INTERNATIONAL DESIGN CONFERENCE - DESIGN 2002
Dubrovnik, May 14 - 17, 2002.

A DIFFERENT VIEW ON PDM AND ITS FUTURE POTENTIALS

Christian Weber, Horst Werner and Till Deubel

Keywords: Design Theory, EDM/PDM, CAx-Integration

1. Introduction

The objective of this paper is to re-conceive the nature and development potentials of PDM-systems based on a new approach to design theory called "Property-Driven Development/Design" (PDD) which has been presented in [WeWe-00, WeWe-01].
Property-Driven Development/Design (PDD) is a new modelling approach for products and product development processes which is intended to give a theoretical description of design and designing which is closer to practice as conventional design theories and methodologies such as [VDI-2221]. The distinctive feature of the PDD approach is that it considers functional and other properties on equal terms and offers a seamless description of all phases of the development/design process. Due to its formal consistency and flexibility it can serve as a theoretical basis for the concept of future PDM systems[1].
Today's PDM-systems suffer from a lack of formalised representation of products: They handle *data*, not *knowledge*. The information is usually buried within various documents and being processed by different software-tools which have no common "language". This paper proposes a way to improve the design process by means of an advanced kind of PDM system based on a comprehensive product model.

2. Property-Driven Development/Design (PDD)

The concept of PDD is mainly based on the distinction between characteristics (in German: "*Merkmale*") and properties ("*Eigenschaften*") of a product: The characteristics are the product-defining parameters and their structure (its "*Beschaffenheit*"), the properties describe the product's behaviour ("*Verhalten*"). While the characteristics can be directly determined by the designer, the properties are a result of the chosen characteristics and can not be directly influenced.
The characteristics are very similar to what Hubka and Hubka/Eder call "internal properties" [Hubk-73, Hubk-84, HuEd-92, HuEd-96] and what Suh calls "design parameters" [Suh-90], i.e. parts' structure, geometry, material and surface characteristics of a product. The properties are related to Hubka's and Eder's "external properties" and to Suh's "functional requirements", repectively, e.g. weight, safety and reliability, aesthetic properties, but also things like "manufacturability", "assembability", "testability", "environmental friendliness" and cost of a product.
To be able to handle characteristics and properties – literally thousands of them in complex products and to keep track of them in the development process they have to be structured. **Figure 1** shows on the left a fairly obvious proposition for the (hierarchical) structuring of characteristics which follows the parts' tree of a product. (Other methods of structuring characteristics are theoretically possible, but

[1] In this article no distinction is made between "PDM" and "EDM" or "EDM/PDM" (Product/Engineering Data Management), but the least "bulky" term is preferred.

not discussed here.) On the right of figure 1 the most important ("top-level") classes of properties are given as a first entry into their structuring. Of course, these also should be structured more deeply by further decomposing them (in any case hierarchically?). It is the authors' opinion that such a decomposition of properties is always specific to individual industries (product classes), often even specific to individual companies within the respective industry. Because the issue of this article is more general no attempt is made here to the further structuring of properties.

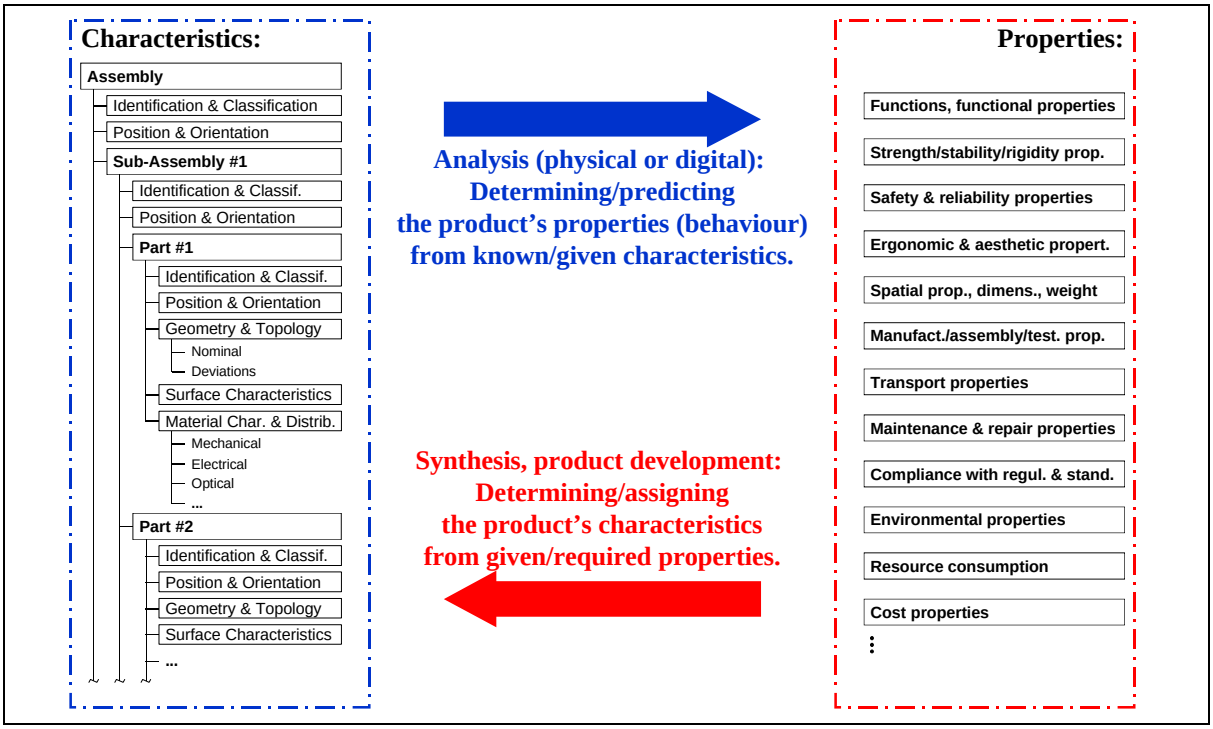

Figure 1. Characteristics and properties with the two main relations between the two

Figure 1 also shows the two main relations between characteristics and properties which correspond with the two main activities in the product development/design process:

- **Analysis:** Based on known/given characteristics of a product its properties are determined or if the product does not yet exist in reality predicted. Analyses can, in principle, be performed via experimentally (e.g. using a prototype) or "virtually" (e.g. using digital simulation tools).
- **Synthesis:** Based on given, i.e. required, properties the product's characteristics are to be determined. Synthesis is the main activity in product development: For the customer mainly (only?) properties are relevant, thus the development/design process begins with a list of required properties. The designer's task is to find appropriate solution patterns and determine/assign their respective characteristics in a such way that the required properties are met to the customer's satisfaction.

In the PDD approach the two main relations between characteristics and properties are modelled in more detail, in principle following a network-like structure. **Figures 2 and 3** show the two basic models for analysis and synthesis, respectively. The expressions used in the figures have the following meaning:

C_i:	Characteristics ("*Merkmale*")	**R_j:**	Relations between characteristics and properties
P_j:	Properties ("*Eigenschaften*")	**EC_j:**	External conditions

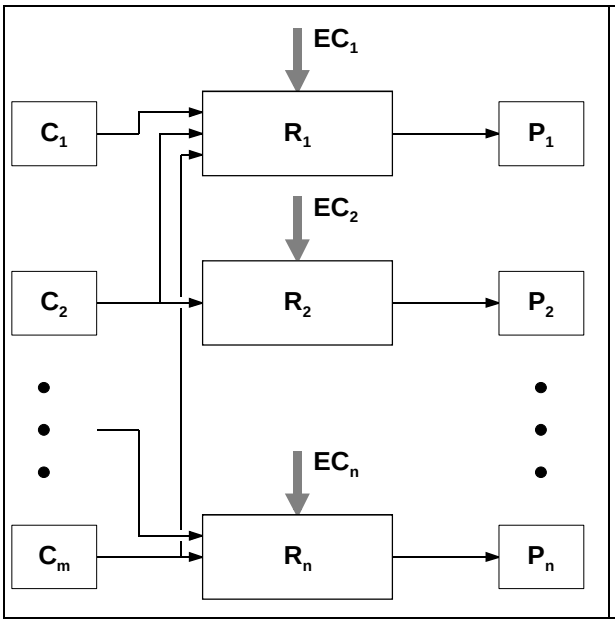

Figure 2. Basic model – analysis

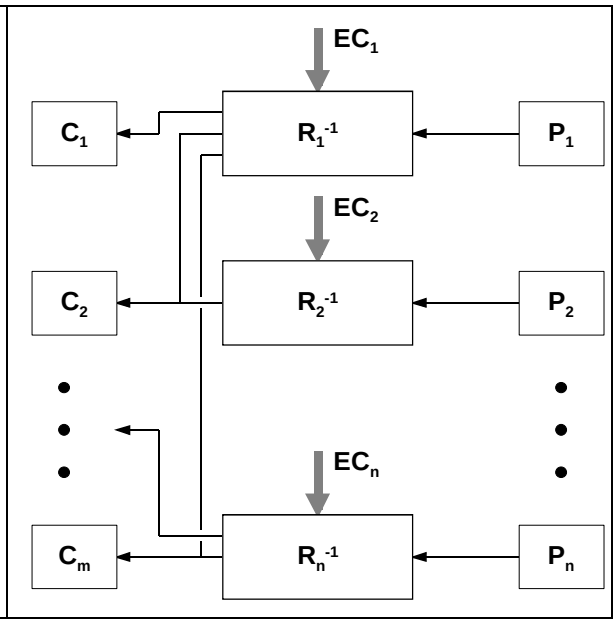

Figure 3. Basic model – synthesis

Once the product is realised (i.e.: the product's characteristics $\mathbf{C_i}$ are physically given), its properties/ behaviour ($\mathbf{P_j}$) can be analysed by measuring and testing (albeit this may be quite time- and money-consuming sometimes, e.g. when testing/checking the product's durability). In this case the product itself is the representation of the relations ($\mathbf{R_j}$). As is well known, measuring/testing properties alone does not reveal ***why*** the product behaves as it behaves. To answer this question, abstract models, methods and tools have to be established which are exactly what the relation-boxes ($\mathbf{R_j}$) in figure 2 stand for.

During the product development process, however, when there is not yet a finished product, its properties can ***only*** be analysed by means of appropriate models, methods and tools which represent the relations ($\mathbf{R_j}$) and tell about the influences the relevant characteristics ($\mathbf{C_i}$) have on the respective properties ($\mathbf{P_j}$), thus ***predicting*** the properties depending on characteristics given at that moment.

Models, methods and tools to realise the relation-boxes could be physical (e.g. [component] prototypes and specified test procedures). But increasingly non-physical models and procedures are applied, in many cases mathematical ones. The table shown in **figure 4** gives a rough list of different (classes of) methods applied for analysing (predicting) a product's properties during the development process.

• Guesswork, estimation • Experience • Customer interrogation • Physical tests/experiments • Tables, diagrams	• Calculation/simulation – Traditional/simplified – Numerical/computer-supported – Model-based – Rule-based, "fuzzy" – Semantic/neural networks • ...

Figure 4. **Methods and tools to support (engineering) analysis**

The basic model according to figure 2 needs two additions which are particularly important in the context of "real-life" development/design processes and their computer support (**figure 5**):

- A product may have more properties than the ones originally considered or even required. In PDD they are called "additional properties" ($\mathbf{R_{j+}}$). They may contribute positively or negatively to the considered/required properties or may be not relevant for them. In case that they are considered disturbances, their suppression/diminishing becomes a new required property.

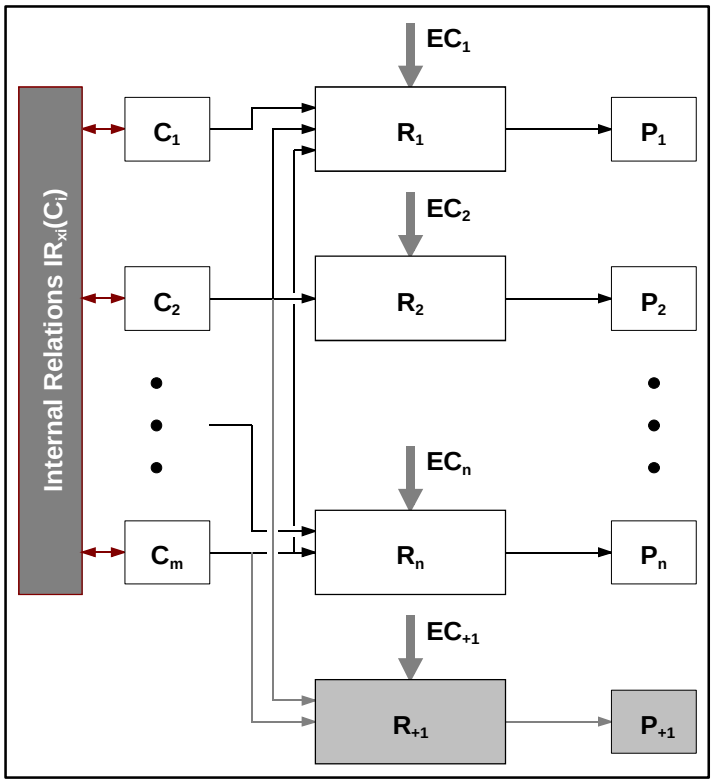

Figure 5. Additional properties and internal relations (constraints) between characteristics

- Very often there are fixed relations between different characteristics of a product, e.g. spatial ("element B parallel to A"), geometrical ("same diameter as ..."), or material. In PDD they are called "internal relations between characteristics" ($\mathbf{IR_i}$). In a mathematical sense these are constraints which decrease the number of free variables on the side of the characteristics. As is well known, some spatial relations as well as geometrical relations can be captured and administered by today's parametric CAD systems.

Synthesis (figure 3) is formally "just" the inversion of analysis (figure 2): Based on given (required), properties ($\mathbf{P_j}$) the product's characteristics ($\mathbf{C_i}$) are to be determined. While in nature the "products" (creatures) themselves even during their lifetimes somehow seem to have the ability to modify their characteristics (e.g. structure, geometry, material) according to changed requirements (required properties), in the technical world we are still very far away from concepts like this. Even new approaches such as "adaptronics" stand for much simpler concepts.

Therefore, the only way to do synthesis in engineering is to use "inverted relation-boxes" ($\mathbf{R_j^{-1}}$) according to figure 3 which stand for appropriate synthesis methods and tools. These are sometimes, but by no means always based on models in the scientific sense. The table shown in **figure 6** gives a rough list of different (classes of) methods that can support synthesis, i.e. which can help to determine a product's characteristics from (required) properties during the development process.

• Human genius[2] • Association – technical patterns – patterns in nature ("bionics") • Experience • Standard/catalogue solutions • Collection of rules	• Methodical approaches • Inverted calculations • Computer tools, e.g. – Model-based, e.g. structural optimisation, genetic algorithms – Semantic/neural networks • ...

Figure 6. **Methods and tools to support (engineering) synthesis**

In this article it is not possible to comment on all the implications of the new approach to synthesis. Considerations on this are to be covered in a book which is in preparation. Only three remarks shall be made, as they have a close relation to the issues discussed here:

- To see synthesis merely as an inversion of analysis is a big over-simplification which has to be discussed in more detail. But here only one aspect shall be mentioned: Even the still very simple basic model according to figure 3 shows that in synthesis conflicts are inevitable,

[2] The same as (quick) association?

because different (required) properties have influences on identical characteristics, and as is well known often adverse ones (**figure 7**).

- The next remark adresses the external conditions ($\mathbf{EC_j}$): They have great influence both on analysis and on synthesis. This was shown to some extend in [WeWe-01] in the context of DfX (design for X) and shall not be repeated here.
- When again: either in analysis or in synthesis computers are used to model the relations $\mathbf{R_j}$ or $\mathbf{R_j^{-1}}$, respectively, the modelling conditions ($\mathbf{MC_j}$) must be considered (**figure 8**). Otherwise it is not possible to interpret and evaluate the modelling/simulation results correctly.

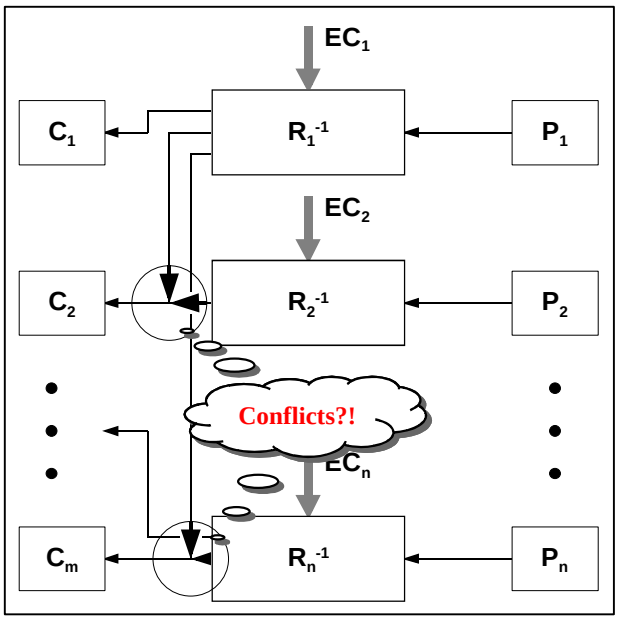

Figure 7. Conflicts in synthesis (schematic)

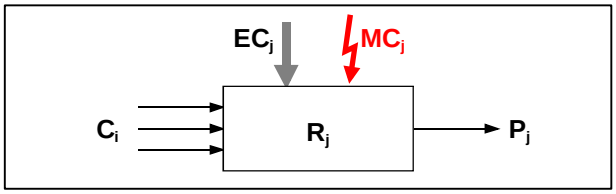

Figure 8. Modelling conditions, important when modelling with computers

Based on the considerations on the new approach to modelling products, now the consequences for the modelling of product development processes are introduced. The product development process can be seen as an activity which, in principle ("strategically"), follows the synthesis model according to figure 3, but has in between ("tactically") many analysis steps according to figure 2. During the process in every synthesis step ever more characteristics of the product are assigned and determined, and in parallel by means of the analysis steps ever more and ever more precise knowledge of the product's properties/behaviour is generated.

The product development process terminates when

- all characteristics (needed for manufacturing and assembly of the product) are assigned,
- all (relevant) properties can be determined/predicted
- with sufficient safety and accuracy, and
- all determined/predicted properties meet (i.e.: are close enough to) the required properties.

Figure 9 gives a schematic overview on this interpretation of the product development process. To avoid too much complexity, in this figure only one synthesis-analysis-evaluation cycle is shown (which, of course, is closely related to the so-called TOTE-scheme described in [Ehrl-95]). Therefore, the growing number of characteristics and known properties from one cycle to the next can not be demonstrated directly, but should be borne in mind (there is one figure with this focus in [WeWe-01]).

One last aspect of the PDD-process concept introduced here should be mentioned: The term "early phases" has in PDD a quite different meaning than in known design theories and methodologies. Here it is not defined with regard to contents of working steps (e.g. in [VDI-2221]: considering functional aspects and solution principles = working in an early phase), but by the number of characteristics and properties which are already known. Then it is easily understandable (and theoretically explainable) that the same properties (function, strength, safety, ergonomics, manufacturing, cost, ...) have to be considered several times in the process. The difference is that in "early phases", where only a couple of characteristics are given, very simple methods and tools are required (giving a rough calculation/ estimation based on a couple of parameters = characteristics), whereas in "late phases" the focus lies on a calculation/simulation as precise as possible (which is based on and also requires a lot of parameters!). Accordingly, in the product development process, different methods and tools for the analysis of the same properties have to be provided.

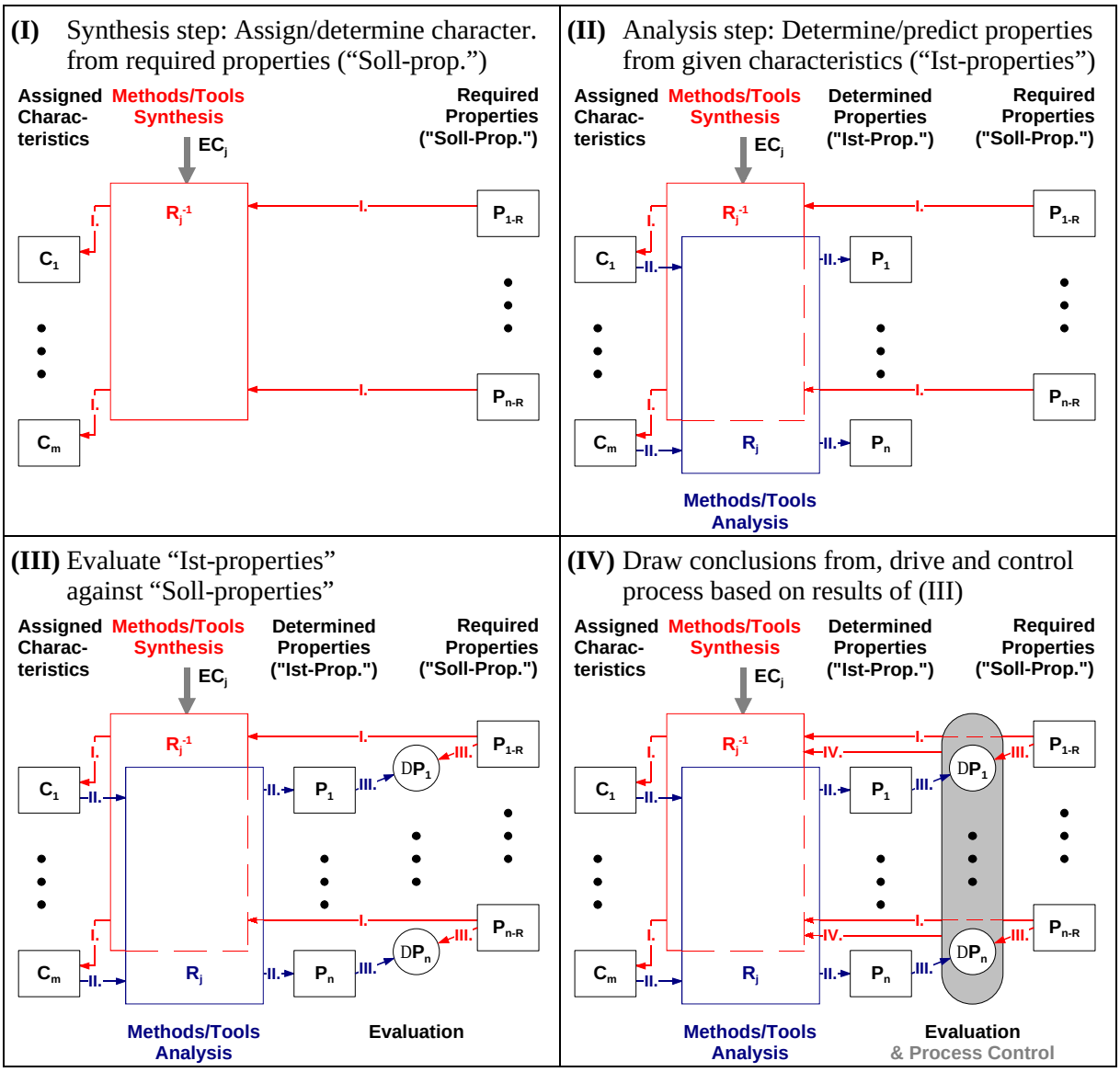

Figure 9. Product development process (schematic)

3. State of the Art in PDM and CAx

The environment of today's PDM systems is characterised by the co-existence of various independent tools, each based on its own specific product model. In this environment the PDM system mainly focuses on the administration of computer files generated by these tools without having much access to the actual contents of the files. The workflow management is limited to "static" processing of predefined process patterns.

The co-existence of the CAx tools' specific product models brings along a high redundancy in the (digital) product documentation. Therefore, a "master system" is needed which contains the currently valid model(s) and acts as a source for all tools and systems involved.

By tradition, this is the role of the CAD system, since geometry makes up for the major part of the product-defining characteristics and is needed as a basis for most existing CAx tools. Only recently PDM systems have gained a significant function in product development. But since they don't offer a product model themselves, new required functionalities involving non-geometric product characteristics and properties have been incorporated into CAD systems, emphasising their role as "master system". But this development has bred new problems:

- The concept of CAD systems is clearly oriented to geometry processing, and thus all additional product characteristics and properties need to be connected to geometry elements. This impedes a continuous support of the development process, since in "early phases" there is not much geometry (very early: no geometry at all), only required properties. Furthermore it does not seem sensible to structure all product-related knowledge along an assembly hierarchy.
- The "master system" should provide some means for the integration of CAx tools and organise the exchange of data between them. Some CAD vendors "solve" this problem by offering a whole range of further CAx systems which are adapted to their respective CAD system (and vice versa). The CAD customers (and even their component-suppliers) get extremely dependent on their respective vendor, as the system does not offer much support for the integration of third party tools. Unfortunately, the CAD vendors seem to have grabbed a bigger piece of the cake than they can eat, i.e. they don't have the resources to offer all the functionalities all of their customers need.

So a huge effort is put into the integration of third party software into the existing software environment by means of individually developed converters and other auxiliary constructions which in turn further increases the dependency, since the investment in these solutions is lost if and when the CAD system is changed.

It is usually the job of the PDM system to hold together all the loose ends, i.e. to integrate all the partly overlapping system-specific product models which have no common base (apart from geometry) and vary widely in respect to their degree of formalisation.

The biggest deficit is that today's PDM systems don't *know* anything about characteristics and properties of a product, let alone their interrelationships, so they can't really offer a continuous support of the whole product development process as it was discussed in section 2.

4. Development Potentials for PDM

In the authors' opinion, the future "master system" should be an (advanced) PDM system instead of a CAD system. It should be based on a comprehensive product model capable to represent all interesting characteristics and properties and their interrelationships during the whole life-cycle. The following subsections give an overview over the general architecture and advantages of such a system.

4.1 Contents of an Advanced Product Model

As the product development process aims at fulfilling required properties by means of solution patterns which imply certain characteristics, these are the central elements of an advanced product model. The required properties are the very first information appearing in the development of a new product, and they play an important role throughout the whole development process since they must be permanently evaluated against the predicted/verified properties of the solution (see figure 9). In fact, it is the gap between required and predicted properties which drives the development/design process. Therefore, three data structures make up the core of a product documentation from the PDD point of view (the term "list" is only used for the sake of simplicity and does not imply the form of implementation):

4.1.1 Property List

This list is actually a hierarchy of property classes and contains all considered properties which are mainly *required* properties. For each property the following should be known:

- Required value or interval ("Soll-value", if at all quantifiable)
- Currently estimated value ("Ist-value")
- References to methods/tools to predict/verify the property
- Accuracy of estimation
- References to solution patterns and characteristics influencing the property, preferably including sensitivity to those characteristics

- References to the life-cycle partner systems[3] influencing the property
- Persons/teams responsible for the property[4]

4.1.2 Solution Patterns and Characteristics

The solution patterns could be organised in cards/records of different abstraction levels where a card may represent a solution sketch, an organ, an assembly, a part or even a geometrical region ("feature" in the traditional sense). Each of these cards should contain the following information:

- Main characteristics, including geometry (if applicable)
- Assigned values for characteristics, including "rigidity" (value unknown/preliminary/fixed)
- References to affected properties (marked as intentional or side-effects)
- Constraints, i.e. references to connected characteristics
- References to methods/tools to determine appropriate values for the characteristics based on the values of related properties
- References to alternative solutions/solution patterns
- Remarks about why this solution pattern has been chosen, including references to constraints/ properties which have influenced this decision
- Persons/teams responsible for the solution patterns/characteristics[4]

4.1.3 List of Open Problems

In order to be able to find solutions which make use of synergy-effects, it seems sensible to keep a list of open problems. This list must be based on the results of the evaluation between "Ist"- and "Soll"-properties (see figure 9 (III/IV)) and shows the area of current work.
Typically there are three stages of current activities which can be characterised by the number of characteristics of the solution patterns which are known:

- No characteristics known at all: A combination of required properties is given, but no well-proven solution pattern is known. This corresponds to the conceptual stage in [VDI-2221] and only occurs in original design. The first task in the development process is to find or establish appropriate solution principles.
- Several alternative solution principles are known: The alternatives are to be evaluated in order to predict if and to which degree they are capable to fulfil the required properties. This usually involves considerable elaborating of these alternatives in the first place in order to prepare a sufficient base for an exact evaluation.
- A solution pattern is chosen and partly detailed, but in its current state of elaboration the characteristics as well as the properties are not yet completely determined. This corresponds to the detailing phase of [VDI-2221].

The list of open problems contains all properties which are not yet predicted or fulfilled and references to the solution patterns influencing these properties. It indicates the current state of elaboration of these solution patterns and constraints between them. The list of open problems can be derived from the lists of properties and solution patterns if these are kept up-to-date during the whole development/ design process, so it is not a constituent part of the product model, but a certain important view on this information.

---

[3] E.g. the manufacturing/assembly/testing systems for the life-cycle phase of production, but also things like the service/repair infrastructure, even the social environment the product has to be operated in. As was shown in [WeWe-01] the external properties (**ECj**) in the PDD product and process models introduced in section 2 of this article can be seen as the properties of the respective life-cycle partner systems.

[4] At present we find almost universally a distribution of responsibilities according to components and parts, i.e. according to solution patterns and their characteristics. Why don't we think at least in some respects about distributing responsibilities according to properties? The authors have the impression that a certain part of the restructuring in companies (e.g. matrix-organisation, cross-sectional responsibilities) stem exactly from that question.

4.2 Supporting the Design Process

4.2.1 Identifying Design Degrees of Freedom

With the formalised lists explained in section 4.1, possible conflicts or synergy-effects can be identified more easily in the early stages. Furthermore, they help to identify the design degrees of freedom which can be used to solve a problem. Design degrees of freedom are those characteristics of solution patterns or life-cycle partner systems which affect the considered properties and are not yet definitively determined, either explicitly or implicitly, by constraints.

4.2.2 Process Control

As was described in the sections above from different points of view, the actual driver of the product development/design process is the continuous evaluation of properties of the solution and even more so the comparison of the results against the required properties (see figure 9). Therefore, the most important pre-requisite for enhanced PDM systems is that they are put into a position to keep track of the sequence of the synthesis-analysis-evaluation cycles and their results. In an ideal scenario exactly this "knowledge" is used by the PDM system to control the process and not as is the case today pre-defined and in consequence rather inflexible process patterns.

4.2.3 Enhancing Cooperative Work/Simultaneous Engineering

An aspect of increasing importance in PDM development is workflow management. While in conventional approaches the assignment of persons to certain activities or development states is more or less fixed, a system based on an extended product model can associate persons to certain characteristics and property classes. This allows for a much more flexible determination of who is affected by a certain design decision.
A further advantage of having ***one*** consistent product model covering the whole development process is that teams responsible for different life-cycle phases of the product have simultaneous access to the current state of the design and therefore can identify conflicts much earlier than in a conventional procedure, where a product model is passed on to the "next" team only when a certain state of maturity is achieved.
The development process in big teams is usually an iterative procedure: The developers/designers tend to create a solution which is mainly oriented by functional aspects in the first place. Experts for other life-cycle phases (which frequently are the developers/designers/providers of the respective life-cycle partner systems) evaluate the design from their point of view and ask for changes. The developers/designers responsible for the product modify the design and it is re-evaluated.
Since the time to market is being constantly decreased, there is less time for these iterations. Consequently there is a growing pressure on the developers/designers to get everything right in the first place. But due to the big number of relevant life-cycle partner systems and corresponding properties, acquiring all the necessary expertise would overburden the designers.
The developers/designers can only be supported if the respective experts can participate in crucial decisions from the beginning. This can be achieved by modelling the relationships between characteristics and affected properties (e.g. by referring to the persons responsible for crucial properties in the property list, see section 4.1.1!). An additional benefit can be achieved by referring to the methods/ tools to evaluate the effects of design decisions to certain properties directly from the product model.
The PDM system could also handle parallel design alternatives to offer the experts involved a possibility to take part in the selection of an alternative and, in consequence, to avoid the elaboration of solutions which are not viable. The number of required iterations should significantly decrease if all experts concerned can comment on all design alternatives from the beginning. The ability of the chosen product model to represent solutions ***before*** the exact geometry is determined, is vital for conveying those early alternatives.

4.2.4 Integration of CAx tools

If the “master system” is based on a comprehensive and consistent product model, the roles of other CAx (including CAD) systems change significantly. Ideally they would not contain redundant parts of the product model, but rather act as “service-providers” which administrate specific parts of the product model and determine certain properties (analysis) or characteristics (synthesis).
So, the CAD system would still handle geometry, but nothing else. It would offer direct access even to parts of the geometry by means of standardised procedures which can be addressed and utilised by the master system. The master system would only handle “pointers” to the geometry (parts) representing certain solution patterns. By means of these pointers it could pick the respective geometry and pass it on to another system which might evaluate a certain property. It might as well call procedures to extract certain properties (like volume or cross section area) directly from the CAD model.
The PDD view facilitates continuous computer support in so far as it proposes that there is no fundamental difference between “early” and “late” phases. The integration of non-geometric characteristics and properties in a product model allows for a representation of requirements/specifications from the beginning, i.e. at a time when maybe no characteristics are known at all. The successive determination of solution patterns, assignment of their characteristics, prediction of properties and comparison with the specifications (required properties) is a continuous process which can completely be supported by the computer if an appropriate product model is provided.
Important in the “early phases” are relations ($\mathbf{R_j}$, $\mathbf{R_j^{-1}}$, see figures 2 and 3) which are based on very few characteristics and properties and often can’t be put in equations. Typically this kind of relations is not yet put in computer models, but hidden in the minds of experienced designers. Thus it is important to find an appropriate representation for rather vague interdependencies which is close to natural language but still accessible for algorithms. Semantic nets, which are increasingly discussed in the public, could offer such a representation.

4.2.5 Design Re-Use

One of the most obvious benefits of an extended product model is the use of formalised solution patterns (templates) and past designs which convey design knowledge. This mainly refers to the kind of advanced features which have been developed in research [Webe-96, VaWe-99a, VaWe-99b]. While a completely explicit product model enables the distributed development and arbitrary combination of such features, the formal declaration of their characteristics and properties allows for well-structured catalogues and partly automated selection steps of solution patterns for required properties.
A similar field is the formalised representation of life-cycle partner systems which can also be supplied by means of electronic catalogues. An important advantage of such a formalised representation is that the reasons for certain constraints can be tracked and thus the applicability of these constraints can be verified. Furthermore, potential design degrees of freedoms can be identified in life-cycle partner systems which is a very important aspect in simultaneous engineering.

4.3 Implementation Issues

It is well known that an advanced support of development activities by means of computers is mainly a problem of the employed product model. The gap between the possibilities demonstrated in research systems and the poor support commercial CAx systems can provide is mainly due to the lack of a sensible standardised product model. The immense effort spent on the definition and implementation of the STEP standard and the fact that STEP is still mainly used to exchange geometry and aggregtion information imply that a conventional approach can not solve the product model issue.
The key to a new kind of product model covering a wide range of characteristics and properties may be object-oriented modelling with soft-coded algorithms. Such a completely explicit product model at the same time offers a high flexibility and a high degree of formalisation by including the necessary knowledge into the product model instead of into the CAx systems [WeWe-98, WeWe-99,Wern-01].

This concept is essentially based on a modelling language which is able to describe any characteristics/properties as well as the corresponding algorithms (relations) and which is interpreted by the master system. The role of the other CAx systems in this context is to provide basic data structures and functions to manipulate them. Since the master system can delegate the more sophisticated tasks to the respective CAx systems, it needs only a very basic functionality itself and therefore can be implemented with relatively low effort.
The product model itself is highly formalised and contained in knowledge modules which represent the current design as well as generic solution patterns (templates, "features"), OEM-parts, life-cycle partner systems or even tools to evaluate relations ($\mathbf{R_j}$, $\mathbf{R_j^{-1}}$) which are not covered by standard CAx systems. It is very likely that with the establishment of such a master system a new market segment for knowledge modules will emerge. Since the knowledge modules can be arbitrarily added to a product model without changes in the "master system", the free market may solve the problem of supplying any required functionality which a single CAD supplier can never achieve.
The knowledge modules define all relevant characteristics and properties of the current design as well as the case-specific relations ($\mathbf{R_j}$, $\mathbf{R_j^{-1}}$). They can also provide knowledge on life-cycle partner systems (at least incorporate relevant external conditions $\mathbf{EC_j}$ which are determined by those partner-systems) and modelling conditions ($\mathbf{MC_j}$). The master system interprets this language and links formal relations ($\mathbf{R_j}$) to the corresponding functionality of different specialised CAx systems.

> Example: A relation defined in the product model might be "part.weight = part.volume × material. density". While the material density is a simple scalar characteristic that may be stored in the master system, the exact determination of the part's volume is a task which should be carried out by a CAD system. The master system therefore needs to pass through the request for the part's volume to the respective CAD system.

Other relations may be formulated more vaguely, sometimes even verbally in text documents. This is the classic domain of PDM systems, and the challenge is to incorporate these into a highly formalised product model as described above. While a computer won't be able to understand the content of a text document before long, it is possible to link the document to the corresponding characteristics and properties.

5. Summary

Today, many aspects of the product life-cycle can already supported by different CAx tools. Since they often use more or less incompatible data or product models, it is necessary to work with converters or to stick to the software (family) of one CAx supplier in order to be able to use product data throughout the whole development/design process or even to re-use data/solutions from the past.
In this article, a new approach of modelling products and, consequently, product development processes is introduced (PDD – property-driven development/design), and conclusions are drawn from it concerning architecture, functionalities and realisation of advanced PDM systems. The proposed advanced PDM system uses a consistent, redundancy-free product model which not only handles the characteristics of the solution, but also its properties and the relations between the two categories.
Thus the PDM system could help to show how a change of the characteristics would affect the properties and vice-versa. The system integrates design tools for different phases of product development. Collaborative and simultaneous engineering is supported by providing access to the current state of development/design for all people involved in the process and by handling different design alternatives.
Finally, some implementation issues are discussed, based on the authors' experiences from realising prototype systems in related fields.

References

[Ehrl-95] Ehrlenspiel, K.: Integrierte Produktentwicklung. Hanser, München, 1995.
[Hubk-73] Hubka, V.: Theorie der Maschinensysteme. Springer, Belin, 1973 (1. ed.).
[Hubk-84] Hubka, V.: Theorie technischer Systeme. Springer, Berlin, 1984 (2. ed. of [Hubk73]).
[HuEd-92] Hubka, V.; Eder, W.E.: Einführung in die Konstruktionswissenschaft. Springer, Berlin, 1992.
[HuEd-96] Hubka, V.; Eder, W.E.: Design Science. Springer-Verlag, Berlin, 1996.

[Suh-90] Suh, N.P.: The Principles of Design. Oxford University Press, 1990.

[VaWe-99a] Vajna, S., Weber, C.: Teilmodelle im Konstruktionsprozeß – Bindeglied zwischen methodischer und rechnerunterstützter Konstruktion. Konstruktion 51 (1999) 4, p. 46-50.

[VaWe-99b] Vajna, S.; Weber, C.: Sequenzarme Konstruktion mit Teilmodellen – Ein Beitrag zur Evolution des Konstruktionsprozesses. Konstruktion 51 (1999) 5, p. 35-38.

[VDI-2221/86] VDI-Richtlinie 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. VDI, Düsseldorf, 1986.

[VDI-2221/87] VDI-Guideline 2221: Systematic Approach to the Design of Technical Systems and Products. VDI, Düsseldorf, 1987 (English version of [VDI-2221/86]).

[Webe-96] Weber, C.: What is a Feature and What is its Use? Proceedings of the 29th International Symposium on Automotive Technology and Automation 1996 (ISATA 96), Florence, 1996, p. 109-116.

[Wern-01] Werner, H.: Integration von CAx-Funktionalitäten in einem neuartigen Konstruktionssystem. Dissertation, Universität des Saarlandes, Saarbrücken, 2001.

[WeWe-00] Weber, C.; Werner, H.: Klassifizierung von CAx-Werkzeugen für die Produktentwicklung auf der Basis eines neuartigen Produkt- und Prozessmodells. Proceedings of the 11th Symposium "DfX", Erlangen/Schnaittach, 2000, p. 126-143.

[WeWe-01] Weber, C.; Werner, H.: Schlußfolgerungen für „Design for X“ (DfX) aus der Perspektive eines neuen Ansatzes zur Modellierung von Produkten und Produktentwicklungsprozessen. Proceedings of the 12th Symposium "DfX", Erlangen/Neukirchen, 2001, p. 37-48.

[WeWe-98] Weber, C.; Werner, H.: Ligo – an Object-Oriented Modelling Tool for Integrated Product Development. Proceedings of the 2nd International Workshop "Integrated Product Development" (IPD 98), Magdeburg, 1998, p. 44-51.

[WeWe-99] Weber, C.; Werner, H.: Implizite und explizite Bestandteile des Produktmodells und ihre Bedeutung für die Entwicklung von CAx-Systemen.. Proceedings of the 10th Symposium „Fertigungsgerechtes Konstruieren“, Erlangen/Schnaittach, 1999, p. 1-6.

Prof. Dr.-Ing. Christian Weber
Saarland University, Engineering Design/CAD
PO Box 15 11 50
D – 66041 Saarbrücken, Germany
Phone: +49 / (0)681 / 302 – 3075; Fax: +49 / (0)681 / 302 – 4858
Email: weber@cad.uni-saarland.de
URL: http://www.cad.uni-sb.d