerequisites for applying such a technology are a well-established method of accessing a wide variety of data sources combined with the data integration and data mining/analysis capabilities as described above. Only the fully attributed set of data items provides the platform for non-trivial checking tasks.

Further work is, of course, desirable. For example, the data analysis tasks can be further extended in deal better with uncertainty. The whole area of fact/knowledge discovery algorithms holds many more options for sophisticated processing of fuzzy data. Advanced user support for formulation of rule definitions is needed to lower the entry barrier and training effort needed.

The integration of external validation actions seems to be another route to expansion. Using the toolbox mechanism described, external heavy-weight verification tools like numerically more intense strength analysis, production simulation, or NC data processing simulation could provide powerful tools to verify produce-ability of designs.

10. Acknowledgement

Parts of the work described in this paper have been supported by the German Federal Ministry of Economics and Technology (BMWi) under grant 03SX230D.

11. References

ANOKHIN, P; MOTRO, A (2006), *Fusionplex: Resolution of Data Inconsistencies in the* Integration of Heterogeneous Information Sources, in: Information Fusion, Volume 7, Issue 2, pp. 176-196, ISSN: 1566-2535, Elsevier Science Publishers B.V., Amsterdam.

AES (2009), *Topgallant® Adapter Development Kit*, Documentation Rev. 1.5, Atlantec Enterprise Solutions, http://www.atlantec-es.com.

BRONSART, R; ZIMMERMANN, M (2006), *Application of Knowledge Based Engineering Methods for Standardization and Quality Assurance in Ship Structural Design*, World Maritime Technology Conference (WMTC), London.

FRIEDMAN-HILL, E. (2003), Jess in Action, Manning, Greenwich.

HAN, J.; KAMBER, M. (2006), *Data Mining – Concepts and Techniques*, 2nd edition, Morgan Kaufmann Publisher, San Francisco.

ISO 214 (2004), Industrial automation systems and integration – product data representation and exchange – part 214: Core data for automotive mechanical design processes. IS 10303-214:2004, ISO, Geneva.

ISO 215 (2004), Industrial automation systems and integration – product data representation and exchange – part 215: Ship arrangement. IS 10303-215:2004, ISO, Geneva.

ISO 216 (2004), Industrial automation systems and integration – product data representation and exchange – part 216: Ship moulded forms. IS 10303-216:2004, ISO, Geneva.

ISO 218 (2004), Industrial automation systems and integration – product data representation and exchange – part 218: Application protocol: Ship structures. IS 10303-218:2004, ISO, Geneva.

ISO 227 (2005), Industrial automation systems and integration – product data representation and exchange – part 227: Application protocol: Plant spatial configuration. IS 10303-227:2005, ISO, Geneva.

JBOSS (2009), Drools *Expert User Guide*, in: Drools 5.0 Manual Set, available from: http://www.jboss.org/drools/downloads.html (last accessed January 28, 2010).

KOCH, T.; et.al. (2010), *Produktivitätssteigerung durch Qualitätssicherung schiffstechnischer Produktdaten - Abschlussbericht*, Hamburg.