Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2009
August 30 - September 2, 2009, San Diego, California, USA

DETC2009-86063

KNOWLEDGE DRIVEN DESIGN FEATURES FOR THE PRODUCT LIFE CYCLE OF ENGINE PARTS

Jan Tim Jagenberg
Erik A. Gilsdorf
Reiner Anderl
Department of Computer Integrated Design
Technische Universität Darmstadt
64287 Darmstadt, Germany
E-Mail: {jagenberg, gilsdorf, anderl}
@dik.tu-darmstadt.de

Thomas Bornkessel
Rolls-Royce Deutschland Ltd & Co KG
Eschenweg 11
Dahlewitz
15827 Blankenfelde-Mahlow, Germany
E-Mail: thomas.bornkessel@rolls-royce.com

ABSTRACT

The high competitive pressure in the aero-engine market demands higher quality products in shorter time at lower costs. In order to achieve this, a close integration of the product lifecycle with early design stages is necessary. Decisions made in design have an impact on later lifecycle areas like manufacturing and aftermarket, which a design may not foresee without the relevant information. This leads to avoidable iterations in the product development process. This paper illustrates a concept for a design decision support system on feature level. Key knowledge of different design domains is provided within the available design systems during the product development phases.

INTRODUCTION

The high competitiveness in the aero-engine market demands better products at lower costs. In today's engineering solutions, standardised parts provide significant cost savings [1]. They enable the reuse of not only the part itself but also the associated experience and key knowledge. Standards are often derived from best practices of other domains which may differ from the approach design would advise. A standardised feature, as well as the process needed for verification[1], only needs to be validated once. Depending on the parametric range of a standardised feature, the same machines and tools can be reused during manufacture and design. It is also possible to supply simulation or surrogate models with the part, further supporting the design and validation processes. Although almost all engineering companies have internalised the idea of standard parts for reuse, it is often limited to documents and drawings. A first step has been the deployment of enterprise knowledge portals for information management and retrieval. Knowledge Based Engineering (KBE) refined that idea by integration into CAx applications.

The Standard Feature Project is an initiative to develop a stronger linkage across the product life cycle by standardisation of common parts and processes at Rolls Royce plc. As the integration of different functionalities into one part is a central strategy for lightweight design, the reusability of the individual parts is reduced. In order to support a reuse approach even for parts with a high level of function integration, a concept has been developed that adapts all advantages of standardised parts to the lower levels of the product model, the level of individual features. In this context, a feature represents a functional element of a part. Similar to a parts library [2], a hierarchical library of reusable features is established exemplary for a jet engine's High Pressure Turbine Disc (HPT). By linking expert knowledge to these features, conformance with best practise in manufacturing can be promoted. The content of this library is accessible di-

[1]Verification ensures that a part complies with specifications. Validation ensures that the part accomplishes its intended requirements.

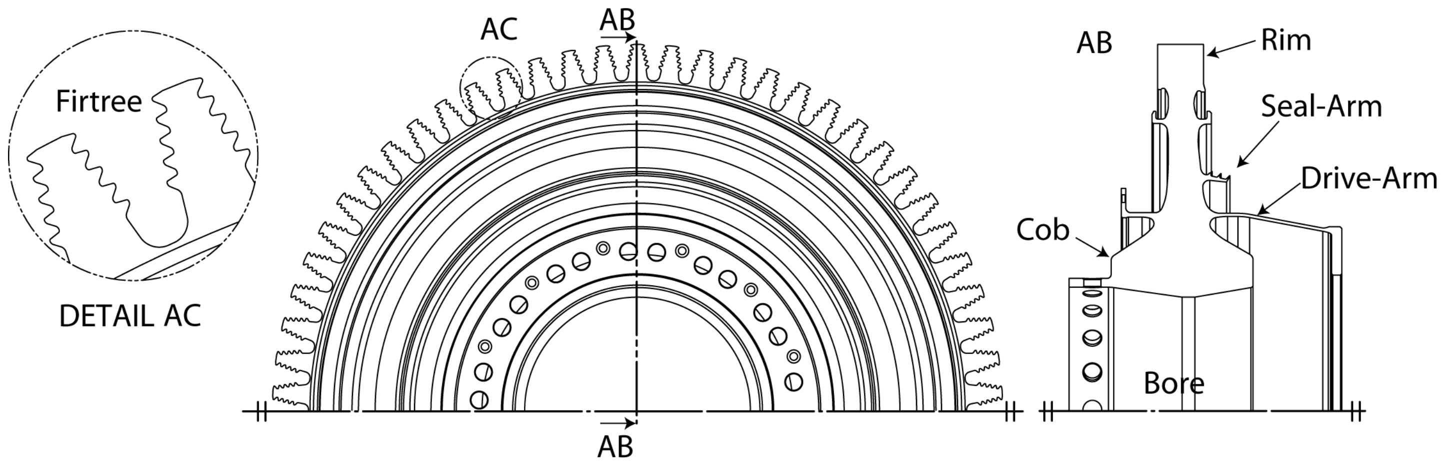

Figure 1. Drawing of an HPT

rectly from within the CAD application via a context sensitive user interface.

On the following pages we will first introduce the reader to the use of hierarchies as a tool to structure the reuse feature library. We then explain why separation of knowledge and geometry is considered essential to the concept. After these general topics the implementation of the library is explained in more detail also highlighting the process established to collect the necessary information. The last two sections deal with the underlying modelling technique which enables the modular reuse of individual features and the user interaction implemented in the demonstrator application.

MODULARISATION

As mentioned in the introduction, the concept of standardisation is not restricted to the part level. In order to extend it to feature level, a modularisation of parts is necessary. This has been done by analysing a high pressure turbine disc from a functional point of view.

For a basic understanding of the functionality provided by the high pressure turbine disc the essential aspects will be briefly explained: The disc is located right behind the burning chamber of a jet turbine and transmits the aerodynamic forces created by the turbine blades to the shaft, which then drives the high pressure compressor disc. In order to achieve this, the disc has to provide three main functions; positioning of the blades in the gas path, torque transmission to the shaft and sealing against the gasses coming from the gas path. The blades have 'firtree' like shapes at their base which are inserted into corresponding slots in the outer rim of the disc (see fig. 1). In combination with the radial forces when spinning this provides fixture for the blades. The torque transmission is achieved via the 'drive-arm' a tube-like structure with a flange which is then bolted to the next stage of a shaft. In order to avoid gas leaks from the gas path, a labyrinth seal, the 'seal-fins' are placed on another tubular structure, the 'seal-arm'. These very thin and sharp fins are positioned in such a way, that they run into a softer element on the countering part and thus block gas streams. The disc-shaped 'disc main body' consists of a 'rim' and a rather thin 'diaphragm' connecting the rim to the massive central 'cob'. The diaphragm transfers the forces created at the blades to the drive arm, while the cob absorbs the centripetal forces created by the other rotating elements.

To come to a generic solution for the modularisation, archives have been searched for drawings of all past jet engines. The process of capturing the design and manufacturing knowledge for a given part contains four major steps; the definition of the parts and features hierarchies, the parametrisation of the identified features, and the externalisation[2] of the expert knowledge. The definition of the final part and feature hierarchies has been achieved during multiple interviews with design engineers. Most function-oriented modularisations represent a trade-off between different perspectives on a feature and the resulting hierarchy depends on the people involved. In order to generate a flexible and accepted hierarchy, a compromise between involved parties has to be reached.
With the feature hierarchy established, manufacturing engineers were interviewed in order to identify and derive knowledge about common issues on these features.

HIERARCHIES

As hierarchies are used on multiple occasions throughout the data model, they are explained further in this section.
Because hierarchies impose more strict rules on the organisational structure of a number of entities, they limit the resulting

[2]abstraction of a persons knowledge into a machine interpretable form

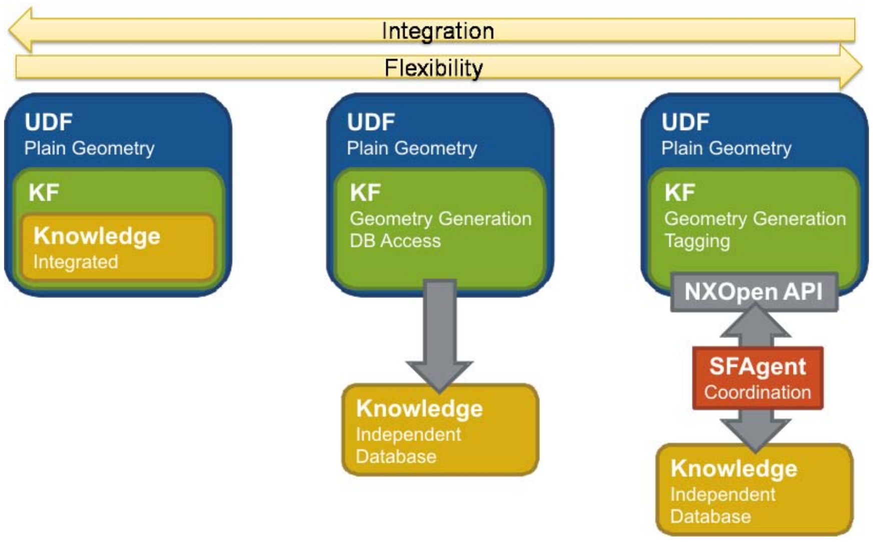

Figure 2. Separation of Geometry and Knowledge

complexity and thus enable easier understanding and maintainability. Hierarchies appear in many places where a high number of entities interact with each other [2]. Regarding humans this might be in administrations, authorities, companies or in the military. In the context of computers the most used hierarchy is the file system, where each file or folder belong to exactly one superordinate folder.

Two distinct types of hierarchies can be identified based on the number of super elements that can be assigned to a sub element [3]. The more relaxed variant is the poly-hierarchy, which allows multiple super elements per sub element. This can result in a very complex directed acyclic graph. By reducing the number of possible super elements to one, the mono-hierarchy limits the possible complexity to an easier to grasp tree. For two subsequent elements in the tree there is only one route connecting the elements.
Though there are cases which cannot be modelled by mono-hierarchies. (e.g. a bicycle is a vehicle and a sportive device) mono-hierarchies still show some very desirable advantages; By limiting the possible number of super elements, they are easier to grasp. As mono-hierarchies are basically a tree structure they are very easy to visualise and navigate. The biggest advantage of allowing only one super element is the avoidance of ‘collisions’. A collision occurs when a sub element inherits properties from two super elements, and those two define opposing values.

Commonly hierarchies are combined with one of the central concepts of object orientation, inheritance [4]. Similar to the inheritance that is known in biology, inheritance in object orientation enables the propagation of properties across hierarchy levels. Because sub elements inherit properties from their super elements, a property that is common to a whole family of elements only has to be defined once on the appropriate level. By adapting this behaviour, redundancy of information can be dramatically reduced which increases the maintainability of a hierarchy structure.

SEPARATION OF KNOWLEDGE AND GEOMETRY

Integrating the knowledge into the geometry offers an easy way to start in the field of KBE. Siemens PLM NX, the CAD environment used to implement the prototype, offers two ways to interact with the geometry in a programmatic fashion:
Knowledge Fusion (KF), an object oriented programming language based on Intent! is merged into the system and offers close interaction via a selection of NX-specific libraries.
NXOpen, a language agnostic application programming interface (API) which enables the access to a wide range of NX functionality from programming languages like Visual Basic, C# or Java.
With the former solution, knowledge or the access to external knowledge can be integrated into the CAD-model while the latter provides an interface for external applications (see fig. 2[3]).

While the KF-based approach offers advantages regarding provided functionality it also has some severe disadvantages which can be avoided by separating geometry and knowledge representation by an intermediate layer which is implemented using the NXOpen API:

1. Separation evades vendor lock-in, which happens when knowledge is represented in a CAD system specific form inside the CAD model.
2. Separation enables reuse of knowledge by other software systems outside of the CAD environment which would not

[3]UDF = User Defined Feature

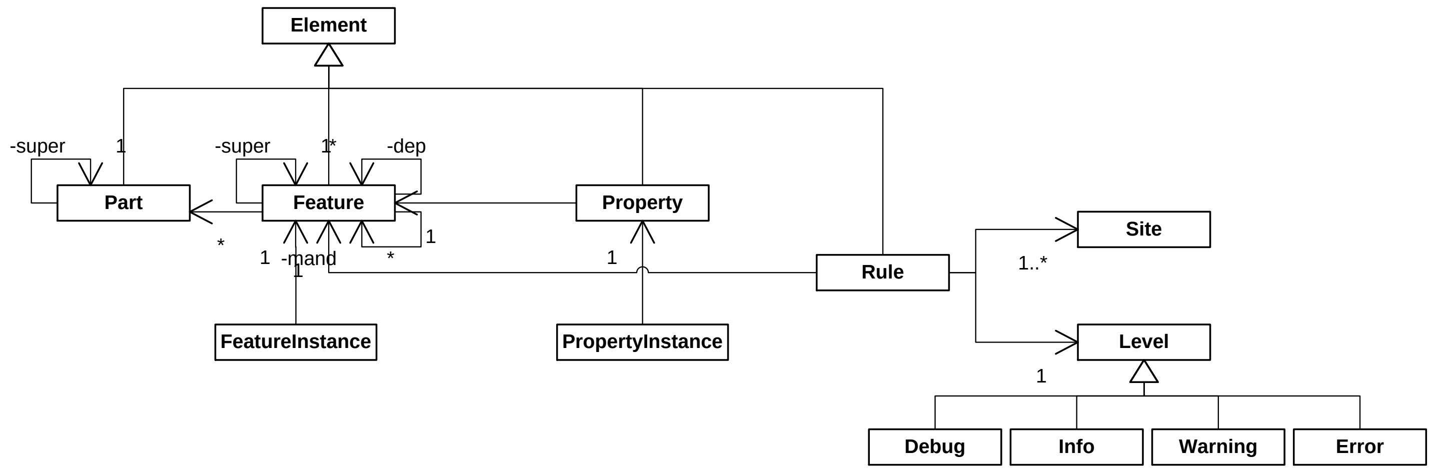

Figure 3. UML Diagram

be able to access information stored in a system specific way.

3. Separation enables reuse of knowledge without creating redundancy. For example if a certain information is valid for multiple geometric elements, it is only stored once and associated multiple times.
4. Separation massively reduces redundancy for feature family structures with inheritance. Knowledge is associated to the highest meaningful level in the inheritance hierarchy with the association being inherited by the children.

When supporting separation of knowledge and geometry, there are two options how to access the knowledge storage. The access routines can be integrated into each feature or an intermediated layer evaluates the knowledge for an associated feature.
The latter has the advantage of reducing the redundancy of code and thus supporting better maintainability. If changes need to be made to how the knowledge storage is accessed or how the knowledge is evaluated, the scope is limited to the coordinating layer. Thus the knowledge derived in the Standard Features Project is stored in an external database which is accessed and evaluated by an independent component, the Standard Features Agent[4], shown in fig. 2. The presented organisation of the Standard Features Library follows principles of object orientation and is partially aligned to the parts library concept defined in ISO13584 [2]. The library arranges the Standard Features in a mono-hierarchy and links them with semantic associations and thus provides clear traceability and structured accessibility.

[4]The definition of 'agency' is discussed in [5].

IMPLEMENTING THE DEMONSTRATOR LIBRARY

All elements described in the following section and the UML[5] diagram shown in fig. 3 share the same metadata[6] which is defined in the class Element. Information such as the preferred name or a definition support the selection and correct application of Standard Features. Metadata like revision numbers or contact information of the responsible designer provide valuable organisational details.

A ***Feature*** is the central element in the data structure. On the information side, it allows the linking of additional information elements from various domains. On the CAD side, it also provides a geometric representation which can be manipulated parametrically. The methodology of modelling the geometry as a UDF is described in chapter .
A Feature is associated to another Feature, the so called super Feature. In object oriented terms this is called a generalisation, as the super Feature is a more general version of the current Feature. In order to limit the propagation of changes through the Standard Features Library, this association is directed. A Feature 'knows' the more general Feature it is derived from, but the super Feature does not know about its sub Features. Deriving a new Feature from an existing one does thus not change the 'parent'. Another important property of a Feature in the context of this generalisation hierarchy is abstractness. An abstract Feature does not provide an 'implementation', a geometric representation, it merely serves as an structural element to enable further structuring of the feature library. For example the various implementations of a firtree are all derived from an abstract firtree Feature. Aside organising the library better, this also reduces redundancy, as Properties which are valid for all types of firtrees are defined on the common but abstract level.

[5]Unified Modelling Language, used for object oriented design and communicating ideas between team members
[6]Data about data

A Feature also can be associated to additional Features, with the associations being two distinguished types of dependencies. The first is the so called ‘existence dependency’. The Feature defining this dependency can only be inserted into the CAD-model if the associated Features are already present. The second association is the ‘mandatory dependency’. It allows a Feature to define a number of other Features which need to be inserted after itself. Aside the mentioned dependencies, another property is used to decide about the applicability of a Feature; its multiplicity. The multiplicity decides whether a Feature can occur multiple times in the same Part.
Another important concept adapted from object orientation is the differentiation between an object and its class. While a Feature resembles a class, just stating which Properties are used to define it, a FeatureInstance resembles the actual object with distinct values given for each PropertyInstance. While a Rule is associated to a Feature, the values from the corresponding FeatureInstance are used to evaluate it.

The elements reflecting the parameters of the CAD-model on the information side are the ***Properties***. Though they are strongly linked to the parameters of the geometric model (‘expression’ in NX[7] terms), they are modelled in a way that allows later extension to other sources as well. In order to limit the complexity, a Property can only be used on one Feature and is associated to this Feature via a bidirectional association. A Property ‘knows’ which Feature it belongs to and a Feature is ‘aware’ about its associated Properties.
It is important to note, that the Properties on the information side do not hold any values. They merely define what type of Properties exist for a Feature. The concrete value of a Property for a given model is acquired from a PropertyInstance which connects to the CAD system.

Generally speaking a ***Part*** resembles a possible composition of Features. Aside this product structure oriented definition, a Part also provides a filter for Features. As not every type of Feature is meaningful in every type of Part, only relevant Features for the current type of Part are made available in the Standard Features Library browser. In order for this mechanism to work, each Feature is associated to a number of Parts which this type can be used on. Similar to the super Feature association this link is directed, thus enabling a new Feature on a certain Part does not change the Part itself.

Rules are the generic element used to create an abstract representation of knowledge. A Rule is associated to a Feature and has full access to all information on Feature level. In the current implementation, Rules are limited to boolean return values but the architecture is already aligned to allow future Rule implementations with other return types. When testing a Rule, the current values for all PropertyInstances are passed to the Rule, which then evaluates a mathematical expression using those values. Depending on the outcome, the Rule returns either a valid status or fails.

Levels If a Rule fails, the further steps depend on the severity Level assigned to the Rule. The Levels are:
Debug: This is the lowest Level of severity and is only used during the development process.
Info: A violation of this Level does not pose a problem, it merely informs the user about further implications of his decision.
Warning: A warning level violation indicates that the user should take action to avoid the described problem. But he can still proceed with his design workflow.
Error: When a violated Rules is of error Level, this indicated that something major is wrong and the user cannot progress in the design process until the issue is fixed.
If an error Level Rule blocks further progress, the user can override this Rule while providing a reason why he chose to do so. After this step the Rule is marked overridden, which is visualised by a different icon and a change in colour, and the further process steps are unlocked.

Sites offer another filter for the evaluation of Rules. For each Rule the information for which Sites it is valid is stored. During the design process, the user can select which Sites should be taken into account. This information is then used to only show the appropriate warnings to the user and hide any unnecessary information which could distract him.

After this introduction of the elements used to represent the library, the next paragraphs will describe the process used to actually establish the content.
The definition of hierarchies starts with an overview of existing discs and the features used on these discs. The differentiation of the features in this first step will in most cases not resemble the variety of features after hierarchically organising them, but nonetheless it still enables a first understanding of the variability. By structuring the discs and used features in a matrix, the accessibility of the gathered information is further enhanced.
After analysing the variant space, the parts hierarchy is established. It tends to be already reflected in the naming scheme of many engineering parts. The name ‘high-pressure turbine’ hints the three hierarchy levels part, type of part and pressure level.

The features hierarchy is driven by three desires: maintainability of the features library, flexibility when reusing functional features and separation of topological changes. In order to support the creation and maintenance of the Standard Features Library, the number of features, the number of inheritance levels and the number of interdependencies should be kept as low as possible. This tendency is counterweighted by the initial desire to have a comprehensive modular set of reusable features which represent certain functionalities and also by the necessity to separate topological variation into individual features.
The latter is a good instrument to choose the level of granularity when decomposing a part into individual features. This is best explained if we look at at simple disc main body, with four

[7] CAD-System by Siemens PLM based on the Parasolid geometric kernel

types of topologically different firtrees (truncated/non-truncated and barreled or flat) and three types of topologically different bores (straight, stepped and angled). Suppose we model this part in a traditional 'all-in-one' model, we end up with 12 different geometries we have to create. If we split it up into a static disc main body, 4 firtree modules and 3 bore modules, the number of necessary models is reduced to 8.

In a 'all-in-one' part the number of models follows a product rule ($n \cdot m$) while a modularised part is described by a sum rule ($n + m$). So even if presently a part of the geometry exists in only one variant, it can be useful to separate it into an individual feature, if it is realistically conceivable that one or more variants will be added in the future.

Good examples of the decomposition into features are the seal-arm and the seal-fins. Though the seal-Arm presently only exists in one variant, these features are separated for two reasons: first it is conceivable that another variant of the seal-arm is developed, second the separation aides the reuse of the function the seal-fins could provide in combination with other features. Another example highlighting the opposing force, the strive towards a simple library is the disc main body. It combines the rim, diaphragm and cob into one feature, as all identified variants are based on parametric variation.

The resulting features hierarchy is shown in fig. 4.

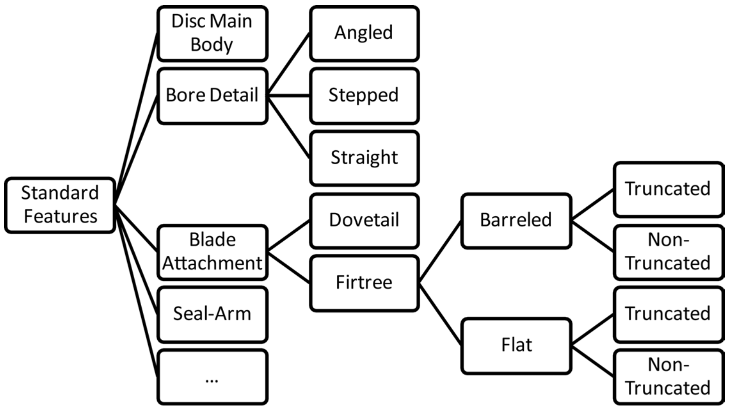

Figure 4. Features Hierarchy

The selection of parameters used to define each feature is based on existing parametrisations derived from the traditional, sketch based design process. A Rolls-Royce plc. internal Quality Function Deployment (QFD) process was applied to establish the appropriate parametrisation according to the needs of the involved parties.

Based on the existing information about the parametrisation and the features hierarchy developed in the previous steps, a consistent naming convention is established. The naming convention used in the Standard Features Project is driven by three main factors:

1. The available character set in the CAD application.
2. The fact that the CAD application (Siemens PLM NX) does not allow the same expression names on different features of the same part.
3. The behaviour of NX regarding the extension of identifiers with a history number upon insertion of UDFs.

With the geometry centric content of the library established, the next step is to create abstract representations of the experts knowledge and link this to the appropriate features.

Based on the modular features previously established, the knowledge of the manufacturing experts is externalised. Using simplified sketches of the individual features which highlight the parametrisation, the lists of available parameters and the naming convention, interviews are conducted to derive computer interpretable representations of the experts' experience and advice.

Fixed constraints are captured as equations which are stored as rules in the knowledge database. The representation as rules enables the attachment of additional information to the bare equation. Commonly a human language description of the derived rule is stored in the rule definition. Additional information like involved machines or processes are stored in the note and remark field.

Weak statements which cannot be expressed as a mathematical expression are attached as rules of information level and allow to inform the end user about preferred values or variants. For example the straight bore, is associated with a rule of information level which always is visible, stating that the use of this version of the bore feature is discouraged by manufacturing, as is poses problems during the measuring and verification process.

GEOMETRIC MODELLING

User defined features (UDFs) are the easiest way to link up design elements and product knowledge, for example manufacturing knowledge. A User Defined Feature is a container for geometric information like feature operations, geometry elements and values. UDFs only require little additional familiarisation compared to other approaches like API[8] implementations or scripting languages and can be used and edited easily by every designer. In contrast to other approaches [6], UDFs used in Standard Feature Agent do not contain information about the knowledge database (reasons are provided in section). Only geometry and information about modifiable parameters and geometric references are stored within.

For a robust and sustainable library, best practises for UDF cre-

[8]Application Programming Interface, a set of methods and classes enabling close application integration

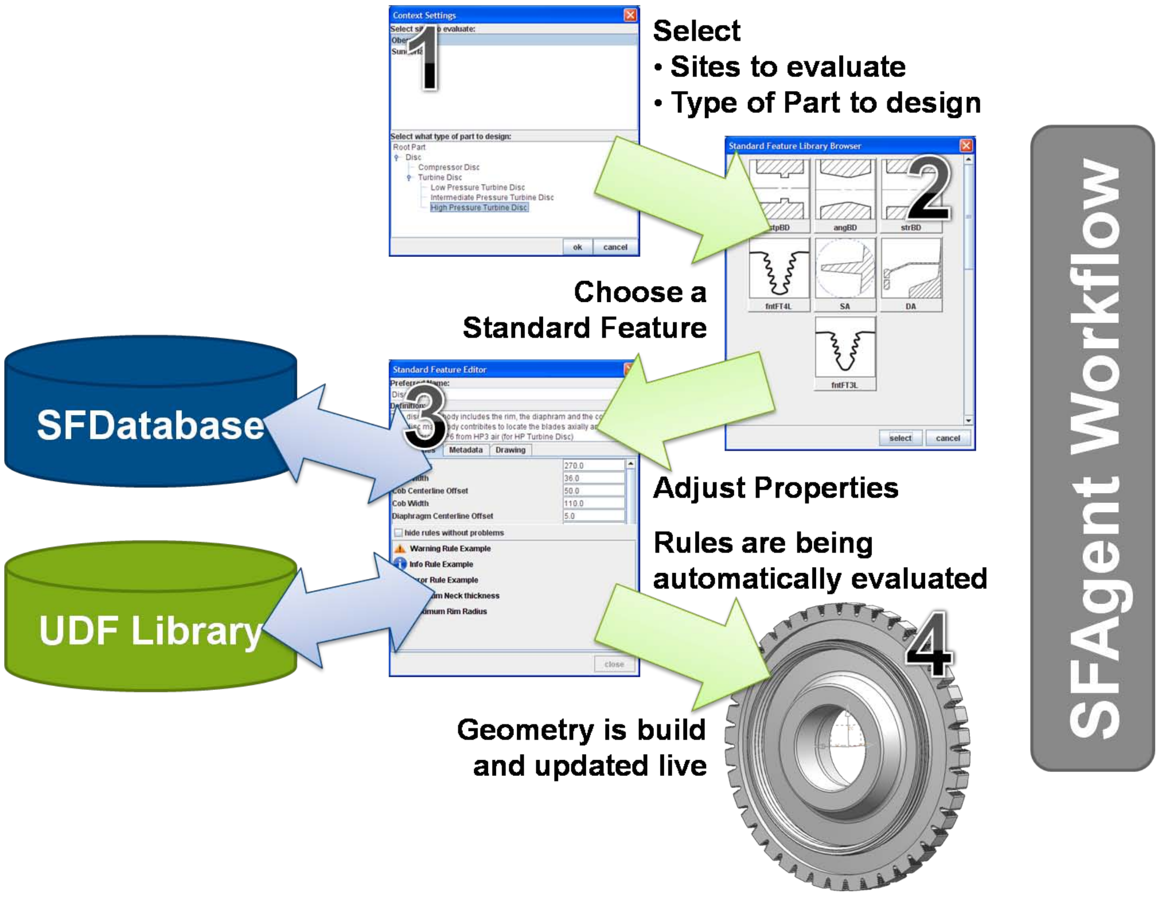

Figure 5. SFAgent Workflow

ation need to be defined. Especially geometric interfaces for interaction with other features have to be well defined. For that reason, only topological elements like edges, faces and bodies are valid interfaces as they represent the solid part. This ensures replaceability and extendability of library contents. Auxiliary geometry like sketches and curves are unreliable elements, as they limit modelling to certain techniques. For example a cylinder might be created as revolved or extruded sketch. Though outcome geometry is the same, associated sketches are different and therefore not suitable as references.

USER INTERACTION

The concept has been implemented as a demonstrator application in NX 6 with the help of NX Open API[9] and Knowledge Fusion[10]. A Java framework joins all elements of the Standard Feature Agent. An object-oriented database has been chosen as knowledge storage. The demonstrator allows the use of traditional NX modelling techniques and adds support for quick model (re)creation and editing. The demonstrator has been integrated into NX 6 as an application and can be started from the menu bar. The context settings dialogue enables the user to select one part from the parts hierarchy and multiple sites from a list of available sites. The part selection influences which features are made available depending on their applicability and the sites selection decides which rules will be evaluated.

Beginning with a new part, users can add new Standard Features from a custom browser. The library browser is context-sensitive. Depending on the present Standard Features in the actual CAD model, the Standard Feature Library is evaluated and only features with fulfilled dependencies and valid multiplicity are shown.
Selection of a feature brings up a new window which displays the editable properties list and a rule evaluation at the bottom (see fig. 5).

To adjust parameters to fit actual needs, the designer can

[9]API provided by NX, allowing the implementation of custom applications
[10]Knowledge based engineering language integrated with NX

change them by entering new values. At the same time, the rules below are evaluated and the background colour of the field changes to highlight that a default value has been changed. Hovering the mouse over one of the parameters shows the definition and a double-click provides the user with a simplified sketch describing the parameter.
The tab ‘Metadata’ offers additional organisational information, like synonymous names, versioning information and contact data for responsible designers. ‘Drawing’ gives a general overview over the geometry of the Standard Feature.

Rule evaluation distinguishes different levels. A description of all rules can be found in section . If a rule fails, the designer is still able to overwrite this rule by providing a reason. The rule is marked as overwritten and the ‘insert’-button is made available again. As some features are essential for a basic turbine disc design, they are defined to be mandatory features . For example the blade attachment, in this case a so called firtree, and a bore for the main disc are necessary for torque transmission. When a disc body is inserted via the agent the dependency structure stored in the database is evaluated. The disc main body has a mandatory association to the firtree feature, which itself is abstract and thus does not provide a geometry implementation. The Agent then searches the feature hierarchy for sub-features of the firtree which are not abstract and a pop-up window offers the user a choice of available firtree sub-features.
After the user created a complex part, he is still able to edit Standard Features which he previously inserted. A initial dialogue offers the user a choice of all identified Standard Features in the current part from which one is selected for editing. The same dialogue used during insertion of Standard Features is used to manipulate existing ones as well. When changing preferences in the Standard Feature Editor, the CAD-model and the applying rules are automatically updated and re-evaluated in the background.

In order to support flexibility, the whole concept is designed to be open. The Standard Feature Agent provides a persistent dialogue which can be used to present and evaluate rules even when the Standard Feature Editor is closed. The previously inserted Standard Features can be manipulated like any other UDF while still maintaining the checking against the rule base. With this open approach, the user is free to use the Standard Features Library as a basis for a new design, which he then extends using the native tools provided by the CAD-system.

CONCLUSIONS AND OUTLOOK

A well structured library of reusable features enables the association of expert knowledge and thus streamlines the design process. The two major factors driving this advancement are the modularisation below part level and the frontloading of knowledge from the various product life cycle domains. Especially during the early design stages, decisions with a high impact on the overall outcome are made. By supporting the designer with additional information about downstream implications of his decisions, the product quality can be improved and the cost reduced [7].

Future research will focus on the integration of knowledge from other product life cycle domains, such as lifing or cost estimation. Other areas of interest are the extension of the concept assembly design and process automation. The fully parameterised nature of the concept aligns especially well with automated optimisation, which could also consider knowledge based constraints. The demonstrator described in this paper is implemented as a monolithic application inside NX 6 with all components (database, UDF-library, logic) running in the same context. Thus, in order to achieve maintainability and accessibility, we suggest the integration of the Standard Features Library and the Knowledge Repository into a corporate PLM/PDM system.

REFERENCES

[1] Kim, K., and Chhajed, D., 2000. “Commonality in product design: Cost saving, valuation change and cannibalization”. *European Journal of Operational Research,* **125**, pp. 602–621.

[2] International Organization for Standardization, 1997. ISO/FDIS 13584-42 - Parts library: Description methodology: Methodology for structuring parts families, September.

[3] Gaus, W., 2005. *Dokumentations- und Ordnungslehre*. Springer.

[4] Poo, D., Kiong, D., and Ashok, S., 2007. *Object-Oriented Programming and Java*. Springer London.

[5] Franklin, S., and Graesser, A., 1997. *Intelligent Agents III Agent Theories, Architectures, and Languages*, Vol. 1193/1997. Springer, ch. Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents, p. 26.

[6] Danjou, S., Lupa, N., and Köhler, P., 2008. “Approach for automated product modeling using knowledge-based design features”. *Computer-Aided Design and Applications,* **5**, pp. 622–629.

[7] Asiedu, Y., and Gu, P., 1998. “Product life cycle cost analysis: state of the art review”. *International Journal of Production Economics,* **36**(4), pp. 883–908.