7th Workshop on Product Structuring – Product Platform Development
Chalmers University of Technology, Göteborg, March 24-25, 2004

MODELLING ON MULTIPLE ABSTRACTION LEVELS

Kaj A. Jørgensen

Abstract

Modelling is becoming more and more important as a design approach caused by the increasing availability of computer-based modelling tools. With this approach, a model is created in order to capture as many design decisions as possible. Hence, it is assumed that the model is manipulated through the design processes and end up as a detailed model.

To think of modelling as a process, which is extending the model with more and more details is very typically and is the primary view derived from the predominant type of modelling tools, the CAD software, which focus on geometry. It is, however, very important to state that design is also performed on different levels of abstraction. This is to various extend covered by most design theories but not supported by many modelling tools. Furthermore, such approaches are very often imprecise and loosely developed.

In this paper, modelling on multiple abstraction levels is outlined in combination with the abstraction mechanisms classification and composition and with object-oriented analysis and design. Based on the theory of general systems, some possible abstraction levels are identified. Multiple abstraction levels can be utilised in connection with both analytic and synthetic modelling. Hence, they can be applied to both requirement definition and design by modelling. The modelling approach is related to product modelling, product family modelling and building modelling. In addition, the differences between modelling towards lower levels of abstraction and towards higher degree of detail are addressed

1 Models and Modelling

Modelling is a very important approach in many design projects where the designed artefact is very complex, i.e. with a large number of properties, with a large number of components and/or with a complicated structure. The fact that computer-based modelling tools have been developed to be more useful and with an increasing number of functionalities make modelling even more important. Often, the modelling tools dictate certain modelling methodologies with a number of limitations. However, modelling can be performed in many ways and can have different meanings to designers. The emphasis can put on many subjects, decisions can be sequenced in many ways and resources can be allocated variously.

Methodologies for system development are often based on concepts derived from General Systems Theory. According to this theory, a *system model* is an intentionally simplified description of a system, fulfilling a certain purpose. Hence, the simplifications imply that some choices are made in order to select the most important properties, components and

relationships. Thus, a system model can e.g. be suitable for communication between designers, because with the model, it will be possible to concentrate on the most important aspects of the system.

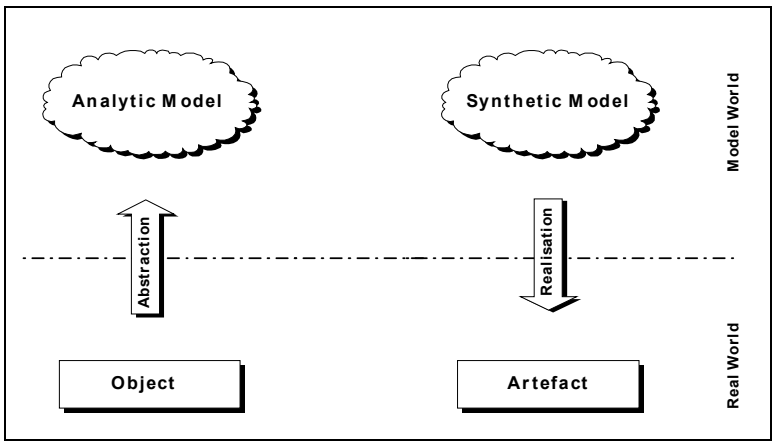

Figure 1 - Analytic and synthetic modelling

Very often, models are considered *analytic models*, i.e. models of something existing, often physical. Such a model serves as a description, an abstraction, where a deliberate simplification is made, i.e. a selected set of properties, components and structures.

In contrast to analytic models, *synthetic models* are not built from anything existing but instead, a synthetic model is created as a foundation for construction of something physical – an artefact. Hence, synthetic models are built purely from ideas, thoughts and imaginations and the result is obtained in some kind of representation. The two modelling approaches are illustrated by figure 1. Consequently, design by modelling is a development approach, where a synthetic model is designed as an intermediate result and the final result is an implementation of the model in the real world.

The most typical reason for synthetic modelling is to be able to manipulate and test the model before the actual physical system is built. Modelling makes it possible to ensure that the design is correct and by various presentations of the model at different stages, it is possible to see the consequences of decisions and to reach a good impression of the final result. When synthetic modelling is performed, it is often important to view the model from many different aspects and to represent the model on many different abstraction levels. This is especially necessary at the beginning of the modelling process before any decisions are made about e.g. form and location.

2 Fundamental Issues of Modelling

An important fundamental issue of modelling is *abstraction mechanisms*, which provide the means for identification and design of invariant components and structures ([Smith 1977a], [Smith 1977b], [Rosch 1978] and [Sowa 1984]). Two abstraction mechanisms are defined here: *composition* and *classification.* In essence, composition focuses on the components while classification focuses on attributes. Together, they cover modelling of component *types* as the basis for component *instances* of the model and they provide the means to set particular focus on the most invariant decisions. A classification process results in a basic hierarchy of types and a composition process results in a basic hierarchy of components. Further, each of

the abstraction mechanisms is complemented by two underlying mechanisms: in classification: *generalisation* versus *specialisation* and in composition: *aggregation* versus *separation*.

Another important issue of modelling is the *object-oriented paradigm,* which can be adopted in harmony with the abstraction mechanisms. In this paradigm, each model component is regarded as a living organism, which act and interact with other components. Thus, object-oriented components are equipped with behavioural attributes, which enable them to respond to requests and, consequently, even if a real world component is non-living, the corresponding model is created as an active component.

The composition abstraction mechanism is used to identify possible internal components and structure. Composition includes also the opposite, to identify components, which are assemblies of other components. In principle, separation can be continued to any detail and aggregation can be continued to include everything. For a given component, the hierarchical structure of all its sub-components is the *composition hierarchy*; see figure 2. As indicated, such a structure will always be a *tree structure*, and the terms *top-down* and *bottom-up* are clearly defined according to the structure. The composition hierarchy gives an excellent overview of the selected components. However, it is important to underline that, in a composition hierarchy, all other cross-going relationships are omitted.

Example 1:	Example 2:
Building + B1 – B2 Building section + BS1 – BS2 Floor + F1 – F2 Room + R1 – R2 Furniture ...	**Floor** + F1 – F2 Building element + BE1 – BE2 Building element + BE21 – BE22 Building element + BE221 – BE222 ...

Figure 2 - Sample composition hierarchies

The classification abstraction mechanism is used to identify types of components on the basis of the attributes, they include. The types are organised in a *taxonomy*, where the most general types are placed at the higher levels and the most special types are placed at the lower levels; see figure 3. According to the object-oriented paradigm, attributes of super-types are *inherited* to sub-types. This means that each type adds to its own list of attributes the attributes of all its ancestor-types. So for each component type, it is only necessary to define and specify the attributes, which specifically distinguish this type from its super-types; all other attributes are inherited. One has to bear in mind that, for a given set of components, different classifications can be performed dependant of the criterion that is selected. Thus, in order to carry out a classification, such a criterion has to be identified.

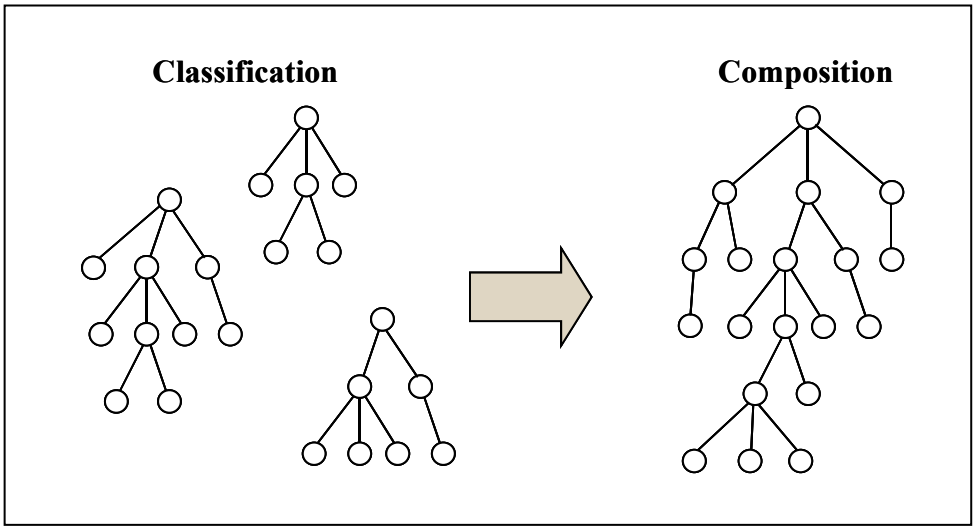

Figure 3 - Classification and composition hierarchies

3 Data Models and Information Models

Computer-based models are fundamentally stored in computers as *data objects* and *data structures*, which can be manipulated by applications. Therefore, development of tools for modelling includes both a development of a *data model* and *applications*, which can operate on the data model. One of the most important requirements for the data model is that it is non-redundant so that no data value is store more than once. In order to ensure that this requirement is fulfilled, the model representation has to be considered very carefully based on the meaning of data, the semantics. Therefore, the foundation for a data model is an *information model*, created in combination with semantics from the domain, which the design model is addressing.

Most often, each modelling tool has its own proprietary data model and file format, incompatible with each other. This means that models can only be exchanged between tools with loss of various amounts of data. Therefore, international organisations like ISO have worked on development of general acceptable representations of a large variety of products. ISO has been developing the "STandard for Exchange of Product model data" (STEP / ISO 10303), covering a great variety of areas for product modelling. With the corresponding file format, it is possible to build applications, which can exchange models without loss of data. Similarly, the International Alliance for Interoperability (IAI) has been developing the Industrial Foundation Classes (IFC / ISO PAS 16739) for representation and exchange of building models.

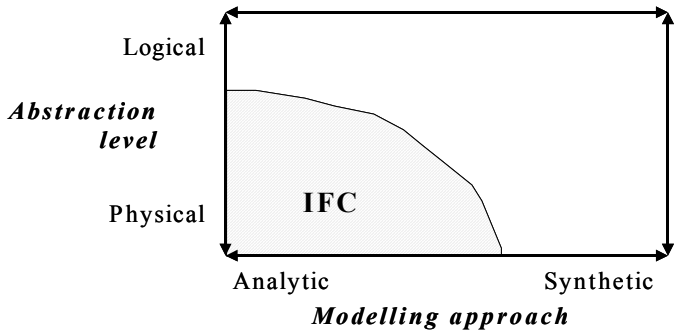

Figure 4 - Characterisation of IFC

It is characteristic for both standards that they are rather analytic oriented and they are primarily developed to represent products/buildings from a physical point of view. The components and structures of the model are primarily exact representations of the corresponding physical components. Clearly, this is indicated by the fact that geometry data are the primary data necessary in order to create the model components. These characteristics do not mean that they cannot be used to represent models, which are created as the result of a synthetic approach. But because synthetic modelling can be very different from analytic modelling, there are many aspects of this approach, which cannot be modelled or can only be modelled with difficulties. One aspect is modelling on different abstraction levels.

4 Generic Information Model Component

As stated, information models ([Hammer 1978]) are basis for data models and, subsequently, computer models are created from data models, i.e. data objects and data structures are created as instances of the defined data object types and data structure types of the data model.

In order to be able to create all sorts of models and to perform many different modelling processes, a conception of a *generic information model component* is proposed. This component is inspired from general systems theory and from object-oriented modelling. Based on this model component, a number of basic modelling aspects will be described.

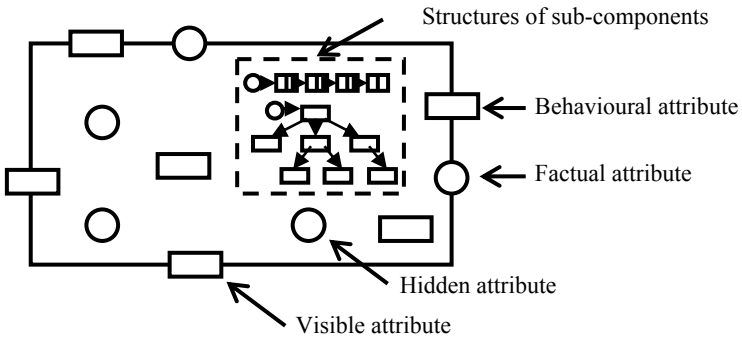

Figure 5 - Generic information model component

The generic component consists of a set of *attributes* and a *structure of sub-components*; see figure 5. Some attributes are *factual attributes*, defining the state of the component, and some attributes are *behavioural attributes*, defining the operations, which the component can carry out. An alternative division of attributes defines some attributes as *visible attributes*, which can be called from other components, and some are defined as *hidden attributes*. The structure establishes the *relationships* between the component and other components internally as well as externally. All sub-components are regarded the same way, recursively. With this generic component, it is obvious that it is possible to address the following important issues of a top-down system modelling: purpose, function (visible behavioural attributes), form (visible factual attributes), internal (hidden) attributes and internal structure.

All structures can be represented by two kinds of relationships in the information model: *references* and *collections*. A reference is a special attribute of which the value holds a direct link from one component to another. The main purpose of references is to create structures, which avoid *redundancy.* References are very simple and easily understood instruments for solving this because, the need for having multiple copies of data values in different components can be eliminated.

A collection is also defined as a directed relationship between components. The component, which holds the collection, is the *anchor* component and the components included in the collection are the *body* components. Generally, each collection can consist of zero, one or more body components but, for individual collections, specific constrains can exist. From each anchor component, the members of the body can be accessed as a whole, or individually. Hence, each collection defines an access path from the anchor to the body components and, to make such access paths as efficient as possible, information structures can be defined. When such structures are implemented, it is important to state that collections are reduced to structures, which consist of only references.

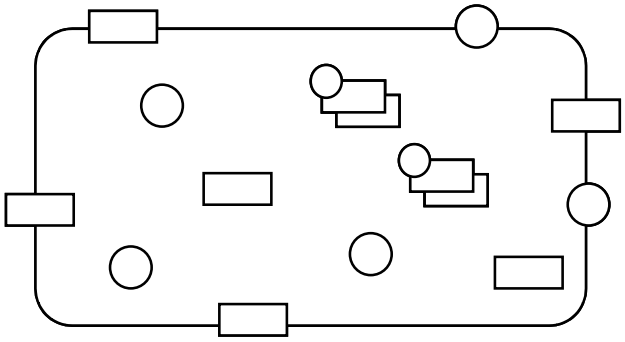

Figure 6 - Generic information model component type

The specifications of components are made in *component types*, which are the primary content of information models; shown in figure 6 as a rounded box. From each type, an indefinite number of components, instances, can be generated; see figure 7. First of all, each type includes a specification of a set of attributes with *name*, *data type* and perhaps *constraints* about what values can be assigned to the attributes of the instantiated components. Based on attributes, the component types can be classified according to the classification abstraction mechanism. The result of classification is a *taxonomy* of the identified types. In the information model, all components must be specified by a component type, even if some types may only have single instances.

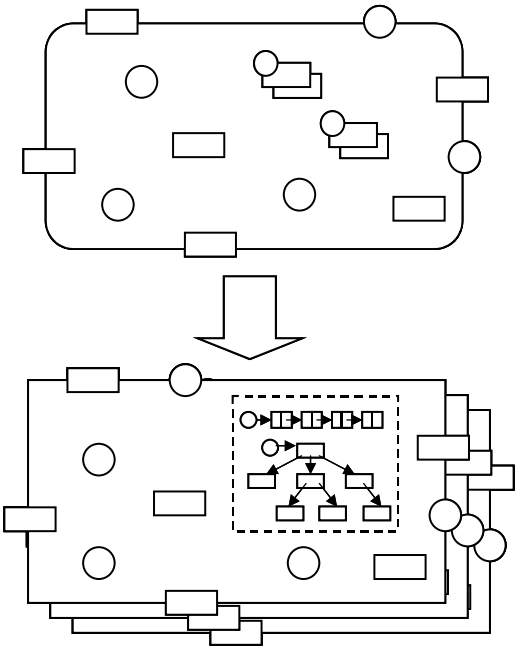

Figure 7 - Generation of instances from a type

Identification and specification of structures can also be included in the component types by creating the *relationship types*: *reference types* and *collection types,* indicated in figure 6. It is important to underline that, because references are attributes, they are also defined by means of the classification abstraction mechanism. Similarly, each collection type can be defined by a special attribute representing the anchor. Therefore, such attributes can also be incorporated in the classification process. However, when further specifications have to be added to relationship types, special component types have to be added to the information model in order to represent the implementation of the collection type. The specifications include semantics like uniqueness, integrity, cardinality, etc.

When an information model is transformed to a data model, for instance a relational data model, the information model types are mapped to data object types and data structure types. Depending of the implementation, the data model can obtain some of the semantics of the information model but, usually, the remaining semantics has to be included repeatedly in applications, which operate on the data. When the applications are developed on the basis of a data model, the data object types are used as templates to cast the data objects and the data structure types are used to create data objects to hold the data structures.

5 Information Models Development

When a new information model is created, the information model types are identified and defined by classification with a taxonomy as the result. In the taxonomy, reference types and collection types are added in order to represent the necessary compositions. Further, the semantics is included.

As stated, information models can cover any domain; so in order to create a new information model, the *purpose* of the model must be defined and the selected domain must be analysed and delimited. As stated above, classification of a set of types can be performed in many ways, so a *classification criterion* must be defined also.

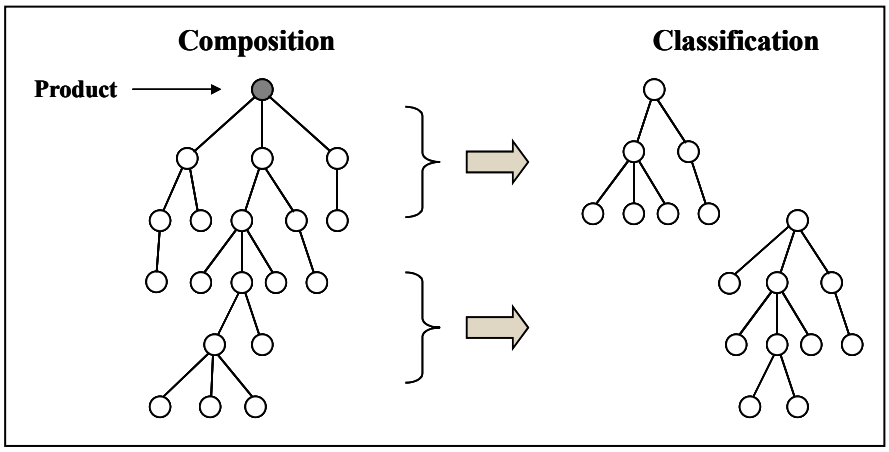

Figure 8 - Classification on different levels of composition

Very often, information models are aimed at certain levels of a composition hierarchy. This is illustrated by the STEP and IFC standards, which covers modelling of product/building parts (elements) and their assemblies. As stated, these information models indicate that a relatively analytic approach has been followed. The top of a composition hierarchy is the product (see figure 8) and, underneath, the product may consist of modules, which, of cause, may also be modelled in information models. Such modules are often defined in models of product families ([Jørgensen 2003]) and these models illustrate that a product model is raised to a higher abstraction level; see figure 9.

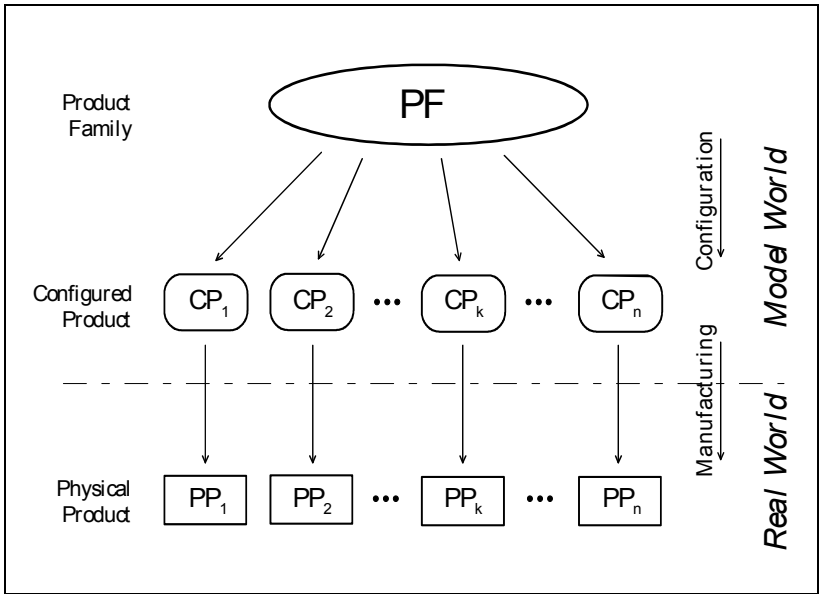

Figure 9 - Product family model

First of all, a product or a product family can be seen in a wider context, if the system (target system), where the product is going to function, is also taken into account. This view can be related to the composition hierarchy by adding further levels on top of it; see figure 10.

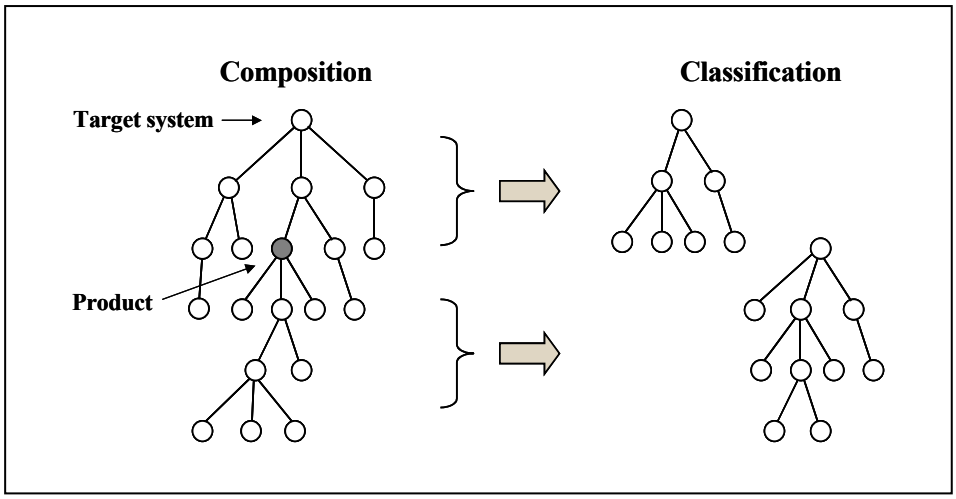

Figure 10 - Classification of products seen from the target system level

At any level in the composition hierarchy, a broader view of synthetic product/building models can be addressed. Many different issues can be modelled and multiple abstraction levels can be identified. In any view, modelling must of cause include considerations about structure as well as attributes of the products/modules/components, i.e. visible as well as hidden attributes and factual as well as behavioural attributes. However, focus on different attributes in a specific order is also essential for modelling on multiple abstraction levels. In order to address the ability to perform functions, the emphasis should be put on the visible behavioural attributes as the top level of abstraction; see figure 11. Next, the visible factual attributes are typical the most interesting. Based on these specifications, the hidden attributes and the structure would be specified until a complete specification is carried out, as illustrated by figure 5.

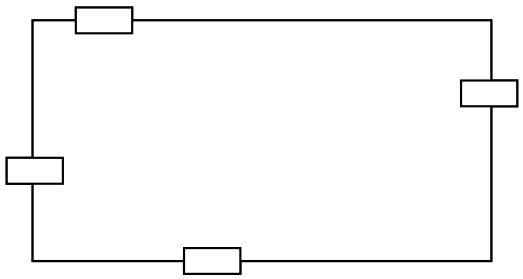

Figure 11 - Focus on the behavioural attributes at the top level of abstraction

6 Conclusion

In general, the following top-down approach of synthetic modelling should be followed regarding levels of detail: *purpose*, *function* (how they take part in the surrounding processes), *form* (visible factual and behavioural attributes), *content* (internal hidden attributes) and *structure* (internal). When performing modelling of information, e.g. for products or product families, each type of component can handled this way. In addition, it is important also to carry out classification of the identified types. Based on such information models it will be possible to generate data objects and data structures of a computer-based model.

References

[Hammer 1978]
Michael Hammer and Dennis McLeod: *The Semantic Data Model: A Modelling Mechanism for Data Base Applications.* Proceedings of ACM/SIGMOD International Conference on Management of Data. Austin Texas, pp.144-156, 1978.

[Jørgensen 2003]
Kaj A. Jørgensen: *Information Models Representing Product Families*. Proceedings of 6th Workshop on Product Structuring, 23rd and 24th January 2003, Technical University of Denmark, Dept. of Mechanical Engineering.

[Rosch 1978]
Eleanor Rosch: *Principles of Categorisation.* In: Cognition and Categorization. Laurence Erlbaum, Hillsdale, Jew Jersey, 1978.

[Smith 1977a]
J. M. Smith, D. C. P. Smith: *Database Abstractions: Aggregation.* Communications of the ACM vol.20, no.6. pp.405-413 New York 1977.

[Smith 1977b]
J. M. Smith, D. C. P. Smith: *Database Abstractions: Aggregation and Generalization.* ACM transactions on Data Base Systems, vol.2, no.2. pp.105-133 New York 1977.

[Sowa 1984]
John F. Sowa: *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, 1984.

Kaj A. Jørgensen
Associate Professor

Aalborg University, Department of Production
Fibigerstræde 16
DK-9220 Aalborg Øst, Denmark.
Phone: +45 9635 8945. Fax: +45 9815 3030.
E-mail: kaj@iprod.auc.dk
Web: http://www.iprod.auc.dk/~kaj