INTERNATIONAL DESIGN CONFERENCE - DESIGN 2008
Dubrovnik - Croatia, May 19 - 22, 2008.

DESIGN 2008

APPROACH ON THE CONTROL OF ITERATIONS IN THE MULTIDISCIPLINARY DEVELOPMENT OF TECHNICAL SYSTEMS

H. Krehmer, C. Stöber and H. Meerkamm

Keywords: multidisciplinary product development, concurrent engineering , iterations, property-driven parameter exchange

1. Introduction

Specific knowledge of one domain alone is no longer sufficient for the successful development of modern technical systems. In the course of more and more increasing demands, for example made on speed, precision, robustness, autonomy and adaptability of technical systems, the complexity in product development rose continuously. Associated with that is an increased penetration of those systems by electrical, software-based and electronic components, whereby also multidisciplinary orientation and combined with it the challenges in the development of technical systems steadily increase. Not only due to the many times unclear definition of interfaces between the spatially spread out development partners of the individual domains mechanical engineering, automatic control engineering, software design, electrical engineering, and information technology, but also through the iterative progression of the product development process itself, interdisciplinary product development is made considerably more difficult. Since however a seamless collaboration of all involved disciplines as well as the reliable control of the products' complexity, of related processes, as well as of iterations represent a basic prerequisite for successful product development, it is necessary to interdisciplinary integrate different approaches from all faculties for the supporting of development. [Paetzold06].

2. Challenge in the multidisciplinary development of technical systems

Each domain disposes of its own vocabulary of concepts and experiences and often of methods and forms of description that have grown for decades [VDI2206]. So have different specific product perceptions solidified in the domains, adequate to the assignment of tasks, which each put one of the aspects "structure", "function", or "behavior" of the product in the focus of considerations. These perceptions are – possibly without being explicitly pronounced – deliberated in each domain, however clear domain-specific emphases will result: Mechanical engineering focuses prior-ranking on structure due to the great significance of geometry. In electrical engineering, behavior moves to the center of attention while for information technology first of all function is of importance. By *structure* of a system, its internal composition is understood, so for example of which components the system consists and how their geometric and relational correlations among each other are pronounced [Hubka84, VDI2221]. The structure of a product can be described in form of geometry and material specifications. Source code can be seen as structure in information technology for instance, while structure in terms of electrical engineering may be a switching circuit for example. This shows that structure is very heterogeneously pronounced in the different domains, i.e. the term structure is domain-specific differently interpreted and understood so that under a structure-oriented view on the

product, misunderstandings may quickly result. By *function* of a system, its task, its sense, its purpose is understood [Paetzold06, Hubka84]. Hence the function answers the question for what the system was created, or which effect is achieved by the system [Hubka76]. The function of a technical system demonstrates the coherence between input and output parameters of the system. Therefore the formulation of function itself is an abstract and solution-independent description. By *behavior* of a technical system is understood, how that system interacts with its environment, so what the system does respectively has to do to fulfill its purpose (= function) [Hubka84, Ariyo06, Gero04]. Consequently from the behavior of a system not only emanates what its task is, but at the same time how this task is fulfilled. Therefore, description of behavior is not solution-independent in contrast to the function description. The requirements made on a technical system determine its function. Starting from that function description, a certain desired behavior is defined in all domains, which is attempted to be achieved in the following by structural determinations. By the fact that the defined structure determines the behavior, it also fulfills its function. Thus the behavior of a system results form the chosen solution and is unambiguously determined by the system's structure. The other way around however, from the behavior respectively function, no conclusion can be made on the structure: The same function or the same behavior respectively can theoretically be achieved by an unlimited number of different structural characteristics, and one and the same structure can fulfill several functions under different circumstances. So no direct causal coherence exists between the structure and the function of a technical system (Figure 1).

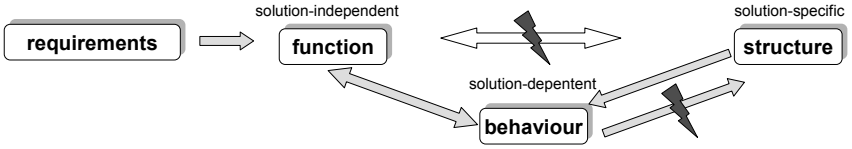

Figure 1. Structure - function- behavior

The increasing requirements made on the functionality of products and the complexity of the product and of the processes associated with it can only be counteracted by splitting up development work among specialized competent departments: By means of such specialization, the know-how specific to the development of complex products can be built up, and methods and tools adapted to the situation can be applied. Such shared development sometimes takes place spread out across different departments, company branches, or even different companies (concurrent engineering). For that purpose it is necessary to divide the product up into domain-specifically processable modules, which is contradictory to the approach of mutually accomplished inter-domain product development. The difficulty in the coordination between mutual and specialty-specific, i.e. split up development lies in finding the optimum distribution of individual tasks (differentiation) while at the same time assuring the compatibility of components developed separately from each other (integration). For that purpose first of all the interfaces between the individual structural elements to be developed must be precisely defined and observed, domain-internally as well as inter-domain. The interfaces resulting from such modularization within the product itself often also constitute interfaces within the product development process since modules mostly are developed in parallel to each other and therefore, the individual subprocesses need to be coordinated.

Due to the network of dependencies arising between the individual work steps the increased complexity of modern technical systems also entails in an increased complexity of the entire product development process. Therefore the course of the product development process is not foreseeable in detail, is difficult to plan, and only develops iteratively in its progression. Typical for human problem-solving is a permanent change between abstracting and concretizing as well as between concentrating on the whole or on its parts [VDI2221].

Consequently also some jumping back to previous sections of the development process takes place. Especially for complex products the solution cannot be found in one step. The iterative procedure of the product development process materializes from boundary conditions, faulty decisions made on basis of unclear or uncertain assumptions only becoming concrete in the course of it. Furthermore

iterations are induced due to the high degree of division of labor and the lacking communication associated with it, or by optimization potentials being recognized in later work steps only. In most cases cognizances that influence decisions made during previous steps retrospectively are only acquired during later phases. Further triggers for iterations may be, among others, functional improvements, elimination of faults, changes in customers' requirements, or changing needs of the market. Thereby the development period extends and by iterations, the traceablility of the process as well as the reuse of the found solution under similar boundary conditions are made considerably more difficult. Iteration on the other hand raises the level of information for a step to be gone through again and is therefore rather to be considered as a learning process than an unnecessary detour. By these comments it becomes clear how important it is to support the developer in situations in which the necessity of iteration arises.

3. Approach on the control of iterations

The aim of this approach is to support the developer in the adequate handling of iterations. This is required by innovation cycles of technical systems getting shorter and shorter as well as by the permanently increasing quality demands on part of customers. It is impeded by the varied characteristics of iterations, which reflects in different triggers and in the extend of effects they have on the product development process.

3.1 Classification of iterations

To enable supporting the developer in the dealing with such iterations as appropriate to the situation, it is necessary to make the developer aware of the fact that iterations can be traced back to a limited number of classes. Classification may help to distinguish unnecessary iterations from helpful ones and to avoid time-consuming and cost-intensive detours. The following main groups can be identified:

- **"Small iteration": Quantitative approximation**
  Because of the fact that a solution cannot always be found in one step, but the developer instead has to approach the final solution iteratively through several partial steps, sometimes to be gone through repeatedly, the first type of iteration arises. The sheet thickness of a pressing may be taken as an example, which is altered until the required stiffness is achieved at the allowed weight. Since a multiple passing through of one or a limited number of partial steps is the consequence of such iteration, this type of iteration merely means a limited fallback in the product development process and therefore has relatively little effects. Therefore it is called here "small iteration".
- **"Large iteration": Changing of requirements or of information basis respectively**
  The second type of iterations emerges from a change in the information basis or in the requirements respectively: On the one hand boundary conditions and requirements being unclear in the beginning concretize only in the course of the process. On the other hand new cognitions regarding the effects on the total system result from the fixing of a certain solution so that a completion of the data basis is reached only in the progression of the process. From this new cognitions or requirement changes may result, by which eventually iterations are induced. In the processing of these iterations in principle the whole process is to be passed through again, more specifically the results of all partial steps are to be verified again in respect of the fulfillment of the modified requirements. From the point from which the results of partial steps achieved up to now diverge from the modified requirements, the product development process must be run through again. This means that such iteration may entail in a return to the beginning of the product development process. Therefore the denomination "large iteration" is used here. Due to the complexity of modern technical systems, in the course of integration of individual components iterations may become necessary for the purpose of their adaptation to each other. Finally this iteration is caused by the case of changed requirements respectively information basis, since a revision of one or more partial systems ultimately results from a change in the requirements made on the individual partial

systems to be changed, or from an expanded information basis respectively. For this reason the product development process for those partial systems is to be run though again.

3.2 Methods of resolution for the control of iterations

Now based on the classification of iterations it is possible to provide the developer with a practical method of resolution for each identified class of iterations, by which the developer can be purposefully supported. These methods of resolution are highlighted in the following.

3.2.1 Property-driven development approach according to Weber for supporting the "small iteration"

Iterations in which one or more partial steps are run through several times can be supported by means of the PDD approach according to Weber [Weber05]. Basis of this approach is the CPM concept (characteristic property modeling), which provides for the strict separation of properties and characteristics: In this connection characteristics are understood as specifications concerning structure, material, shape, or condition of the product. Those can be determined directly by the developer and therefore, define the product. The properties of a product however describe its behavior. Directly measurable and assessable properties (safety, manufacturability, environmental compatibility, esthetic aspects) are ranking among this. The properties of a product cannot directly be determined by the developer, they result from the determination of characteristics.
The approach of property-driven development (PDD) provides here for a model of synthesis, analysis, comparison, and assessment to be run through in multiple cycles. By means of comparison, a conclusion on weak points of the product in its momentary configuration can be made: In case a great difference between required and realized properties results, the characteristics are not determined appropriately. So the difference between nominal and actual condition represents the process-driving parameter. Criterion for completion of the product development process is a sufficiently small difference between nominal and actual condition. In this way the PDD approach supports the developer in the approximation to the final solution by multiple passing through the cycle.

3.2.2 Property-driven parameter exchange for the accomplishment of the "large iteration"

An approach that wants to support the holistic development process of multidisciplinary technical systems must allow the domains to proceed in the normal course of action and with the implementation of methods and tools usual in that domain. At the same time this means however, that the detailing of partial systems takes place separated from each other in the individual domains. Resulting thereof is the requirement that interfaces in the process must be clearly defined, information must be communicated to a sufficient extend. Otherwise the risk of unnecessary iterations increases. To support the class of large iterations, on the one hand the PDD approach, on the other the property-driven parameter exchange (see next section) is applied. While the PDD approach enables the early recognition of additional properties through the permanently repeating analysis and therefore, finally promotes the early completion of the information basis, the communication respectively distribution of those relevant information is enhanced by the property-driven parameter exchange. By completion of the provided information (e.g. through parameters from datasheets) and their subsequent classification in internal and external parameters, interdependencies between individual partial solutions can be discovered and mapped in due time while transparency in the product development process is facilitated.

The property-driven parameter exchange

In the multidisciplinary development of technical systems taking place spread out among the domains, an increased need exists to keep the data basis at all times up-to-date and to facilitate continuous accessibility of such data to all developers involved. Consequently, the inter-domain circulation and distribution of relevant information must be integrated into any procedure for the multidisciplinary development of technical systems. In order to guide the developer in this matter, the property-driven parameter exchange was developed. At this the exchange of information takes place in form of external and internal parameters. External parameters are the ones that influence each partial system

and each department as they constitute inter-domain boundary conditions for all partial systems and therefore, need to be considered in the development of all partial systems by all persons involved. In addition to operational conditions (climate, percussions, vibrations, electric and/or magnetic fields), also safety regulations, the required lifetime, generally applicable boundary conditions such as weight, material, and costs as well as restrictions due to manufacture, assembly, and maintenance are belonging to this. Also influences caused by assembly (e.g. heat development from welding, accessibility for tooling) need to be considered here. Internal parameters however are playing a role in the selection and dimensioning of the respective partial system, as they have no direct influence on other partial systems, but only become effective internally within the partial system. To this belong for example in case of sensors the quantities to be measured, specific measuring ranges, and the required resolution, for actors in turn occurring forces for instance, certain efficiencies, or auxiliary energies needed.

In setting up a hierarchic product structure, the elements of components are on the bottom level of the structure. The characteristics of the element represent the internal parameters, while the element properties resulting thereof correspond to the external parameters. At the higher levels of the product structure, at which elements are combined to modules or partial systems, external parameters of one level may become internal parameters of the superior level. However they also might retain their status of being external parameters at the higher level. Since internal parameters only have influence on the considered partial system level itself, they don't reappear in the higher levels. Important in this exchange of information is that parameter exchange is taking place on the same level, i.e. that external parameters of a component are serving as input information for other components of the same level. Once these elements or components are composed to a module or partial system, again parameters need to be identified on this newly-developed next-higher hierarchic level, which following are to be differentiated into internal and external parameters. After that information identified as external parameters must be communicated at the same level again.

In order to illustrate this approach, that parameter classification is depicted in the following by the example of a transmission (Figure 2). This classification may be applied analogously to all technical systems. In this example a group of components consisting of shaft, bearing, locking ring, distance sleeve, slotted nut, and lock washer is considered. The internal parameters of these elements are for instance their materials, for bearings for example the shape and the number of rolling elements. To enable a reliable coordination between the individual systems (elements or groups of components), the exchange of external parameters is made. Here external parameters are dimensions, for the shaft in addition the transmittable torque for example, for the bearing the required lubrication, the RPM limit and the permissible bearing load. For the lock washer, shape and number of lugs can be identified as external parameters, and for the slotted nut thread and the main dimensions. In the assembly of these components the internal parameters of the components disappear, and the components involved and their external parameters become internal parameters. Individual external parameters of the components however remain external parameters also in the assembly, thus in the level above, like the required lubrication, the bearing capacity, the maximum transmittable torque, as well as the maximum rotational speed permissible. Analog to that, the classification of parameters is also to be made at the other levels. In the example of the transmission, the load capacity, the maximum rotational speed permissible for the bearings as well as the maximum transmittable torque of the shaft can still represent external parameters even at the level of the total system. This example highlights the potential of the property-driven parameter exchange in respect of the coordination of partial systems among each other, not only with regard to usage, but also regarding assembly, production, and other sectors of the product lifecycle (Design for X).

Certain parameters represent influencing or disturbance variables for other components that may entail in interdependencies and negative effects on the behavior of the total system, which are not foreseeable without building a physical prototype or only under massive application of virtual safeguarding measures. The particularly with mechatronic systems in the course of system integration frequently occurring iterations finally are just a consequence resulting from lack of coordination between parallel partial developments, and increase the complexity and unstructurability of the product development process. Those iterations can also be supported by the approach for an early

property-driven parameter exchange: By early communication and circulation of all relevant information, problems in the later integration of partial systems are prevented.
Due to the fact that parameters, which are exchanged, are having influence on the properties of the total system, this is a case of property-driven parameter exchange. An important aspect of this approach is the fact that individual domains remain autonomous and can use their accustomed development methods, but in doing so at the same time supply the other involved development departments with the information necessary for a smooth process flow. Living an open, interdisciplinary cooperation is thereby enabled.

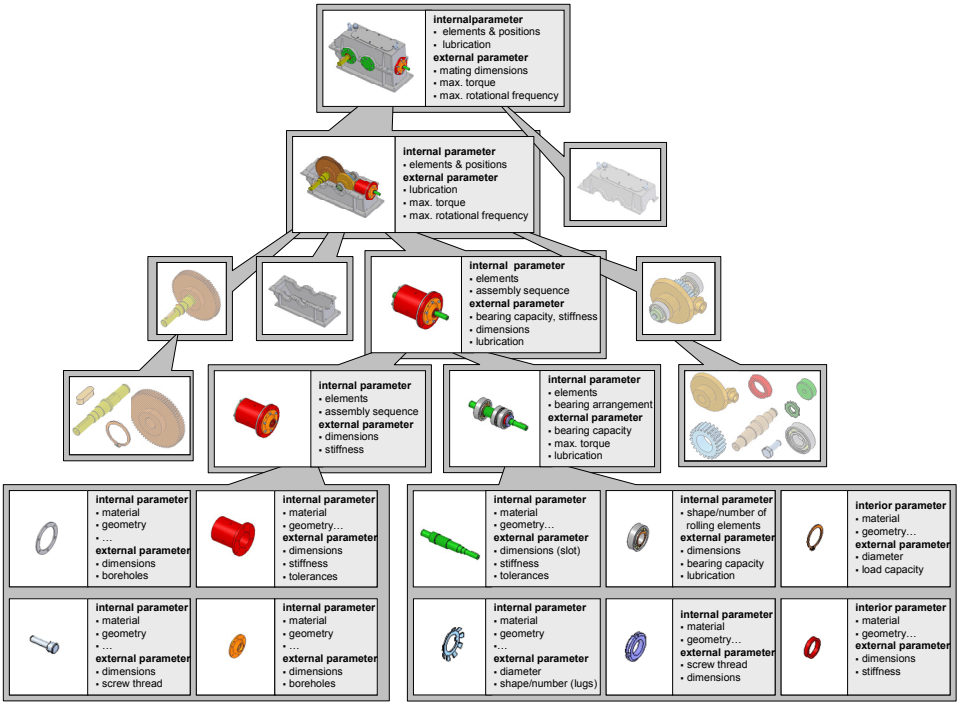

Figure 2. Structure of a transmission

3.2.3 Approach on control of iterations

The procedure according to PDD and property-driven parameter exchange provide for a microcycle, which is cycling over and over independent from the progress of the process. The scheme of this procedure is shown in Figure 3. During each run of the microcycle after the synthesis of characteristics is made, the thereof resulting properties are to be analyzed. Important is here that available knowledge from experience as well as the information about neighboring partial systems resulting from property-driven parameter exchange enter in this analysis. In the next step of the microcycle an assessment of the realized properties is made, which were discovered through the analysis. Here the following method is to be applied:
At first is checked whether any of the identified properties have any negative effects on the total system. If negative effects are detected, the characteristics defined in the last synthesis must be reconsidered again. If no negative effects are detected, it must be checked whether the properties of the system in its momentary configuration conform to the properties required. Is this the case, a verification of the degree of fulfillment of these requirements will follow ("completely" or "partially fulfilled"). In case the identified properties deviate from the required ones, they can be categorized as "synergy", "counteractive", or "neutral".

Depending on the preceding classification of properties, now the conclusion respectively the decision about how to proceed from there is made. Has the assessment of properties arrived at a positive result (classification as "completely fulfilled", "synergy", or "neutral"), after the property-driven parameter exchange the characteristics synthesis for the next property can be started. The results of the property-driven parameter exchange are made available to other partial systems in form of knowledge from experience and external parameters. In case a required property was "partially fulfilled", another characteristics synthesis is to be made for that property ("small iteration"). In case a realized property does not comply with the required ones, and was it classified as "counteractive", another characteristics synthesis for that same property ("small iteration"), or for a preceding one respectively ("large iteration") is to be made.

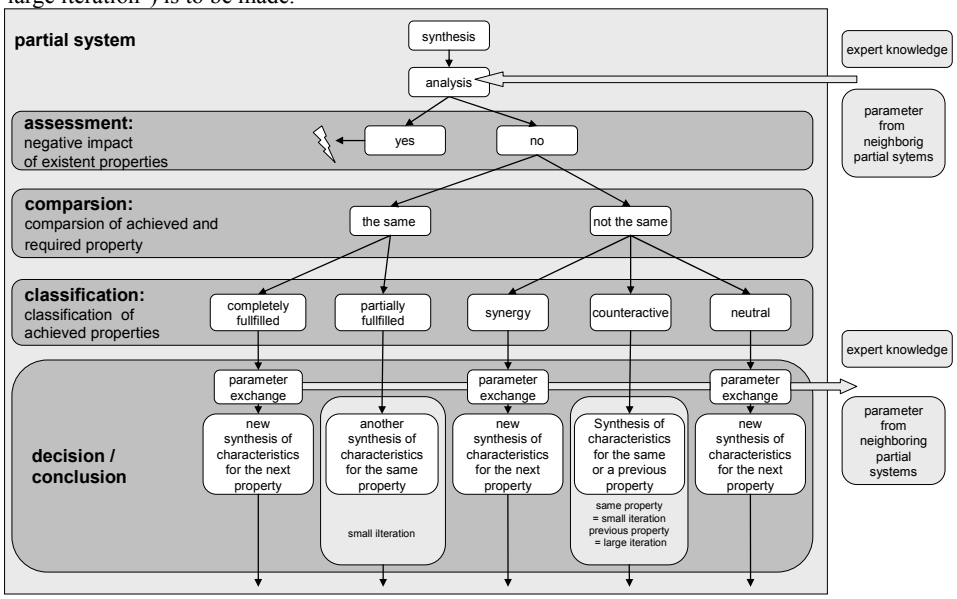

Figure 3. Microcycle based on property-driven parameter exchange

4. Conclusions

By the presented procedure, the developer is supported in the handling of iterations, which inevitably occur during product development. On the one hand the developer is aided in the accomplishment of those inevitable iterations by getting a procedure on hand that allows at an early stage reliably recognizing and evaluating effects of decisions made. In addition, the early communication between development partners is promoted. On the other hand unnecessary iterations are counteracted through the consequent application of this procedure, which first of all are advantaged by the frequently lacking communication.

5. Further Research

The classification of iterations is to be advanced in further research activities. In the process of it guidelines are to be elaborated, which – depending on the trigger and the type of iteration – indicate at which point of progression of the product development process it is necessary to return, respectively which and to what extend work steps are to be cycled though again. In continuative studies the correlations between characteristics and properties are to be detailed, also with regard to the influence by Design for X, to broaden the present approach.

Acknowledgements

This article originated within the scope of the research network ForFlow patronized by the Bavarian Research Foundation in cooperation of both partial projects 1.1 "Representation of Multidisciplinary Products by a Functional Mock-up" and 2.2 "Influence of the Aspects of Design for X on the Product Development Process". The research foundation has the objective of supporting interdisciplinary product development as appropriate to situation.

References

Ariyo, O. O., Eckert, C. M.; Clarkson, P. J., "On the use of functions, behaviour and structural relations as cues for engineering change prediction", Proceedings on the 9th International Design Conference - DESIGN 2006, D. Marjanović (Ed.), FMENA, Zagreb, 2006, pp. 773-781.

Gero, J. S. , Kannengiesser, U., "The situated function-behaviour-stucture framework", in: Gero, J. S. and Kannengiesser, U.: The situated Function-Behaviour-Structure framework, Design Studies 25(4), 373-391. 2004, Quelle:http://wwwpeople.arch.usyd.edu.au/~john/publications/2004.html (16. Januar 2007).

Gausemeier, J., Ebbesmeyer, P., Kallmeyer, F., "Produktinnovation - Strategische Planung und Entwicklung der Produkte von morgen", Carl Hanser Verlag München Wien, 2001.

Hubka V., "Theorie der Konstruktionsprozesse – Analyse der Konstruktionstätigkeit", Springer Verlag Berlin Heidelberg, 1976.

Hubka V., "Theorie technischer Systeme – Grundlagen einer wissenschaftlichen Konstruktionslehre", 2. Auflage, Springer Verlag Berlin Heidelberg, 1984

Krause, F., Franke, H.-J., Gausemerier, J., "Innovationspotentiale in der Produktentwicklung", Carl Hanser Verlag München Wien, 2007.

Paetzold, K., "Ansätze für eine funktionale Repräsentation multidisziplinärer Produkte", in: Meerkamm, H. (ed.), Beiträge zum „17. Symposium Design for X", 12.-13. Oktober 2006, Erlangen, Lehrstuhl für Konstruktionstechnik, 2006.

Meerkamm, H., Paetzold, K.., "ForFlow – Bayerischer Forschungsverbund für Prozess- und Workflowunterstüt-zung zur Planung und Steuerung der Abläufe in der Produktentwicklung – Langantrag", Lehrstuhl für Konstruktionstechnik; Universität Erlangen-Nürnberg. Printy GmbH München. 2006.

N.N, VDI-Richtlinie 2221, "Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte", VDI-Handbuch Konstruktion, Berlin. Beuth Verlag GmbH Berlin Düsseldorf, 1993.

N.N., VDI-Richtlinie 2206, "Entwicklungsmethodik für mechatronische Systeme", Beuth Verlag GmbH Berlin Düsseldorf, 2004

Weber, C., "CPM/PDD - An extended theoretical approach to modelling products and product development processes", in: Proceedings of the 2nd German-Israeli Symposium on advances in methods and systems for development of product and processes. TU Berlin / Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik (IPK), 07.-08. Juli 2005, p.159-179, Fraunhofer-IRB-Verlag, Stuttgart 2005.

Wynn, D. C., Eckert, C. M., Clarkson, P. J., "Modelling and Simulating Development Processes", Proceedings on the 15th International Conference on Engineering Design 2005 (ICED 05), H. Samuel;W. Lewis (Ed.), Melbourne, 15.-18. August 2005.

Dipl.-Ing. Hartmut Krehmer
University Erlangen-Nuremberg, Department for Engineering Design
Martensstraße 9, D-91058 Erlangen, Germany
Tel..: +49-(0)9131-8523216
Fax.: +49-(0)9131-8523223
Email: krehmer@mfk.uni-erlangen.de
URL: http://www.mfk.uni-erlangen.de/