REQUIREMENTS FOR AN ADVANCED DATABASE TRANSACTION MODEL TO SUPPORT DESIGN FOR MANUFACTURING

P.A.C. Verkoulen, F.J. Faase, A.W. Selders, P.J.J. Oude Egberink
University of Twente[1]
Department of Computer Science
P.O. Box 217
NL-7500 AE Enschede
the Netherlands
e-mail: P.Verkoulen@cs.utwente.nl
phone: +31-53-894312
fax: +31-53-339605

ABSTRACT. The ESPRIT-III project TRANSCOOP aims at providing computer support for cooperative tasks. Design for Manufacturing (DfM) is one of the application areas that is being studied within TRANSCOOP. Within the project, we are developing a specification language for describing cooperative tasks, an advanced database transaction model for supporting such tasks and computer support for the specification language and the transaction model.
It has been recognised that traditional transaction models are not sufficient for this purpose. This has resulted in a number of higher-level transaction models. Problem is that these models are often rather *ad hoc*: we are not aware of results that state systematically what is missing from traditional transaction models.
In our paper, we provide an analysis of the requirements DfM imposes on such a transaction model and its accompanying specification language. We do this by studying DfM scenarios in three case studies: two from practise and one from literature. The results are also useful outside the TRANSCOOP project.

Keywords: technical aspects of information systems and databases; computer-aided design, engineering and manufacturing; computer support for design

1. INTRODUCTION

During the last few years, much attention has been paid to new approaches for product and process design. A well-known approach is *Design for Manufacturing (DfM)*, which can be seen as a minimum variant of *Concurrent Engineering*. DfM especially integrates design, engineering and manufacturing of products. As a result, specialists from these fields have to cooperate closely. For making this cooperation feasible, computer support is absolutely necessary.

[1]This work is done in the ESPRIT project TRANSCOOP (EP 8012), which is partially funded by the Commission of the European Communities. The partners in the TRANSCOOP project are GMD (Germany), University of Twente (The Netherlands), and VTT (Finland).

The ESPRIT-III project TRANSCOOP[2] aims at providing computer support for cooperative tasks. DfM is one of the application areas that is being studied within TRANSCOOP. Within the project, we are developing a specification language for describing cooperative tasks, an advanced database transaction model for supporting such tasks and computer support for the specification language and the transaction model.
In the area of DfM, cooperation is one of the key issues. Specialists from several departments have to work together to accomplish the design of new products and processes. This may be difficult, as those specialists have different technical knowledge and look at the problems from different perspectives. In doing this, information has to be shared by them, in such a way that consistency is guaranteed.
Computer support for the afore-mentioned cooperative effort would make a successful cooperation more feasible. However, conventional database transaction models do not support cooperation. The reason for this is that they do not support real concurrent usage of information, as they guarantee consistency by ensuring that no two users access the same data at the same time. This scenario may be applicable in situations where users only need to access common data very shortly, like credit/debit transactions in a banking environment. However, for cooperative scenarios, like DfM, this is not satisfactory.
So, we need an advanced database transaction model supporting cooperative tasks. Also, a specification language for describing such tasks is required, in order to be able to apply this transaction model. It has been recognised before that traditional transaction models are not sufficient for this purpose. This has resulted in a number of higher-level transaction models. Problem is that these models are often rather *ad hoc*: we are not aware of results that state systematically what is missing from traditional transaction models. In our paper, we provide an analysis of the requirements DfM imposes on such a transaction model and its accompanying specification language.

The rest of this paper looks as follows. We continue with a short explanation of DfM and the differences with the traditional design approach. Then we describe the cases that have been studied to obtain requirements for a cooperative transaction model and the accompanying specification language. In Section 3, we present the requirements we have found. We refer to the TRANSCOOP deliverables [6,8] for more detailed information. We conlude this paper by summarising what has been accomplished and indicating further work.

1.1 Design for Manufacturing

When engineering new products, different kinds of approaches can be used. The traditional approach towards engineering new products is one in which actions are executed in sequence. Over time, however, market demands change and the traditional sequential approach no longer seemed to be the best one. New approaches like DfM evolved.
To understand the reasons why DfM is applied, first the characteristics and limitations of the traditional process are discussed. After that, the characteristics that distinguish the DfM approach from the traditional approach are presented.

1.1.1 Traditional process Traditionally the different specialists from design, production engineering, production planning and manufacturing work in a so-called "over the wall" approach. The order in which activities should be executed and the persons responsible are defined clearly. When specialists from a certain department finish their job, they "throw" their plan or product over the wall to the next department, which then can do their part of the job based on this plan or product. Usually, it appears that a subsequent department finds some shortcomings or impossibilities that result in a throwback of the plan or product to one of the previous departments. Consequently, an improved product or plan needs to be produced, increasing the

[2] TRANSCOOP stands for "Transaction Management Support for Cooperative Applications".

number of product development cycles. In the end, a completed product is delivered after having gone through a number of cycles. In Fig. 1, this process is depicted.

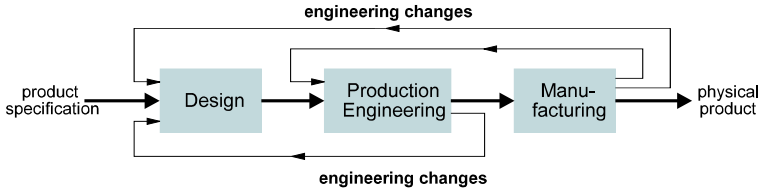

Figure 1: The steps in the traditional sequential product development process

The number of cycles and their wideness influence the time-to-market for a great deal. Because of the existing walls in the traditional approach, there is little information exchange and the manufacturing aspects are considered too less in the design phase. Problems in the design therefore sometimes are detected in a later stage of the product development process.
Besides being cost-effective and producing good quality, organisations need to be flexible, be innovative and reduce the time-to-market for their products to withstand the increasing competition. By being flexible, organisations are able to adapt to new situations in the market quickly, while innovativity is important as this creates opportunities to pull customers to an organisation's market.

1.1.2 Adapting the traditional process to current demands: applying DfM To better fulfill market demands like low price, high quality, flexibility and innovativity, the traditional sequential process needs to be adapted. This is done in the DfM way of working. Below, we discuss the characteristics that make DfM a better approach.
The main characteristic of DfM[3] as opposed to the design phase in the traditional process, is the concept of *early involvement*. By involving production engineers, production planners and manufacturers in the design process, this stage may take more time. However, fewer design cycles are necessary, as problems in the design are detected earlier. Thus, total development time is decreased.
The difference between the traditional sequential and the DfM approach is illustrated in Fig. 2.

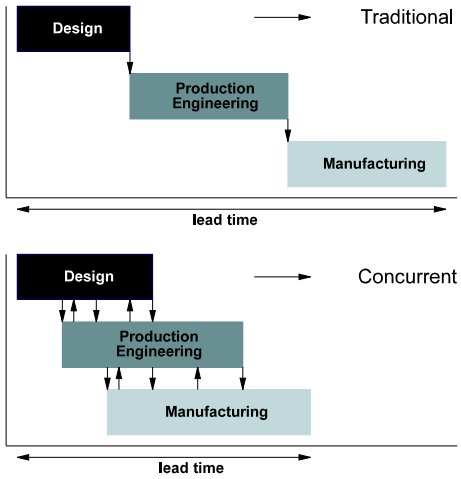

Figure 2: Traditional process and DfM-applied-process in time

As production engineering, production planning and manufacturing aspects are considered early in the development process, products are more likely to be produced in "the best possible way". To achieve this goal, often, so-called *tiger-teams* consisting of experts of all stages are used during the whole product development process.
Because process planners and manufacturers are provided with small batches of (maybe partial preliminary) information earlier, they can start with parts of their jobs earlier. In most cases this also leads to a decrease in the time-to-market.

[3] All the other characteristics to be discussed are related to this main characteristic.

As a result of the early involvement, some of the activities within DfM can be executed concurrently. When people work concurrently, coordination and control are required. This means that frequent exchange of (mostly small batches of) information is necessary, but also that the participants have to work in a systematic way.
Within DfM the different stages (and sub-stages) of the product development process overlap. Because of this reason it is difficult to point out the exact transitions between the stages. To guarantee the continuation of the product development, formal transition points are introduced. These transition points are known as milestones or as baselines, and they are often predefined by stating the required deliverables. They involve a formal evaluation of the states of the product development project.

2. SHORT DESCRIPTION OF THE CASES

By studying cases from industry we have tried to trace the characteristics of cooperation in DfM. This section shortly describes selected cases from industry and from literature. First, we introduce the case *Philips Medical Systems*. Afterwards, a case from literature about *cooperative aircraft design* is sketched. We conclude with the case of *Fokker Aircraft B.V.*[4]
More details about the selected cases can be found in TRANSCOOP deliverables DII.1 and DII.2, which can be obtained from the authors of this paper.

2.1 Case Description Philips Medical Systems

One company that deals with cooperative activities in engineering design is Philips Medical Systems [2]. Philips Medical Systems is an international oriented product division of N.V. Philips' Gloeilampenfabrieken. The headquarters of Philips Medical Systems are located in Best, the Netherlands.
Philips Medical Systems produces high-quality image handling diagnostic systems and X-ray equipment. It develops large complex non-consumer systems in small batches of 50-200 per annum. Development projects take about 2-3 years, while the life cycle of products –without development time– is usually longer than 5 years.
Besides these systems, customers are also supplied with world-wide support and educational services. Research, development, manufacturing and supply are situated in several international centra. Philips Medical Systems in general only develops and assembles products. Manufacturing is mainly subcontracted to other divisions and to external companies.
In one of the Business Groups of Philips Medical Systems, professional X-ray systems are being developed for cardio-, vascular and surgery disciplines. Such complete X-ray systems consist of several products developed by different Product Groups. The following products can be distinguished: the image detection system, image display system, table, control desk, X-ray tube with generator and stand.
Within Philips Medical Systems the development process is structured and controlled with the help of an in-house developed management method named *Systems Management*. This method is comparable with the Systems Engineering approach.
Systems Management divides the development process into seven stages in which certain activities are carried out by project teams. These stages are:

(1) Policy Study;
(2) Feasibility Study;
(3) Overall Design (or Concept Design);
(4) Detail Design;

[4]We thank Philips Medical Systems (Best, the Netherlands) and Fokker Aircraft B.V. (Amsterdam, the Netherlands) for making available the necessary information, giving us the possibility to perform our case studies.

(5) Engineering, Integration, Testing & Market Preparation;
(6) Preproduction, Market Introduction;
(7) Production, Sales, Installation & Maintenance.

Each stage is formally closed with a review, based on a report that contains all results and the (consequences of) taken design decisions. All relevant departments who participate in the development team make contributions to each report. One of the results of the review is a go/no-go decision, taken by the management.
Systems Management aims at involving all departments in each stage of the development process and to enhance communication and commitment. This results in a close cooperation between the participating designers of the different departments. As this is a DfM way of orking, this case is relevant for our purposes.
We have limited ourselves to stages 3, 4 and 5, which are represented in Fig. 3.

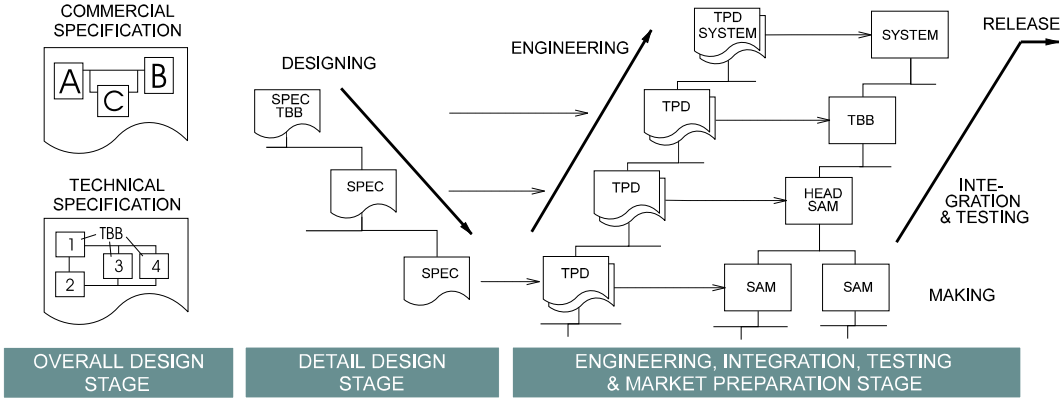

Figure 3: Stage 3,4 and 5 of Philips' Systems Management

2.2 Case Description Cooperation in Aircraft Design

Not only from industry, but also from literature interesting cases can be selected. In this section the case "Cooperation in Aircraft Design" is described [3]. Alan Bond and Richard Ricci have described how aircraft are being designed in a large organisation. In their paper they describe

- how the cooperative product development process is initiated by the specification from the customer;
- the work that is done by the different specialists who are involved in the development process, the models they use, and their interactions;
- a typical complete scheme of design goals and stages in the development of an aircraft to prototype stage;
- the kind of higher level negotiation that is taking place in aircraft design.

Aircraft design is an iterative process of refining a common product model. The product model consists of large amounts of (complex) data. Specialist models are derived from (parts of) this product model. A new version of the product model results in new derived models. The refinement of the product model can be found again in the way decisions are made. First, higher level decisions (considering both engineering decisions and project management decisions) are made. These result in lower level constraints. Only in few selected cases, higher level decisions will be renegotiated.
From the customer specifications, a high-level conceptual design is made. Then, in every iteration a group of specialists from different specialisms computes appropriate detailed models from the product model available at the beginning of the iteration.
Depending on the specialism, the output of the models or simply the design serves as an input

for discussions on improvement of the design. In these discussions, a shared vocabulary is used. The specific expertise of a specialist determines his contribution to the discussion. This signals the different kinds of roles that may appear.
By negotiating on the suggested changes, a new design, edited by the designers, will evolve. During a new iteration, other change suggestions may arise which again will be discussed. It seems to be the case that the designers are the only ones allowed to change the product model. Other specialists only are allowed to read data from the product model.
Besides on these engineering decisions, negotiation also takes place on project management decisions. These project management decisions consider topics that are related to the coordination of the cooperation. Tasks per cooperating member and the tools that are going to be used are two of these topics. Related to the second topic is the format in which data is exchanged between different specialists.

2.3 Case Description Fokker Aircraft B.V.

After the previous section, in which we sketched a case from literature about cooperative aircraft design, we describe in this section the cooperative development process of Fokker Aircraft B.V. This case description is based on several visits and reports from this company [7].
Fokker Aircraft B.V. is one of the four subcompanies of Fokker Holding B.V., which has been raised in 1993 on account of the cooperation between Fokker and Deutsche Aerospace (DASA). Fokker Aircraft B.V. designs, builds and sells civil aeroplanes and has a leading position in the market segment of aeroplanes with 65-130 seats. The most important products are the Fokker-50 and the Fokker-100 aeroplanes. Besides this, Fokker Aircraft B.V is involved in several civil and military programs and acts as a subcontractor for other aircraft developing companies.
Fokker Aircraft B.V. is divided into operational units which are responsible for the production of successively electronic systems, composites/metal laminates parts and sheet metal components, and for the subassembly and assembly of components into aeroplanes.
The central engineering department is located at Schiphol, Amsterdam, while the production departments are distributed over several locations. Fokker has subcontracted many of its manufacturing activities to suppliers. Besides that, Fokker is considering to cast off some of its operational production units.
Like Philips Medical Systems, the development of new products at Fokker is based on the Systems Engineering concept. A division into stages is used to structure the development process. At the end of each stage – at a stage transition – a baseline is presented and a review of this baseline takes place. Fokker distinguishes the stages and baselines as is shown in Fig. 4.

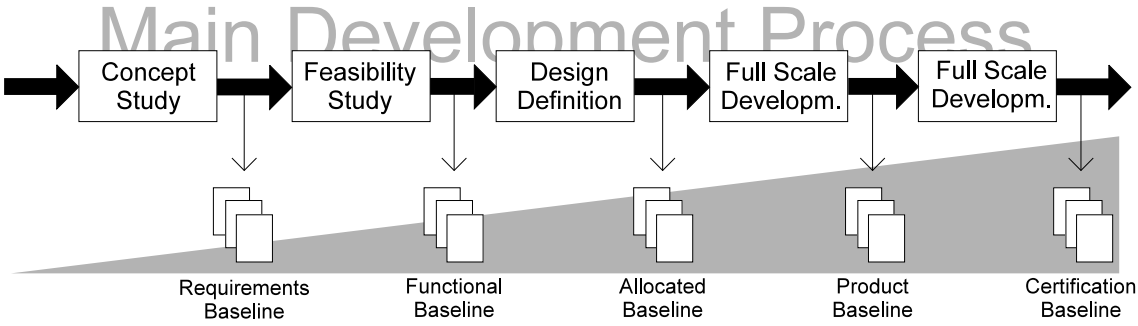

Figure 4: Systems Engineering at Fokker Aircraft B.V.

Each baseline consists of a set of documents. These sets of documents contain technical design information as well as design information related to project management.
Although some manufacturing engineering activities are performed already parallel to engineering design activities, Fokker is experimenting with the use of design build teams. Through these design build teams, formed of specialists from different disciplines, the contribution of manufacturing experts is increased in the beginning of a development project. According to Fokker this

results in a higher quality of the product and a shorter development lead time.
From the above, it is clear that the Fokker case is relevant with respect to DfM analysis.

3. DFM REQUIREMENTS FOR TRANSCOOP

As we want to develop an advanced database transaction model and an accompanying specification language to support (among others) cooperative design in general and DfM in particular, we have to know what requirements should be imposed on this model and its language. That is the reason why we have performed studies in the three aforementioned cases. In this section, we describe the requirements that came up during the study of these cases. The Philips case provided especially global information on a high level of abstraction. The Fokker case offered the most concrete input. The cooperative aircraft design case was somewhere in between.
We distinguish four different kinds of aspects: *activities*, *users*, *data* and *general* requirements. Sometimes, a requirement could be classified in more than one subsection. However, we have not done this; all requirements have only been mentioned within one category.

3.1 Requirements: activity aspects

3.1.1 Different levels of management Design scenarios occur on several levels of granularity within the information system. The highest level is related to planning design activities (cf. workflow); the lowest level is related to updating records in a database document (cf. database transactions). The most important characteristic is that scenarios on a higher level, control the ones on a lower level. E.g., on a higher level, some lower level part may be cancelled. It is not clear whether this layering can be described sufficiently by existing advanced transaction models like Open Nested Transactions [5]. On all levels, forms of co-operative transactions are needed.
Requirement:
The transaction model should support scenarios on multiple levels, where the different levels may influence each other. The specification language should be capable of specifying these different levels and the way in which they influence each other.

3.1.2 Overview of ongoing scenarios Because of the aforementioned multiple levels of scenarios, the transaction model should have the ability to overview ongoing scenarios. The scenarios on a lower level could be considered as objects on a higher level. In current practise, the high level scenarios are often implicitly implemented in the operations on objects.
Requirement:
On a higher level of scenarios, it should be possible to manipulate (e.g. view and lock) scenarios on a lower level as objects.

3.1.3 Hierarchical transactions The cases support the need for hierarchical transactions: e.g. in the Fokker case, one states the need for a design methodology that is based on a structured way of decomposing. A complex design process often has a hierarchical structure. This requirement is related to that from Section 3.1.1. However, this form of hierarchy is much more concrete and in fact, it is provided already by modern high-level transaction models [5].
Requirement:
The transaction model should support hierarchical transactions. The specification language has to allow for specifying this, both in a top-down and in a bottom-up way.

3.1.4 History of scenarios In a design environment, there is a need for knowledge about all design solutions that have been invented ever in the past. This way, the designers will not repeat work that has been done already or look for solutions in a direction that has already proven to be useless. Many scenarios in an information system that is used to support design activities, are related to design history.

Requirement:
The transaction model should be able to keep a history of transactions. Within a scenario, it should be possible to incorporate transactions that have been executed in the past. Specification and manipulation of historic information should be supported by the specification language.

3.1.5 Milestones and deadlines In all design projects, milestones and deadlines play a certain role. When deadlines are not met, this may mean that corrective actions have to be executed.
Requirement:
The transaction model has to support milestones and deadlines. Among other things, this means that it has to be possible that corrective actions are performed when a deadline is not met. More generally, one could say that description and enforcement of project phasing has to be supported by the transaction model and the specification language.

3.1.6 Timing and scheduling The requirement from Section 3.1.5 also implies the need for some mechanism of *timing*: one has to be able to specify deadlines and actions that have to be performed upon reaching a milestone. Examples of the latter are corrective actions when a deadline has not been met or making a document available to other employees, because it has been approved now. The concepts of timing and scheduling are required for other purposes too. An example in the area of authorisation is that one may want to be able to deal with a situation where a certain designer may have exclusive access to certain data during a certain period of time, after which he has to share it again with his colleagues. Another example is the situation in which some action can only be performed after some other actions have been terminated successfully.
Requirement:
This all means that the specification language should allow to describe timing aspects. The transaction model has to comply to these timing requirements.

3.2 Requirements: data aspects

3.2.1 Authorisation Within DfM, the aspect of authorisation plays an important role. Not in the first place to control team members, but especially to prevent loss of information and to guide workflow.
Certain data can only be made available after a formal approval (cf. Section 3.1.5). Note that this restricts the concurrent way of working in DfM: e.g., product support can only start after the engineering data have been made available. It depends on the concrete scenario at hand to what extent the concurrent process is limited by such constraints.
Requirement:
There will be rules stating who may access or modify which data. It should be possible to state these rules, using the specification language. The transaction model has to be such that these rules are obeyed.

3.2.2 Version management In order to support the development of design documents, the transaction model should be able to support versioning techniques.
An important aspect of version management is the release of a document. In the Philips case, a document is released by its creator. This generally happens after a project review or an internal walkthrough. The latest version of a document is stored in a generally accessible central archive. Of course, one may also imagine a completely different scenario, where some central daemon releases the documents and is responsible for version management. Releasing documents is strongly related to consistency matters.
Requirement:
The TransCoop *framework should be as flexible as possible, in order to be able to deal with as*

many different versioning mechanisms and version management scenarios as possible. It should at least support versioning for complex data structures where there are complex relationships between the documents to which versioning is applied.

3.2.3 Complex data structures and relationships Complexity of the data structures is found both in the structure of the information systems and in the data stored in these systems. In the Fokker case, it has been observed that 3D information plays an essential role in the design process: the design process is intrinsically a three-dimensional one. This means that complex datastructures with associated primitive operations are needed for storing and manipulating this information.
Another important observation is the following statement that has been made in one of the Fokker documents: "The *kind* of data is not influenced by the organisation or the project structure [...]; in other words: when the same plane is developed within a total different organisation or project structure, still the same kind of design data will be relevant." Fokker proposes to use this stable factor to develop generic design support tools.
Requirement:
From the above, it follows generally that the transaction model should support complex operations. Exchange of data between different data structures has to be supported. This is necessary to be able to deal with views, which is also a requirement (cf. Section 3.2.5). E.g., in the Fokker case we saw the transformation from 3D into 2D information and vice versa.

3.2.4 Shared data structures Often, information is distributed physically over an organisation. Especially for large DfM projects, geographical distribution is inevitable. By coupling all available data directly (all data available to everyone) or indirectly (only title and global contents available to everyone) via a network, a database is accomplished that can be accessed from different locations and by different disciplines as one logical engineering database.
Requirement:
The transaction model has to be such that the distributed data remain consistent. In an ideal situation, everyone can do whatever he wants (provided he has authorisation to do so) without interference from other designers. This ideal situation will not be feasible, but we should try to do the best we can. The specification language has to be suited for describing all different kinds of distributing and making the data available. This requirement seems to be general for all situations where there is common information to be managed.

3.2.5 Relationships between data As is stated in a document from Fokker: "[...] in many places and in different stages, data are recorded that are related to each other via reuse and iteration. These relationships 'cross' all design stages." An example is the generation of maintenance and instruction manuals from the product information. Another example is the (automatic) generation of NC-machine control data from the design information. These NC-machines can be used to make products, but also to generate production tools.
Moreover, it has to be possible to combine different data into one engineering database and extract certain data from this engineering database again. In all cases, also the need of the designers emerged to use each others information and to be able to cope with relationships between data of several disciplines.
It has been mentioned in the previous section that the information about the design is recorded in several kinds of documents, giving different views on the artifact that is being designed.
Finally, in the context of quality insurance, facilities are needed to "connect" measured values about really produced goods to the original design, in order to determine the need for changes, improvements, etc.
Requirement:
It should be possible to specify different views and the relationships between them, and provide mechanisms to compute and manipulate these views and their relationships. The specification language should allow defining interfaces (between subsystems, but also between geometrical

constructs like two connected compartments) in a straightforward way and the transaction model should follow these rules.

3.2.6 Retrieval of information It occurs often that knowledge or experiences cannot be traced in an organisation. This causes this information to be regenerated, although it is already available somewhere. Moreover, when designers can use each others information in an unambiguous way, the design process will be as concurrent as possible.
The need for high-level queries occurs often during the design process. An example[5] of such a high-level query is the following: "Show all locations where the toilets could be placed in the plane".
Requirement:
A suitable information retrieval mechanism is essential in a tool supporting DfM. For dealing with the aforementioned high-level queries, often a lot of (different) information is needed. The transaction model should be such that these high-level queries can be answered (correctly!) within a reasonable amount of time. This seems a general requirement, though.

3.2.7 Constraints Design information is often recorded in terms of constraints. Sometimes, this usage of the term constraint does not comply to the way it is used in computer science. E.g., some people would model the colour of a car by means of a constraint that states that the car should have a certain colour. Constraints play an important role in a design process.
There are different kinds of constraints. Some constraints simply may never be violated (like "the plane has to be such that it does not crash"). This may seem obvious, but there is another class of constraints: the *deontic* constraints. For example: "the distance between two passenger seats should be at least x meter". We want deontic constraints to be satisfied, but their violation does not cause disasters to happen. You could also call such constraints non-fatal.
For the checking procedures, a mechanism for constraint validation is needed: does the current state of the system satisfy the constraints? Finally, a mechanism is needed to recognise inconsistencies and redundancies. Although we know that decidability problems may occur here, it is clear that the more help a system offers for consistency and redundancy checking, the more useful that system is.
Requirement:
Within the TRANSCOOP *framework, we should be able to specify the database constraints. These could be part of scenarios used in formal sign-off procedures.*
We should support both the distinction between fatal and deontic constraints and the specification and handling of corrective actions.

3.3 Requirements: user aspects

3.3.1 Single user aspects In a realistic DfM situation, there will be a number of different people involved. These people will be organised in several project groups. Some of these groups may have an *ad hoc* nature, whereas others may be formally established ones. E.g., in the Philips situation, group membership is determined by the project leader. A group has permanent members, like the project leader and the quality assurance engineer. There are also temporary members, like specialists from manufacturing or purchasing. A similar remark can be made for the relationships between the users: some relationships will be *ad hoc* and temporary, whereas others may be of a more permanent nature. It is hardly possible to make general remarks about this: each organisation will organise its design process in a (more or less) different way.
Requirement:
The best advise we can give to the designers of the transaction model (and its language) is to try to be as flexible as possible. Try to support as many kinds of user scenarios as feasible!

[5] Although this example is in natural language, we do not intend to support natural language queries.

3.3.2 User groups Generally, there are groups that belong to the structure of the organisation, but not all of them do. Some groups are homogeneous, others may be heterogeneous. There are formal (in most cases permanent) groups, but also informal ones, that generally exist only for a limited period of time. The temporary groups disappear either when they have solved the problem they should, or when a supervisory group or person declares that the work will be stopped. In the cooperative aircraft design case, we observed that the composition of the group of interacting specialists may change because of the specific knowledge of certain specialists.
In general, the task a group has to perform justifies existence of the group. Generally, this implies certain authorisations and duties for the group members.
Generally, design teams are heterogeneous. Group members may have different roles. In the specification language, it should be possible to distinguish different roles. However, this will mainly boil down to different authorisation for different people.
Requirement:
The same remark that has been made in the previous subsection, also holds here. The DfM application area is not specific enough to be able to make deterministic statements about aspects of user groups. Within one particular organisation, some aspects will be very important, whereas in another organisation, the situation may be completely different. We will have to be as flexible as possible to this respect.

3.4 General requirements

3.4.1 Flexibility A system supporting DfM has to be flexible enough to support changing requirements during the project trajectory. One aspect of this is the aforementioned consistency checking of constraints (as the system has to control the consistency of an evolving set of constraints), but there is more to it.
This aspect is one of the most important characteristics of DfM. For DfM scenarios generally take a long period of time and involve a lot of people. It often occurs *during* a DfM project that changes have to be pursued. This may vary from other people being involved in the same activities to a complete redefinition of the most important starting points in the design process. This may for example result in the need for different kinds of `commit` statements in the transaction model, where some committed action is only "really" final when some form of approval has been issued. A related issue is that of information evolution, where the committed data have to remain useful even after a schema evolution.
Requirement:
The specification language and the transaction model have to be designed with this urge for flexibility in mind, although we can not assess at this moment what exactly is needed.

3.4.2 Heterogeneous systems In DfM, heterogeneous systems occur: it may be the case that different designers use different systems under different operating systems. This is not a specific characteristic of a DfM scenario. It may occur in any situation where many people from different departments or organisations have to cooperate.
Another remark is that the environment of a design situation often imposes requirements on the information that is produced by the organisation or constrains the information that is provided to the organisation. E.g., Fokker has to provide certain documents in a certain form to governmental safety commissions. Another example is the fixed format in which subcontractors deliver their designs.
Requirement:
The specification language should allow the description of heterogeneous systems. Also, a facility would be useful that enables one to describe "black-box" situations. Often, we only want to say that something is happening, without already specifying it. It is important to be able to specify interfaces between (parts of) the organisation and external parties.

4. CONCLUSION

Cooperative applications like Design for Manufacturing cannot be supported adequately by traditional database transaction model. The main reason for this is that these model guarantee database correctness by giving the users the idea that they can work with the database in isolation. In a really cooperative environment, this is not the case. Therefore, advanced transaction models are required.
In this paper, we have reported our research for requirements the DfM application area imposes on such an advanced database transaction model and an accompanying specification language. We have done this by performing two case studies (Philips Medical Systems and Fokker Aircraft) and one case from literature (about cooperative aircraft design).
Like the design trajectory in DfM, coming up with requirements for such a transaction model is also design process that takes a lot of time and in which a number of people are involved. Therefore, one cannot expect that the presented requirements are completely "ready", meaning that they have been refined sufficiently and they will never change. When defining the transaction model and the specification language, additional requirements, changes and modifications will come up. However, our analysis definitely forms a good and structured start for this work.
Our work has been performed in the context of the TransCoop project. However, reserach projects that aim at similar computer support for cooperative applications may also bnefit from the results we have accomplished.

REFERENCES

[1] F.A. Andriesse: "Topmanagement Must Break Down Walls to Come to a Better Innovation Process", *De Ingenieur*, Vol. 105, December, 1993 (in Dutch).

[2] L.T.M. Blessing: *A Process Based Approach to Computer Supported Engineering Design*, Ph.D. thesis, University of Twente, 1994.

[3] A.H. Bond and R.J. Ricci: "Cooperation in Aircraft Design", *Research in Engineering Design*, Vol. 4, pp. 115–130, Springer-Verlag, 1992.

[4] K.J. Cleetus and R. Reddy: "Concurrent Engineering Transactions", *CE & CALS '92 Washington Conference & Exposition*, June 1992.

[5] A. K. Elmagarmid (ed.): *Database Transaction Models for Advanced Applications*, Morgan Kaufmann Publishers, 1992.

[6] Thomas Tesch and Peter Verkoulen (eds.): "Requirements for the TransCoop Transaction Model", Report TC/REP/GMD/D2-2/207, January 1995.

[7] H. Timmerman: "IDEE project Baseline 2", Report CC-0417, Fokker Aircraft B.V., May 1994 (in Dutch).

[8] Peter Verkoulen and Thomas Tesch (eds.): "Requirements for the TransCoop Specification Language", Report TC/REP/UT/D2-1/014, January 1995.