Conceptual Modelling and Related Methods and Tools for Computer-Aided Design of Information Systems

Bo Sundgren

SCB

R&D Report
Statistics Sweden
Research - Methods - Development
1990:12

INLEDNING

TILL

R & D report : research, methods, development / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1988-2004. – Nr. 1988:1-2004:2.
Häri ingår Abstracts : sammanfattningar av metodrapporter från SCB med egen numrering.

Föregångare:

Metodinformation : preliminär rapport från Statistiska centralbyrån. – Stockholm : Statistiska centralbyrån. – 1984-1986. – Nr 1984:1-1986:8.

U/ADB / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1986-1987. – Nr E24-E26

R & D report : research, methods, development, U/STM / Statistics Sweden. – Stockholm : Statistiska centralbyrån, 1987. – Nr 29-41.

Efterföljare:

Research and development : methodology reports from Statistics Sweden. – Stockholm : Statistiska centralbyrån. – 2006-. – Nr 2006:1-.

R & D Report 1990:12. Conceptual modelling and related methods and tools for computer-aided design of information systems / Bo Sundgren.
Digitaliserad av Statistiska centralbyrån (SCB) 2016.

urn:nbn:se:scb-1990-X101OP9012

Conceptual Modelling and Related Methods and Tools for Computer-Aided Design of Information Systems

Bo Sundgren

R&D Report
Statistics Sweden
Research - Methods - Development
1990:12

Från trycket	Oktober 1990
Producent	Statistiska centralbyrån, utvecklingsavdelningen
Ansvarig utgivare	Åke Lönnqvist
Förfrågningar	Bo Sundgren, tel. 08-783 41 48

ISSN 0283-8680
Printed in Sweden

CONCEPTUAL MODELLING AND RELATED METHODS AND TOOLS FOR COMPUTER-AIDED DESIGN OF INFORMATION SYSTEMS

Bo Sundgren
Statistics Sweden[1] and
Stockholm School of Economics[2]

Paper for the Second International Conference on Information Systems Developers Workbench, Gdansk, Poland, September 25 - 28, 1990.

Abstract. Conceptual modelling is a good starting-point for design and construction of information systems as well as for computer-based tools supporting such work. The paper discusses how conceptual modelling can be used and extended to become a comprehensive and complete methodology for specification of external, user-relevant properties of an information system and for determination of internal system properties satisfying the specified external requirements. The paper claims that methodologies and computerized tools for information systems development are not only mutually dependent upon each other but must also be tailored to the particular needs and conceptual frameworks of the organizations and applications where they appear. This thesis is illustrated by the author's experience from statistical applications and organizations. As a consequence CASE tool should be open, concept-driven "shells" or "toolkits" rather than closed generalized software products.

1 Information systems design methodologies and computer-based design tools

A starting-point for this paper is that any development of computer-based design tools (like so-called CASE tools, and components of such tools) must be firmly based on information systems theory and information systems design methodology.

---

1) Statistics Sweden, S-115 81 STOCKHOLM, Sweden
2) Stockholm School of Economics, Box 6501, S-113 83 STOCKHOLM, Sweden

Information systems are often large and complex systems. General systems theory and information systems theory (cf Langefors [1]) implies certain principles for the design of such systems. One principle is that complex systems are imperceptible as a whole, and must be broken down into simpler subsystems. Another principle is that the internal properties of a system to be designed should be derived from a specification of the desirable external properties of the system.

Thus information systems theory implies that an information system design methodology must be based on

(1) an **information systems architecture** that partitions any imperceptible information system into perceptible subsystems (parts, functions);

(2) a **specfication methodology** for stating the external properties of (a subsystem of) an information system in such a way that the internal properties of the (sub)system can be derived in a systematical way from the external properties.

In this paper I will discuss an information systems architecture that is based on a general definition of an information system, and a specification methodology that is linked to this architecture and is inspired by the methodology of conceptual modelling.

2 A general architecture of information systems

According to information systems theory [1] an **information system** is a system for

- collection,
- storage,
- processing, and
- presentation (retrieval, distribution)

of information.

An information system is an abstract entity. The concrete realization of an information system is a data processing system. Thus a data processing system is a system for

- collection,
- storage,
- processing, and
- presentation (retrieval, distribution)

of data.

With a slightly modified terminology, which is more adapted to data base orientation (and thus to conceptual modelling, which has its roots in data base theory) we may say that an information system (and a data processing

system) consists of four major subsystems, the subsystems for

- input oriented processes,
- output oriented processes,
- transformation processes, and
- information base (database) processes.

Note. We shall follow the practice of everyday language and let the term "database" denote also what we have called above "information base".

The functions of the four subsystems may be described in the following way:

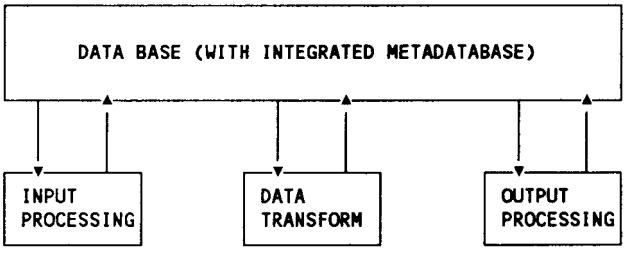
(1) the **database** contains information (represented by data) about the status and development in a selected subsystem of the real world, the **object system**;

(2) the **input-oriented subsystem** makes sure that the database properly reflects the status and development in the object system;

(3) the **output-oriented subsystem** makes the contents of the database available to the users of the information system;

(4) the **transformation subsystem** transforms information in the database in accordance with requests from the other subsystems.

The concepts involved in the above-mentioned definition of an information system may be structured and visualized in the following way:



This conceptualization of information systems and data processing systems may be described as **database oriented**, since the database subsystem is central in several respects:

- the database reflects the object system;
- all operations in all subsystems take all non-initial inputs from the database, and deliver all non-terminal outputs to the database;
- the database is also the central holding of **meta-information** (represented by **metadata**) in the information system; this "database within the database" is called the **metadatabase.**

3 Formal specifications of information systems

3.1 The needs for formal specifications and metainformation

In the previous section we defined a **general** structure for information systems. When we design, operate, and maintain a **particular** information system, there are several needs that motivate the development and maintenance of a formal specification of that particular information system, the **application system**, and its parts. In addition to serving as a basis for computer-aided design, the formal specification should also facilitate

- efficient communication between different categories of people, who are involved in the design, construction, operation, maintenance, and use of the information system;
- reliable documentation of the information system;
- efficient communication between the users and the information system during the operation of the information system.

Thus the formal specification should serve as a common **frame of reference** for different categories of people and as an **interface** between the users and the information system. Furthermore, the formal specification should also serve important internal needs for metadata of the information system itself. For example, most software components of the system will need formal descriptions of the files and records in the database, as well as descriptions of the mappings between the internal data representations and the external views of the data, as seen by the users.

3.2 Conceptual modelling and concept-driven tools and systems

There are different methodologies for formally specifying information systems. Especially in connection with directive information systems [12], such as statistical information systems, it seems suitable to start from a conceptual model of the object system, to be reflected by the database part of the information system. This approach is called 'conceptual modelling'. CASE tools and other computer-based systems for supporting the process of designing and constructing information systems (and their data processing representations), which are consistently based upon conceptual modelling as the method for specifying the (external) requirements, may be called **concept-driven** tools and systems.

A conceptual model can be used to fullfil many of the tasks defined above for a formal specification of an information system. It can serve as the basis for communication between users and designers of information systems. It can also serve as an interface between the contents-oriented and the technically oriented parts of an information system design and construction process. And it can be the model in terms of which the user communicates with the computerized information system.

What then is a conceptual model? The term emanates from database theory. For example, in the ANSI/SPARC [10] three-level schema architecure for databases, the conceptual schema has the function of being a relatively invariant specification of (the contents of) the database, existing as an intermediary level in the mappings between the external schemas, reflecting dynamically changing user needs, and the internal schema, reflecting the technology-dependent, dynamically optimized hardware/software implementation of the database. One way of attaining the desirable invariance in the conceptual schema is to base it on a model of the real world, rather than on a specification of the ever changing information needs and/or data representations.

Other terms that have approximately the same meaning as 'conceptual model' are 'infological model' and 'semantical data model'. The term 'data model' is sometimes used as a synonym to 'conceptual model', but a data model is often assumed to be an abstraction of the physically existing database, more than a model of the real world represented by the database.

Details about one particular approach to conceptual modelling, the Object-Property-Relationship (OPR) approach, can be found in appendix 1, as well as in [2], [12], [13], and [21].

3.3 Modelling the components of an information system

A complete conceptual model of an information system (data processing system) should contain subsets (views) covering the four major components of the system:

- the input oriented view,
- the output oriented view,
- the transformation view, and
- the database view.

One may argue that one or other of these views is redundant, that is, logically superfluous, since it may be derived from the others. For example, it must be possible, in principle, to derive the transformation view from the input oriented view and the output oriented view. However, it is often practical, and even theoretically advantageous, to treat all four views of the complete conceptual model explicitly. Thus, for example, one argument for modelling the transformation view explicitly is that sometimes the easiest and clearest way to define an information output is to describe how it is derived by successive transformations from information inputs.

If one of the four views of the complete conceptual model should be regarded as more basic than the others, it must be the database view. Actually this is something that distinguishes conceptual modelling from other approaches to specifying the external requirements on an information system. As was stated earlier in this paper, a conceptual model is basically a model of the real world that is reflected in the information system.

We shall sometimes call the database view the **base version** of a conceptual model.

3.4 Conceptual algebras

The idea of "conceptual algebras" is an interesting possibility for facilitating consise and precise specifications of conceptual models, views of conceptual models, and - not least - the relationships between the views, and between the views and the complete conceptual model.

A conceptual algebra, like any other algebra, consists of two major components: a set of entities, and a set of operators operating on the entities, thus producing other entities within the same set of entities. In mathematics the entities are typically numbers, and the operators are, for example, "addition", "subtraction", "multiplication", and "division".

Algebras have been used in database theory to formalize database languages, especially so-called query languages. The relational algebra is an example of this, where the entities are relational tables, and the operators are "selection", "projection", "join", etc.

In a conceptual algebra the entities should, of course, be conceptual entities, and the operators should be conceptual operators "producing" (or defining) new conceptual entities from those which have already been specified. Thus, in a conceptual algebra associated with a conceptual model belonging to the OPR "family" the entities would belong to three categories: objects, properties, and relations.

An example of a conceptual algebra, based on the OPR approach to conceptual modelling, is given in appendix 2 and in [20]. The reader is recommended to (at least) "skim through" appendix 2 at this point.

4 Some propositions concerning concept-driven CASE tools

I suppose that many CASE tools that are available on the software market today could be claimed to be concept-driven in the sense that they cover some variation of conceptual modelling (and very often a variation belonging to the OPR "family"). However, I have sometimes the feeling that conceptual modelling (and other popular design techniques) are covered primarily for the simple reason that they are popular and "common practise" among system developers. As a contrast I would like to put forward the following normative proposition for consideration:

Proposition 1. **CASE tools should be based on conceptual modelling, because it is the right starting point for any information system design process. Every design decision suggested by a CASE tool should be derivable from some part of a conceptual specification.**

Of course this proposition is a very strong assertion, and I do not expect it to be accepted without argument, but I think it is worth consideration. There are at least three types of counterarguments:

(1) There are other methodologies (cf [19], for example) than conceptual modelling that are (also) indispensible for a proper specification of the user-relevant, external properties of an information system.

(2) The design decisions suggested by a CASE tool on the basis of a conceptual model may sometimes be clearly inadequate, or non-optimal, and have to be revised or modified on the basis of the (good) judgment of the human designer.

(3) Existing CASE tools sometimes support design methodologies, which are not based on any type of system specification, or are based on a specification of internal system properties rather than on user-relevant, external properties. (Example: A tool that proposes an "optimal" file organization on the basis of variables like "record length" and "transaction frequencies".)

Counterarguments of type (2) are real counterarguments only if the information, which the human designer uses in order to improve the decision suggested by the CASE tool, could not be regarded as a (non-formalized) part of a more complete conceptual model.

Counterarguments of type (3) can probably in most cases be interpreted as criticisms of existing tools and practises, which are often "partial". A real counterargument is at hand only if one could not imagine an improvement of the tool that replaces non-existing or internally oriented system specifications by explicit, externally oriented ones.

Counterarguments of type (1) can probably in most cases be reduced to a semantical problem: how much are we prepared to stretch the concept of "conceptual modelling". No doubt, there are important aspects of conceptual modelling (like dynamics) that are at present underdeveloped, but I can see no fundamental reasons, why we should not in the future "fill in the gaps" and extend the concept of a conceptual model to become a more or less complete specification of all user-relevant, external properties of an information system. Of course, some people may prefer to put other labels (than "conceptual model") on parts of the specification, but this does conflict with the basic idea.

Thus I believe that potential counterarguments to proposition 1 should more likely lead to revisions and extensions of the methodology of conceptual modelling than to a relaxation of the proposition. This leads us to the next proposition:

Proposition 2. CASE tools and information system design methodologies are intimately related to each other, and the development of one must go hand in hand with the development of the other.

This proposition, too, may seem to be rather controversial, since it appears to pull away the ground for the development of generalized CASE tools, at least before we have been able to standardize on one common systems development methodology (which I do not think will ever happen, for reasons that I will explain below).

Actually some CASE tool developers (see [18], for example) have rather long ago realized this potential problem, and have invented the term **"CASE shell"** (in analogy with "expert system shell"). A CASE shell is a platform of tools, or a tool-kit, on the basis of which a user may tailor his own CASE tool as an application (which in turn is instrumental in the development of the "real" applications of a particular organization).

Additional arguments for open CASE shells rather than closed, monolitic CASE tools are suggested by the next proposition:

Proposition 3. Information system development methodologies have to be application and organization dependent.

It is an empirical fact (at least in Sweden) that every organization tends to develop its own system development methodology, often on the basis of one or more "general" methodologies. This can be interpreted either as an undesirable lack of standardization, or as a desirable adaptation to the specific need of an organization and its specific type(s) of applications (or possibly as a bit of both).

For essentially the same reasons that we nowadays decentralize EDP departments and promote integration of EDP with other production factors in the organization's pursuing of its overall goals, I think that many organizations have to tailor an integrated methodology for the conceptualization, control and development of all its functions and activities. This integrated methodology should cover information systems develop-

ment, maintenance, and operation as an integral part, not as an isolated doctrine for EDP specialists.

From Proposition 2 and Proposition 3 follows:

Proposition 4. CASE tools have to be application and organization dependent.

It is tempting to conclude with

Proposition 5. A CASE tool should be an integrated part of an orchestra of knowledge-based tools (expert systems) for the conceptualization, control, and development of the activities of an organization.

5 Statistical information systems

Statistical offices and statistical information system can be seen as one type of organization and one type of application for which the propositions stated in the previous section could be tested. In a way that is what I have been doing during the last two decades, and in this section I will summarize some of the results of this work.

5.1 Some historical background

Conceptual modelling as an analytical tool was introduced at Statistics Sweden at the end of the 1960's, and several projects - methodological as well as software oriented - were carried out in the office during the 1970's. Of course, in the beginning of this work the term "conceptual modelling" had not yet been coined. Instead we called it "infological modelling". The basic theoretical platform was summarized and presented in my doctoral thesis [2]. Many ideas were based on the seminal work of Langefors [1].

One important reason for this early interest in conceptual modelling was the importance of conceptual definitions in the daily work of a statistical office. The task of a statistical office is to supply decision-makers at various levels and positions in society, as well as the public at large, with a reasonably rich and objective picture of the social, economical, and physical conditions in a country, in order to facilitate the democratical process resulting in political decisions, as well as planning and control activities undertaken by companies and public institutions. The conceptual definitions that are underlying the collection, processing, and presentation and analysis of public statistics have a significant impact on everybody's perception and understanding of the "real world" around us. Sometimes an uncertainty about the definition of a key concept, like "unemployment" can cause heated debates. The seemingly neutral and colourless figures of a statistical table often lend themselves to different interpretations, some of which are "reasonable" or at least "possible", whereas others may be due to ignorance or (accidental or deliberate) misunderstandings of the conceptual definitions involved.

Some of the projects that were based on (and promoted) the development of infological/conceptual modelling were:

- the establishment of a conceptual framework of socio-demographical statistics (SSDS) corresponding to the system of national accounts (SNA) in economic statistics;

- the establishment of a metadata repository, or metadatabase, "the Variable Catalogue", containing definitions and descriptions of the surveys, objects, and variables of the statistical office;

- the development of an "archive-statistical system" (ARKSY), consisting of databases, metadata, and software (ARKDABA) for rapid and flexible production, on demand, of "new" (combinations of) statistics, based on existing data sources;

- the development of a system of "statistical databases" (SDB), consisting of databases, metadata, and software for rapid and flexible retrieval, (re)production and analysis of aggregated statistics;

- the development of a high-level (4GL) software family (the TAB68 family) for supporting typical processes in statistics production like tabulation and editing;

- the development of a methodology (the SCB model [22, 24]) for the (infological and datalogical) design and construction of statistical information and data processing systems; key elements and ideas:

    - "infological design" before "datalogical design";
    - "the object graph";
    - "flat files";
    - high-level software components like the in-house developed TAB68 family or commercial products like SAS;

- the development of a CASE tool (the CONDUCTOR [25]) supporting (parts of) the SCB model for systems development; key components:

    - the DOK system for documentation;
    - automatical generation of control statements (in IBMs Job Control Language);
    - automatical generation of applications in some of the software products recommended by the SCB

model for systems development;

- automatical transformation of metadata between some of the recommended software products;

- the development a software system (the Base Operator System, BOS) supporting high-level development of statistical applications on the basis of an algebra for data manipulation adapted to the needs of statistics production (the Base Operator Algebra); this work was carried out jointly with a number of other statistical offices within the framework of the UN/ECE Statistical Computing Project [15].

Some of the projects listed above were more or less successful, others were failures. However, they have all contributed significantly to our understanding of statistical information systems and the proper design and construction of such systems. The most recent project, which could be added to the list above, is a project that aims at integrating statistical design and information systems design. A first step in this work is to harmonize the concepts used by statistical methodologists and the concepts used by EDP specialists, and to include statistical concepts and methodology as an integrated part of the above-mentioned SCB model for systems development. This work is obviously in line with the propositions stated in section 6 of this paper.

5.2 Some characteristics of statistical information systems

A statistical information system may be defined as an information system where the output oriented processes deal with information (macro-information) about groups of objects that is the result of transformation processes that aggregate information (micro-information) about individual objects that is dealt with in the input oriented processes. Typical aggregation operators that control the aggregation processes are frequency counting, summation, averaging, correlation computation, and even sometimes more sophisticated estimations of statistical measures (characteristics, parameters).

A quite different approach to the definition of a statistical information system is to focus on the purpose of the information system. A typical purpose of a statistical information system is to support high-level (strategic, directive) decision-making. This distinguishes statistical information systems from, for example, administrative information systems, the purpose of which is typically to support more routine (operative) decision-making. Moreover, in an administrative information system both the input and the output processes typically deal with information about (the same) individual objects (micro-information).

In passing, it should be pointed out that the classification of information into micro- and macro-information is a relative (rather than an absolute) classification. The macro-information output from one aggregation process may sometimes be fed into another aggregation as input micro-information, and this type of iteration may occur an arbitrary number of times.

5.3 The design of a statistical information system

The design of a statistical information system often starts from (at least) three different directions more or less in parallel:

(1) From the output side: which tables are to be produced? (Specification of tabulation plan.)

(2) From the input side: which questions are to be put to the respondents (specification of survey questionnaire), and which information can we obtain from other sources (specification of the information contents of available and relevant registers etc).

(3) From the problem side: which are the problems that are to be tackled with information from (among other sources) the particular statistical survey or statistical information system under design? (Problem specification.)

Theoretically it is easy to say that (3) should precede (1), which should in turn precede (2), but practically this may be an unrealistical idealization, particularly if we take cost/benefit aspects into serious consideration. For example, a particular potential information output may on the one hand be more or less useful in illuminating a certain problem, and on the other hand more or less costly to produce, depending on, among other things, whether it can be produced from information inputs that can be found in available registers, or whether it has to be produced on the basis of a survey questionnaire.

Of the three design directions mentioned above, (1) is of course closely associated with the input oriented view of the conceptual model, and (2) is associated with the input oriented view. (3) would be facilitated by a stringent "reality model", that is, it would be related to the "database view", or maybe even better with the complete conceptual model, containing and integrating the different views.

6 Modelling the four major subsystems of a statistical information system

We shall now discuss how conceptual modelling can be used for deriving the design of a statistical information system and its four major subsystems, and how we can do this both on a contents-oriented, infological level, and on a hardware/software-oriented, datalogical level.

6.1 Modelling the statistical database

On the infological level the specification of the database is identical with the conceptual model. If we assume that the database should be implemented under a relational database management system, or as a set of flat files managed by some suitable data manipulation language, like the Base Operator System [15] or SAS, the transformation of the infological model

into a datalogical model becomes fairly straightforward, too. Basically objects, many-to-many relations, and multi-valued variables become relational tables (flat files), and many-to-one relations become foreign key columns in the relational table corresponding to the object in the "many-end" of the relationship. This transformation into a **default specification** can easily be automated, but it may sometimes need to be optimized or "tuned" for efficiency reasons. For example, the designer may consider splitting up a big relational table (by columns or by rows) into several smaller ones on the basis of an analysis of the expected transaction traffic against the database. Conversely the designer may contemplate the consolidation of several relational tables into one, in order to reduce the number of table accesses in order to respond to certain transaction types, even though this may lead to (datalogical) redundance, and thus a lower degree of normalization in the sense of the relational theory. Another important design decision on the datalogical level concerns the specification of indexes and other auxiliary structures that aim at speeding up the processing of the expected database traffic.

The conceptual model may contain (infological) redundance. For example, a certain variable may be derivable from other variables. A common situation in statistical databases is that variables of an object on a higher level of aggregation (macro-level) are derivable (by aggregation) from variables on a lower level of aggregation (micro-level). In such a situation the designer may choose to store only the microdata or both the microdata and the macrodata. Ideally a user of the database should never need to know which of the alternatives that the designer has chosen. Actually the user should have the freedom to **think of** the data in the database either as microdata or as macrodata, and to formulate queries accordingly, regardless of how the data are actually stored. Furthermore it is sometimes desirable that the user can think of the database in terms of (imaginary) microdata even when only macrodata are physically stored. Generally speaking, whenever there is redundance in the conceptual model, the designer has the option **either** to store **all** corresponding data, implying (datalogical) redundance, which can be used for speeding up retrieval operations, and possibly for consistency checks, **or** to store **some** corresponding data and derive others by software procedures corresponding to the definitions of redundant concepts.

Many datalogical design decisions are dependent on a good specification of the expected transaction traffic between the database and its environment. This specification should be derivable from the conceptual model. In principle the object system dynamics implies the input-oriented traffic, and the information needs implies the output-oriented traffic, but when estimating the total transaction traffic that will hit the database, one must also take into account additional transactions generated inside the information system itself, for example in the data editing function of the input-oriented subsystem.

6.2 Modelling the input-oriented subsystem

There are two fundamentally different types of inflows of information (and data) to statistical information systems. On the one hand we have the

traditional type of inflow, where the input arrives in **natural batches** corresponding to the executions of censuses and surveys. Such inflows are initiated and controlled by the statistical agency responsible for the censuses and surveys. The information is essentially statical, since it typically informs about states of the surveyed object instances in the object system. A limited amount of dynamical information can be derived by comparing successive states, either on the micro-level (as in so-called longitudinal studies), or (more often) on the macro-level by comparing aggregated figures (example: the price development in terms of the consumer price index).

The other type of inflow to statistical information systems is **event-based.** Every time an event of a certain type (for example a crime, a traffic accident, or some type of decision) occurs in the object system, a report transaction is generated and transmitted (very often through an administrative system) to the statistical information system, where the database can, at least in principle, be updated in much the same way as in an administrative information system. However, the "real time requirements" are typically not so high in a statistical system; some delay and "batch formation" may be quite acceptable.

For input-oriented subsystems with inflows of (only) the first type (natural batches), the updating is in a sense trivial and can be derived from a state-oriented conceptual model alone. The updating part of input-oriented subsystems with inflows of the second type (event-reports) can be derived in much the same way as in administrative information systems, that is, by analyzing the dynamical aspects of the conceptual model, represented by the consequence matrix and/or life history diagrams.

A typical feature of statistical information systems is that the set of observed object instances is sometimes only a subset, a **sample**, of the whole **population** of instances of a certain object type in the object system. In such **sample surveys** the design of suitable **sampling and estimation** procedures is an essential part of the statistical design and requires competent statistical expertise. The sampling process can be regarded as a part of the input-oriented subsystem; it is often based upon a **randomized** and **stratified** selection of objects from a **frame**, that is, a register that represents (or is in a known way related to) the population of interest. Information about the sampling procedure is a type of metadata that is an equally important input to a statistical information system (of sample survey type) as the object data themselves.

The **input view** of the conceptual model should specify the data structures as they are conceptualized during data entry, coding, and editing. If the data is collected by means of questionnaires and forms, via paper and pencil, or directly through a computer terminal, the data structures are often hierarchical, corresponding to hierarchically related objects in the object system. The **routing structure** between the questions in a questionnaire can typically be described in the same way as a structured program [17], and thus modelling techniques known from the area of structured programming, like JSP diagrams could turn out to be useful. Different routings depending on the answer to a particular question

typically correspond to different subtypes in the conceptual model (cf appendix 1).

An interesting methodological work on deriving the questionnaire design from conceptual modelling can be found in Balestrino et al [26].

Coding can be regarded as a transformation from one domain of values to another domain for the same variable.

Theoretically the specification of editing rules becomes trivial if one has a complete conceptual model. In fact one editing rule is enough: the data should conform to the specified conceptual model. Thus checks for the validity and consistency of the input data should as far as possible be automatically derived from the conceptual model, including both its statical and its dynamical parts, rather than "invented" separately; see for example Graves [11]. In practice of course the editing problem is not quite so simple. Even if conceptual modelling is done conscientiously, it will not always be so complete that all desirable editing rules are automatically implied by the conceptual model. In particular, so-called macro-editing rules may still have to be added separately. Too many and too complex editing rules may also be generated, and the conceptual model in itself will not automatically prescribe what to do with data that do not conform with the conceptual model, for example whether it is the incoming data or some data already existing in the database that should be regarded as "wrong".

On the datalogical level it may be a fairly complex task to handle the mapping between a hierarchical, input-oriented view and the base version of the conceptual model, especially if editing should be done interactively with immediate rechecking of manually updated data. However, there are already some commercial software products like SAS and PARADOX that seem to be able to cope with this problem in an acceptable way.

6.3 Modelling the output-oriented subsystem

The output-oriented subsystem should be derived from a specification of the information needs of the users. Such a specification could be regarded as a complement to the conceptual model, and it could also be used as a tool for checking the completeness of the statical and dynamical parts of the conceptual model. Languages for specifying information needs visavi conceptual models have been proposed, for example INFOL based on alfa-beta- and alfa-beta-gamma-analysis [2], [15].

The output-oriented subsystem of a statistical information system should be able to retrieve data from the statistical database, execute different kinds of statistical analyses, and produce tables, graphs, and other forms of presentations of statistical results. On a conceptual level many of these functions may be modelled in terms of **statistical queries**, or so-called **alfa-beta-gamma-delta queries**:

α: for <object type> with <property>
ß: give [<statistical operator>(<list of variables>)]*
τ: by <combination of variables>
δ: where <list of definitions>;

Example. "Number of households with more than one person, and their average income, by region and type of household":

for HOUSEHOLD with size > 1
give count, average(income)
by region*type
where region = REGION(via LIVE_IN).name;

Alternatively, we may think of the statistical query as an expression in the conceptual algebra:

<obj type>(with <prop>)(by <partitioning>).
[<op>([<var>]*)]*

The same example as above could then be phrased:

HOUSEHOLD(with size>1)(by region*type).
(count, average(income))

It should be noted that all properties and variables appearing in the schemes above could themselves be expressions in the conceptual algebra. In the alfa-beta-gamma-delta scheme the delta clause offers a convenient alternative for introducing such derived concepts, and to give them, at the same time, their own names.

The graphical technique used by Shoshani [16] and others is another way of modelling statistical queries. This technique is sometimes not purely conceptual, but takes also into account table layout aspects of the query, like which variables in a cross-classification that should occur along the vertical axis, and which should occur horizontally.

By analyzing a representative set of statistical queries that are expected to hit the database, the designer (and/or the design tool) could make rational decisions concerning access paths to be supported on the datalogical level. For example, variables appearing in the alfa part of an alfa-beta-gamma-delta query are candidates for being indexed.

6.4 Modelling the transformation subsystem

The transformation subsystem should be derivable from the combined specifications of the three other subsystems: different transformations are necessary to transform within and between

- data structures representing the base version of the conceptual model

- data structures representing input-oriented views
- data structures representing output-oriented views

To some extent these transformations can be described precisely, and yet on a relatively high level by means of languages such as (on the infological level) the conceptual algebra illustrated in this paper, and (on the datalogical level) the relational algebra [7], SQL [7], and the base operator algebra [15, 23].

7 Conclusions

Some of the major conclusions of this paper were stated in the form of propositions in section 4. Among other things I seem to have claimed that

- conceptual modelling is (or can be extended to become) a comprehensive and complete methodology for specifying the external requirements on an information system;
- a satisfactory design of an information system and its different subsystems and functions can be derived (in a manual or computer-aided way) from such a comprehensive and complete specification (an **extended conceptual model**);
- the specification should be structured as a set of (sub)specifications, corresponding to views of the conceptual model, in accordance with a **functional architecture** of the information system, containing a database subsystem, an input-oriented subsystem, an output-oriented subsystem, and a transformation subsystem;
- the design methodology and the computerized tools supporting it must be adapted to the organization and the type of applications that the information systems are to support.

The organization and application dependency of information systems development methodology and tools is verified by my experience from information systems development at a statistical office, which I have briefly summarized in the paper.

My conclusions, if correct, imply that so-called CASE tools should be tailored for the particular needs of particular types of applications and organizations, hopefully on the basis of open-ended, generalized application development platforms. Such CASE shells or tool-kits could probably make good use of a number of generalized software products that are already available "off the shelf" like word processors, drawing programs, database management systems, and expert system shells.

8 References

[1] Langefors, B: Theoretical Analysis of Information Systems; Studentlitteratur, Lund, 1966, 1973

[2] Sundgren, B: An Infological Approach to Data Bases; Urval nr 7, Statistics Sweden, Stockholm 1973

[3] Klimbie, J W and Koffeman, K I (editors): Data Base Management; Proceedings of the IFIP Working Conference on Data Base Management, North-Holland 1974

[4] Sundgren, B: Theory of Data Bases; Mason/Charter, New York, 1975

[5] Chen, P P-S: The Entity-Relationship Model - Toward a Unified View of Data; ACM TODS 1, No 1 (March 1976)

[6] Brodie, M L, Mylopoulos J, and Schmidt J W (editors): On Conceptual Modelling - Perspectives from Artificial Intelligence, Databases, and Programming Languages; Springer Verlag, New York, 1984

[7] Date, C J: An Introduction to Database Systems; Volume I, 4th Edition, Addison-Wesley 1986, and Volume II, Addison-Wesley 1983

[8] Smith, J M and Smith D C P, Database Abstractions: Aggregation and Generalization; ACM TODS 2, No 2 (June 1977)

[9] Kent, W: Data and Reality; North-Holland 1978

[10] ANSI/X3/SPARC Study Group on Data Base Management Systems: Interim Report; FDT (ACM SIGMOD Bulltetin) 7, No 2 (1975)

[11] Graves, R B: Data Base Modelling as an Aid to Data Editing; Statistics Canada and UN ECE/CES ISIS'79 Seminar, Bratislava 1979

[12] Sundgren, B: Conceptual Design of Data Bases and Information Systems; R&D Reports E19, Statistics Sweden 1984

[13] Malmborg, E: The OPREM Approach - An Extension of an OPR Approach to Include Dynamics and Classification; R&D Reports E12, Statistics Sweden 1982

[14] Jackson, M: System Development; Prentice-Hall 1983

[15] Sundgren, B: Coordination Proposals to Improve the Impact of SCP Software Development; Report (and appendices) to the UN/ECE Statistical Computing Project, Geneva 1988

[16] Shoshani, A: Statistical Databases: Characteristics, Problems, and Some Solutions; Proceedings of the 8th International Conference on Very Large Data Bases 1982

[17] Bethlehem, J G; Denteneer, D; Hundepool, A J; and Keller, W J: The BLAISE System for Computer-Assisted Survey Processing; Netherlands Central Bureau of Statistics 1987

[18] Dahl, R; Eriksson, D; Johansson, L-Å; Sundin, U; and Torbjär, H: RAMATIC - A Modeling Support System; SYSLAB Report No 34, Chalmers University of Technology and University of Göteborg 1985

[19] Nilsson, A G: Information Systems Development - A Frame of Reference and Classifications; Polish-Scandinavian Seminar, Paraszyno, Poland, June 1988

[20] Sundgren, B: Conceptual Modelling as an Instrument for Formal Specification of Statistical Information Systems; 47th Session of the International Statistical Institute, Paris 1989

[21] Malmborg, E: On the Use of Semantic Models for Specifying Information Needs; Statistics Sweden R&D Reports 1989:5

[22] Sundgren, B & Lagerlöf, B & Malmborg, E: User-Oriented Systems Development at Statistics Sweden; Statistics Sweden R&D Reports E24 and Proceedings of the ISIS'86 Seminar, Bratislava, Czechoslovakia, May 1986

[23] Sundgren, B: Base Operators as a Tool for Systems Development; Statistics Sweden R&D Reports 1988:3

[24] Lagerlöf, B: Development of Systems Design for National Household Surveys; Statistics Sweden R&D Reports 1988:4

[25] Sundgren, B: A Session with the CONDUCTOR; Statistics Sweden R&D Reports E23, 1985

[26] Balestrino, R & Montagna, S & Rossi, M: Survey Questionnaire Design - Work-in-Progress; ISIS'90 Seminar, Bratislava, Czech and Slovak Federative Republic, May 1990

APPENDIX 1.

THE OPR APPROACH TO CONCEPTUAL MODELLING

(See also [12] and [20].)

1 Basic version of the approach

1.1 Basic concepts

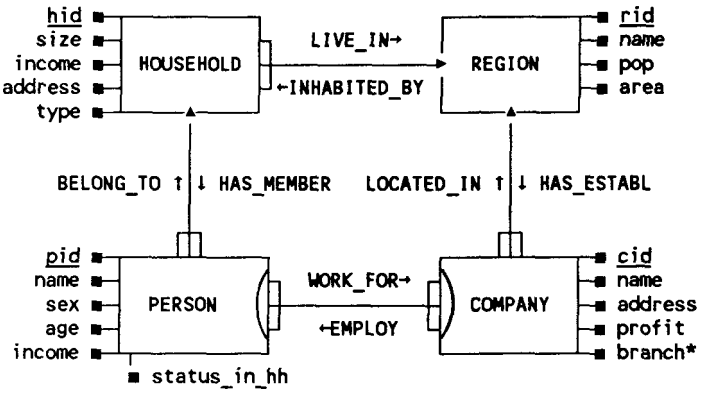
The Object-Property-Relationship (OPR) approach is an example of a methodology for conceptual modelling that is based on three concepts: objects, properties, and relations. With a slightly different terminology, this type of approach is often called the Entity-Attribute-Relationship (EAR) approach, or the Entity-Relationship (ER) approach.

An **object** is any concrete or abstract entity (physical object, living creature, organization, event etc) that the users of the information system may be interested to have information **about**. Objects will always have **properties**, quantitative or qualitative. For example a person (as modelled for a certain purpose) may have an age, a home address, an income, etc. A company may have a certain number of employees, a certain legal form (like 'incorporated'), a certain economical result, etc. A traffic accident may be characterized by the time when it happened, the number of persons and vehicles involved, the properties of the road, where it took place, weather conditions, etc. For both objects and properties it is important to distinguish between **types** and **instances** (occurrences). Property types are usually called **variables** or **attributes**. A variable may also be thought of as a function between a set of object instances belonging to a certain object type and a set of **values** belonging to a certain **domain**. Being a function a variable should be **single-valued**, but sometimes modelling situations occur, where a variable needs to be specified as **multi-valued**. For example, one and the same company may be active in several branches of industry at one and the same time.

The logical link between an object and a property is sometimes called an **association**. Objects may also be linked to other objects. Such links are called **object relations**, or simply **relations** (relationships). Relations are often binary, that is, they link two objects to each other, but there are sometimes relations of a higher **degree**, or **dimension**. For example, a country (object 1) may export a certain commodity (object 2) to another country (object 3); thus this trade relation is an example of an object relation of degree 3.

1.2 The object graph

An object system can be visualized by means of a so-called **object graph**. An example is shown below. In the object graph, the objects are represented by rectangles, the (object) relations are represented by lines between the object rectangles, and the variables of the objects are represented by "dots" linked to the object rectangles. An asterisk (*) after the name of a variable indicates that the variable is multi-valued. All concepts are represented on the **type** level in the object graph.



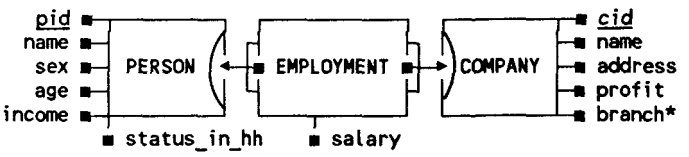
1.3 Relations: functionality, cardinality, and partiality

The object graph indicates the **functionality** for relations. A binary relation may be one-to-one (<--->), one-to-many (<---<), or many-to-many (>---<). **Cardinality** is a more general concept, which can be used also in the specification of relations of higher degree than 2.

The "half-circles" inside object rectangles in the object graph indicate **partiality** (as opposed to **totality**) of the relation visavi the object type, that is, only some (but not all) of the object instances belonging to the object type participate in the relation. For example, only some persons work for companies, and only some companies have persons employed.

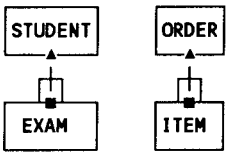
1.4 Relational objects and dependent objects

Sometimes we need to associate a property with a relation rather than with an object. For example, in the object graph above, we might like to indicate the salary that a person obtains by being employed by a certain company. This can be solved by **objectifying** the relation WORK_FOR/EMPLOY into a so-called **relational object** (an entity which is at the same time a relation and an object) EMPLOYMENT:



Relational objects are an example of **dependent** (or **weak**) objects in the sense that all instances of the object are, for their own existence, completely and essentially dependent on the existence of a particular instance of another object. Thus a particular EMPLOYMENT instance in

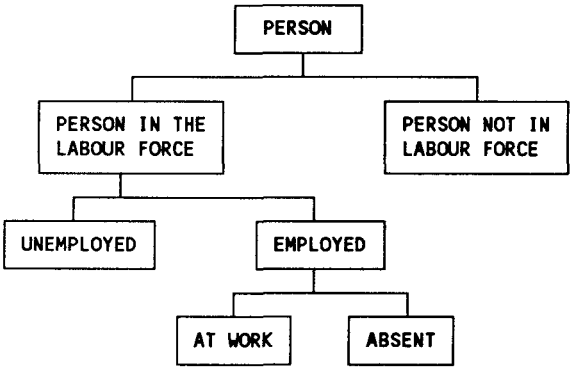
the example above is for its existence essentially dependent on the existence of a particular PERSON instance, as well as on the existence of a particular COMPANY instance. Other examples of dependent objects could be the EXAMs passed by a STUDENT, or the ORDER_ITEMs of an ORDER:



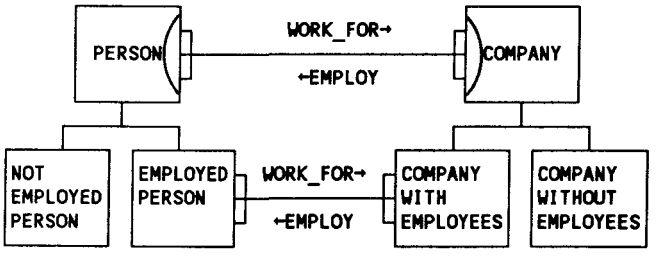
Note the usage of the dot symbol (■) to indicate the dependent object.

1.5 Generic hierarchies and subtyping

One important refinement of the OPR modelling approach is the introduction of **generic hierarchies** of object types ([8], [13]), where objects on lower levels represent **subtypings** of objects on higher levels, and where the lower level objects **inherit** the properties of the higher level objects. An example:



A related refinement is the distinction between **total** and **partial** relationships; a relation is partial with respect to an object type, if only a proper subset of the objects belonging to the object type is involved in the relation. Partiality implies subtyping:



The example shows that by properly subtyping PERSON and COMPANY, we may change the status of the relation WORK_FOR/EMPLOY from 'partial' to 'total'.

As indicated by the object graph the basic version of an OPR model is essentially statical. It models the situation in the object system at a certain point of time. Later developments have introduced dynamical aspects into the OPR approach; see for example works by Jackson [14] and Malmborg [13]. Dynamical aspects are important in several ways:

1. In order to come to grips with the **definition** of a certain object type, it is often essential to penetrate the **criteria for birth and death** of objects belonging to the object type. For example: what is it that constitutes the birth (or the death) of a household, a company, etc; or similarly: what **changes** can an object undergo, and still remain in some sense **the same object**?

2. Births and deaths of objects in the object system, and (other) **changes of states** of (the variables of) the objects imply a need for **update transactions** visavi the information system and its database.

3. The users of the information system are often interested to study the **development over time** in the object system. This development can sometimes be described in terms of the situation in the object system at regular time intervals, but sometimes (for example in connection with longitudinal studies) it is necessary to be able to describe and request information about more complex chains of object-related events and changes of states.

The **consequence matrix** [12] and **life history diagrams** [14] are two examples of conceptual tools and graphical techniques that have been developed to systematize and illustrate the dynamical aspects of an object system. We shall use the following example to illustrate these tools:

2.1 Birth/death analysis

During birth/death analysis we look systematically for **birth/death events** of each one of the objects and each one of the relations in the object system. In the example above we may thus identify events like

```
↑ PERSON: E1 = "physical birth of person";
↓ PERSON: E2 = "physical death of person";

↑ HOUSEHOLD: E3 = "some person(s), but not all, leave a household and form(s) a new one";
↓ HOUSEHOLD: E4 = "all persons (possibly one) leave a household and move into an existing
                   household",
             E5 = "all persons (possibly one) in a household die" (implies event E1*);

↑↓ REGION:   E6 = "political/administrative decision to change the regional structure";
```

```
↑ BELONG_TO/MEMBER: E1, E3, E4,
                    E7 = "some person(s), but not all, leave a household, and move into
                       another household";
↓ BELONG_TO/MEMBER: E2, E3, E4, E5, E7;

↑ LIVE/INHAB: E8 = "a household moves between regions";
↓ LIVE/INHAB: E8, E5.
```

2.2 Consequence analysis and the consequence matrix

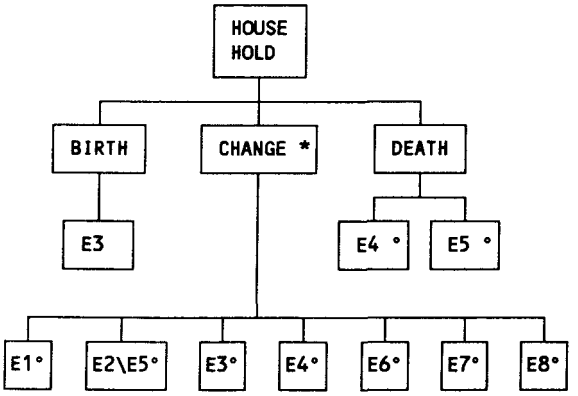
Birth/death analysis is a kind of **precedence analysis** [1]. We may turn it around into **succedence analysis** [1] by asking for all birth/death consequences of a certain event. The results of such a **consequence analysis** may be summarized in a **consequence matrix.** In the example we get:

Event	PERSON	BELONG_ TO	HOUSE- HOLD	LIVE_IN	REGION
E1	↑	↑			
E2	↓	↓	↓°	↓°	
E3		↓*↑*	↑	↑	
E4		↓*↑*	↓	↓	
E5	↓*	↓*	↓	↓	
E6				↑*↓*	↑*↓*
E7		↓*↑*			
E8				↓ ↑	

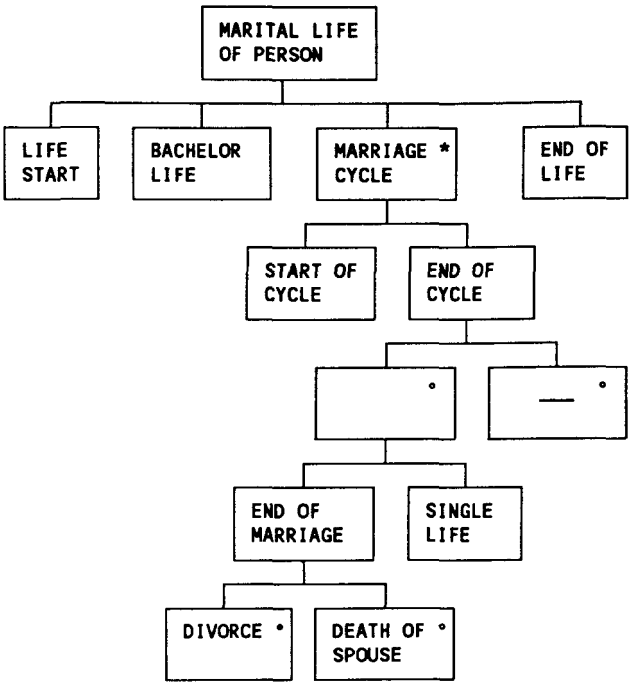
An upward directed arrow means "birth", a downward directed arrow means "death". Asterisk (*) means repetition (zero, one, or more), and a small circle (°) indicates a conditional effect. Every column should contain at least one birth event and (usually) at least one death event, otherwize we have missed something in the analysis.

2.3 Life histories and career analysis

By bringing together the information in the consequence matrix about events concerning a certain object and other objects' relations to this object, and by structuring this information properly, we can outline a typical life history of an instance of the object, as implied by the conceptual model. JSP diagrams [14] may be used to visualize life histories. In the example above we may model the life history of a household in the following way:



Similarly we may model so-called **careers** like the marital life, the education, or the criminal career of a person, the medical record of a patient, etc. Example:


MARITAL LIFE OF PERSON
LIFE START
BACHELOR LIFE
MARRIAGE * CYCLE
END OF LIFE
START OF CYCLE
END OF CYCLE
END OF MARRIAGE
SINGLE LIFE
DIVORCE *
DEATH OF SPOUSE

APPENDIX 2

OUTLINE OF A CONCEPTUAL ALGEBRA FOR FORMING NEW CONCEPTS AND VIEWS

(See also [20].)

It is often necessary to distinguish between the **base version** of a conceptual model and different **views** of it, containing permanent or temporary extensions and modifications that adapts the conceptual model to the needs of a particular (group of) user(s) or application(s). For example, in a statistical information system it is usually important to be able to apply both a **micro perspective** and a **macro perspective** to the same object system, and the same set of data, regardless of how the data are actually stored.

All the objects, properties, and relations in a view must be defined in terms of the concepts specified in the base version of the conceptual model. The author of this paper once outlined a conceptual (or infological) query language [4]. Here we shall show, by means of examples, how a generalized version of this language can be used for the specification of new concepts and views, when modelling a particular object system. A formalization of most of the syntax used here can be found in [15].

The different definition schemas, examplified below, for forming new objects, variables, and relations may be thought of as a set of **conceptual operators** that produces, in the sense of an **algebra**, new concepts from those which have already been defined.

1 Conceptual operators defining new object types

Example 1. Definition of a new object type:

```
HEAD_OF_HOUSEHOLD == PERSON(with status_in_hh = 1);
```

The general definition scheme for deriving new objects is:

<new object type> == <object type>(with <property>);

The <property> may in turn be a derived concept, involving boolean expressions, derived variables (see examples below), etc. The '= =' symbol should be read 'equals by definition'.

2 Conceptual operators defining new variables by means of grouping, arithmetical transformation, aggregation, and adjunction

Example 2. Definition of a new variable by means of grouping:

```
PERSON.agegroup == 1 if PERSON.age<18,
                   2 if PERSON.age>=18
                        and PERSON.age<66,
                   3 if PERSON.age>=66;
```

The general definition scheme for grouping variables is:

<grouped variable> == [<constant> [if <property>]]*;

where [<element>]* is a list of elements, separated by commas, and [<element>] is an optional element.

Example 3. Definition of a new variable by means of arithmetical transformation:

```
HOUSEHOLD.income_per_capita == HOUSEHOLD.(income/size);
```

The general definition scheme for arithmetical transformations of variables is:

<new variable> == [<arithm expr> [if <property>]]*;

Note that with this definition scheme, variable grouping becomes a special case of arithmetical transformation.

Example 4. Definition of a new variable by adjunction:

```
PERSON.address == PERSON.HOUSEHOLD(via BELONG_TO).address;
```

The general definition scheme for adjoined variables is:

<adjoined variable> == <path>.<variable>;

where <path> determines a unique chain of relations from the object of <adjoined variable> to the object of <variable>. Each link in the chain has the form

<object type>[(via <relation>)].

where the 'via' part is necessary if there is more than one relation between two objects in the chain. Further it should be noted that <object type> could be derived. In particular it could be restricted by means of a '(with <property>)' clause in order to ensure that the ultimately generated adjoined variable becomes single-valued; (cf the definition of 'PERSON.main_employer' in Example 5 below).

Example 5. Definition of a new variable by means of aggregation:

```
HOUSEHOLD.size == HOUSEHOLD.PERSON(via HAS_MEMBER).count;

HOUSEHOLD.income == HOUSEHOLD.PERSON(via HAS_MEMBER).sum(income);

PERSON.main_employer == PERSON.EMPLOYMENT(with max(salary)).COMPANY.cid;
```

The effect of inserting an aggregation operator with zero (like count), one (like sum, average, max, min), or more (like correlation) arguments into a derivation is to reduce a set of values into one single value. The aggregation operator also has the effect of adjoining the aggregated variable to an object, which is on the next higher level in an **aggregation hierarchy**.

3 Conceptual operators defining new object relations

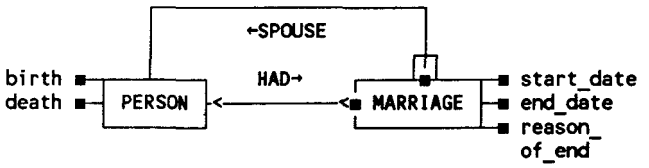
Example 6. Definition of a new object relation:

```
PERSON.REGION(via DOMICILE) == PERSON.HOUSEHOLD(via BELONG_TO).REGION(via LIVE_IN);

PERSON.REGION(via PLACE_OF_WORK) == PERSON.EMPLOYMENT(with max (salary)).COMPANY.REGION;
```

4 Dynamical aspects

In some types of information systems life histories of objects are of great interest, and techniques like JSP diagrams may be used for modelling the life history of an object as a structured flow of events associated with the object. The events that make up the life history of an object may also themselves be regarded as objects. By doing so we make it possible to incorporate the dynamical aspects of an object system in a traditional, state-oriented OPR model. This is done by defining one or more one-to-many relations between the object, whose life history we are interested in, and one or more categories of events. **Time** should be a mandatory property of the event objects, and a possibility to define a **conceptual ordering** of the instances of an object type on the basis of a property (in this case 'time') should be introduced among the conceptual tools of OPR, together with some operators (like 'first', 'last', 'next', 'before' and 'after') for referring to this ordering. For example we could model the marital history of a person like this:



And a derived variable like "a person's average time between marriages" could be expressed something like this (assuming proper handling of null values):

```
PERSON.avg_time_btw_marr == PERSON.MARRIAGE(via HAD).avg(next(start_date)-end_date);
```

R & D Reports är en för U/ADB och U/STM gemensam publikationsserie, som fr o m 1988-01-01 ersätter de tidigare "gula" och "gröna" serierna. I serien ingår även **Abstracts** (sammanfattning av metodrapporter från SCB).

R & D Reports Statistics Sweden are published by the Department of Research & Development within Statistics Sweden. Reports dealing with statistical methods have green (grön) covers. Reports dealing with EDP methods have yellow (gul) covers. In addition, abstracts are published three times a year (light brown /beige/ covers).

Reports published earlier during 1990:

1990:1 (grön)	Calibration Estimators and Generalized Raking Techniques in Survey Sampling **(Jean-Claude Deville, Carl-Erik Särndal)**
1990:2 (grön)	Sampling, Nonresponse and Measurement Issues in the 1984/85 Swedish Time Budget Survey **(Ingrid Lyberg)**
1990:3 (grön)	Om justering för undertäckning vid undersökningar med urval i "rum och tid" **(Bengt Rosén)**
1990:4 (gul)	Data Processing at the Central Statistical Office - Lessons from recent history **(M. Jambwa)**
1990:5 (beige)	Abstracts I - sammanfattning av metodrapporter från SCB
1990:6 (grön)	Sequential Poisson Sampling from a Business Register and its Application to the Swedish Consumer Price Index **(Esbjörn Ohlsson)**
1990:7 (grön)	Kvalitetsrapporten 1990 - huvudrapport **(NN)**
1990:8 (grön)	Kvalitetsrapporten 1990 - bilagor **(NN)**
1990:9 (grön)	Datorstöd vid datainsamlingen **(Carina Persson, Marit Jorsäter)**
1990:10 (beige)	Abstracts II - sammanfattning av metodrapporter från SCB
1990:11 (grön)	Rapport från DATI-projektets produktionsprov i arbetskraftsundersökningarna augusti 1989 - januari 1990 **(NN)**

Kvarvarande **beige** och **gröna** exemplar av ovanstående promemorior kan rekvireras från Inga-Lill Pettersson, U/LEDN, SCB, 115 81 STOCKHOLM, eller per telefon 08-783 49 56.
Kvarvarande **gula** exemplar kan rekvireras från Ingvar Andersson, U/ADB, SCB, 115 81 STOCKHOLM, eller per telefon 08-783 41 47.