TRENDS AND PERSPECTIVES IN CONCEPTUAL MODELLING

Colette ROLLAND
Universite PARIS I - UFR 06
17, Rue de la Sorbonne
75231 PARIS CEDEX 05

Corine CAUVET
Université PARIS I - IAE
162, Rue Saint Charles
75015 PARIS

ABSTRACT:

Conceptual modelling refers to the part of system development that involves investigating the problems and requirements of the users community and from that, developing a specification of the desired system.
Conceptual modelling addresses two major aspects: the conceptual product (the so-called conceptual schema) and the conceptual process (the modelling process to deliver the conceptual product). Contributions to the field of conceptual modelling have emphasized the product aspect. A large variety of conceptual models have proposed high level concepts and abstraction mechanisms by which systems may be described at a conceptual level. Conceptual models have proved to be extremely useful throughout the information system life cycle and, hence, to be one of the most fundamental tools in the area of information systems engineering. However the growing demand for large and complex information systems calls for the introduction of new and more precise, formal techniques to model reality.
In contrast to the mature level of research on conceptual models, there is an evident lack of understanding and formalization of the conceptual modelling process. This consists of knowledge acquisition and the validation cycle. The acquisition step has the purpose of abstracting and conceptualizing relevant parts of the application domain. The validation step has the objective to check whether the conceptual specifications are consistent and whether they correctly express the requirements informally stated by the users. We believe that an important research effort should be done in understanding and formalizing the modelling activities as well as in modelling the process

itself in order to develop advanced and automated supports for process guidance and automatization.
The paper discusses both aspects of conceptual modelling, states the main research results and outlines new research perspectives.

I. INTRODUCTION

The growing demand for information systems of ever-increasing complexity and size, calls for high level concepts and formal techniques to model systems at different levels of abstraction. The need for powerful conceptual tools and better abstraction forms has been recognized, in particular in the earlier phases of systems development, in industry, business and administration.

The conceptual level has been introduced in the context of a three-schema framework [ANSI.75] for database management systems. In this context, a **conceptual schema** is defined as "a unique central description of the information that may be in the database". The conceptual schema also includes the description of permissible actions (updating and retrieval actions) on the database [ISO.82]. Nowadays, a conceptual schema plays a key role in the development of a large variety of systems. It is often considered as a reference for the entire development process of systems.

The term **Conceptual modelling** is used in the following to refer to the activity in systems development which leads to develop a conceptual schema as a specification of the desired system. Research and development in conceptual modelling address two complementary aspects:

- **the conceptual product,** i.e the conceptual schema which is the objective to reach in modelling reality,
- **the conceptual modelling process,** i.e the route to travel to deliver the conceptual product.

During the last decade, researches have emphasized the product aspect. This has caused the introduction of a large variety of conceptual models by which the conceptual schema contents can be specified. Conceptual models mostly consist of modelling concepts and guidelines related to a language for specifying both the structure and the behavior of a system. They have contributed to deal with fundamental aspects in conceptual modelling such as modelling of complex objects, system behavior modelling and time modelling. Current evolution of conceptual models is strongly influenced by the principles of the object-oriented approach.

By contrast, very little attention has been payed to the conceptual modelling process. This results in a low level of support provided to conceptual modelers. However the emphasis on product modelling is shifting to process modelling. Understanding and formalizing the conceptual modelling process in order to develop advanced and automated support is nowadays an important research topic.

The paper is organized as follows: **Section 2** presents the two major aspects addressed by conceptual modelling, namely the conceptual product and the conceptual modelling process and mentions some general research topics. The **third section** outlines the fundamental results achieved in the development of conceptual models and stresses some of the problems that still require new researches. **Section 4** provides some light on the influence of the object-oriented paradigm on the subsequent development of conceptual models; it also defines a number of research topics raised by the emergence of object-oriented models. The **last section** focusses on research perspectives in the field of conceptual modelling process.

II. CONCEPTUAL PRODUCT AND CONCEPTUAL MODELLING PROCESS

Conceptual modelling takes place in the early phases of system development. It concerns the part of the system development that involves investigating the problems and requirements of the user community, and from that, building a requirements specification for the desired system. The term requirements engineering has also been used [HAGE.88], [DUBO.89] for this part of system development. The succeeding development part, where the specification is used to design and implement a working system which is verified against the specification may be called design engineering.

Conceptual modelling can be looked upon in terms of product (the conceptual schema) and process (the modelling process). Figure 1 places conceptual modelling aspects within the entire system development process.

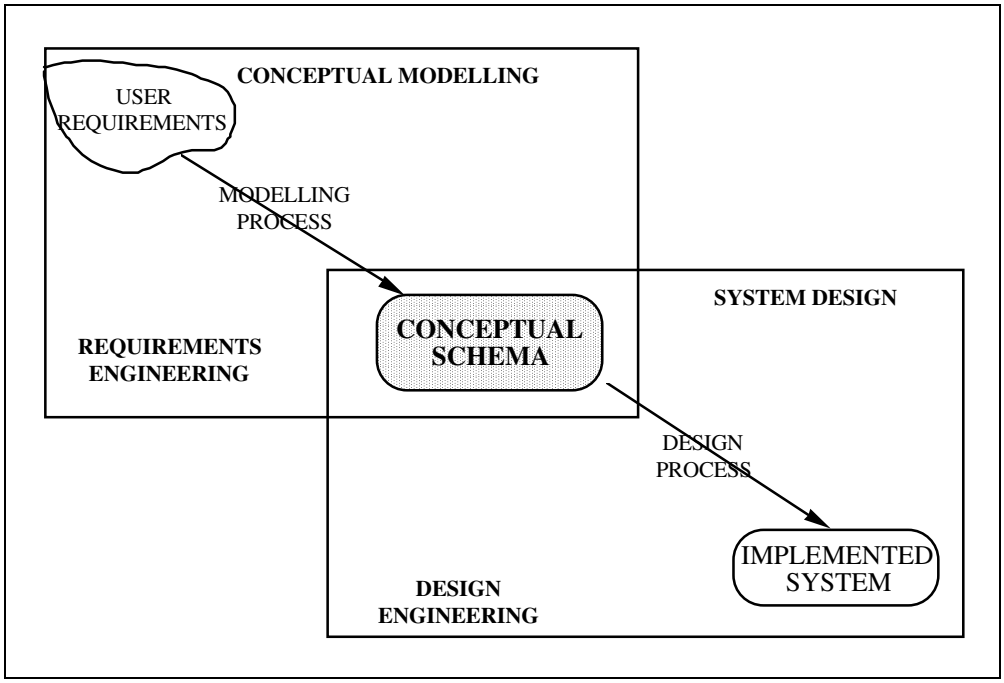

Figure 1 : Conceptual modelling and system development

The distinction product/process is widely accepted in software engineering and recognized as the core of any information system design methodology [OLLE.88]. We will use this distinction to introduce the field of conceptual modelling, its achieved results and unsolved problems.

II.1 THE CONCEPTUAL PRODUCT

The conceptual schema constitutes a specification of the information system to be built. According to the ISO TC97 Working Group [ISO.82], the contents of this specification should follow two major principles:

> *100 percent principle (partly given) : All relevant general static and dynamic aspects i.e. all rules, laws etc., of the universe of discourse should be described in the conceptual schema.....*
>
> *Conceptualization principle : A conceptual schema should only include conceptually relevant aspects, both static and dynamic, of the universe of discourse, thus excluding all aspects of data representation, physical data organization and access as well as all aspects of particular external user representation such as message formats, data structures, etc.*

According to these principles, the conceptual schema specifies the **functional requirements,** this means what the system is required to do. A conceptual schema includes typically conceptual entity classes with their relationships and attributes which represent information about some relevant real objects. The conceptual schema also involves events and / or operations and constraints (temporal and dynamic) which express the system behavior required by the users community. The functional part of the conceptual product is defined according to some modelling formalism, usually refered to as a conceptual model. Research work in the field of conceptual models has produced a number of papers, each of them giving the merits of a specific conceptual model. The document [ISO.82] has proposed several criteria for analysing present and future conceptual models (such criteria are capability of distinguishing between lexical and non-lexical entities, expressive power, ability to take into account static and dynamic aspects of some system....). However, it remains as a research problem to prove whether all existing models are different or whether they are nearly equivalent. Furthermore, existing conceptual models are not rich enough for enforcing the conceptualization principle (incorporating the static as well as dynamic properties and rules in one single conceptual schema) and are often ill-defined and difficult to use. To improve this situation, it is necessary to substantially refine modelling formalisms, make them more expressive and theoritically sound [SYSL.90]. Sections 2 and 3 of the paper present a survey of conceptual models and intend to define the influence of the object-oriented approach on them.

In addition, it is nowadays recognized as necessary to extend the contents of the conceptual product with **non-functional requirements specification**. Non functional requirements correspond to other forms of requirements such as interface constraints, performance constraints, economical constraints... These aspects of a system are often expressed in an unstructured way without any particular formalism. Furthermore, a great deal of time is often spent on functional requirements and the other requirements remain unrecorded or unaccounted and consequently tend to be lost in the system development. A research effort must be done on the problem of formalizing non-functional requirements and relating them to functional requirements.

II.2 THE CONCEPTUAL MODELLING PROCESS

Conceptual modelling also relates to the set of tasks required to produce the conceptual schema. Techniques and guidelines are needed for performing these tasks, for choosing the most appropriate task according a particular situation and for helping the modeler to make decisions. The conceptual modelling process may be viewed as consisting of four main tasks, that we assume to repeatedly take place during the process.

The **knowledge acquisition task** aims at acquiring a comprehensive description of a system. The problem is to capture knowledge, to discover and understand business functioning rules and to acquire the users' needs. Such knowledge may be found in existing automated systems (for example database systems), in documents usually used in information communication and in interviews with people. This task also involves to select and classify pertinent knowledge.

In the conceptual modelling process, the **conceptualization task** corresponds to a formalization task by means of a conceptual formalism. Formalization implies structuring and refining of the acquired knowledge. The most important problem in this task is to find the concepts best suited to express in a formal way the desired system structure and behavior. Acquiring domain knowledge and abstracting from them is nowadays pragmatically done by modellers, based on interviews of domain experts. There is a need for understanding and formalizing the process of knowledge acquisition and the conceptualization task to reach a point where automated support will be possible.

The **validation task** aims at checking whether the conceptual schema is consistent and whether it correctly expresses the requirements informally stated by the users. Correctness checking has been often tackled in association with the definition of the conceptual model. However validation of the specification against the users' requirements is still an opened research problem.

The **evolution management task** concerns the problem of schema evolution according to changes in the real system. The evolutionary nature of the user requirements normally implies that the existing conceptual schema is adapted to new needs. It is conceivable that evolution occurs even as the conceptual modelling activity itself progresses. Thus, a felt need may evolve while the first three tasks of the process are being carried out. A conceptual modelling process that supports evolution will make a significant contribution to the solution of the practical problem of system maintenance.

Today's methodologies and CASE tools provide unsatisfactory support for controlling the conceptual modelling process. The process is often viewed as a linear process in which the four tasks are organized in a predefined way. Producing methodological supports with automated tools for monitoring the conceptual modelling process is a research problem which has not yet been well explored. Corresponding research perspectives are introduced in section 5.

III. SURVEY OF CONCEPTUAL MODELS

Conceptual models were originately influenced by the databases field [ISO.82] [BORG.85], [BROD.86a], [BUBE.86], [SCHM.89]. This influence has lead to the development **database models** such as the hierarchical model, the network model and the relational model. The more recent **semantic data models** have proposed powerful modelling concepts and techniques for dealing with data-intensive systems. In addition modelling dynamic and temporal aspects of

information systems has been felt necessary. One can find in [BROD.86b] an historical view of conceptual models development from database models to models of the future.

The authors consider that the **modelling of complex objects**, **system behavior** and **temporal aspects** is a fundamental aspect of conceptual modelling. The section examines and characterizes main trends related to the three mentionned topics.

Some topics such as classical models (hierarchical, network and relational models) and theoritical work of defining data model semantics are not addressed in this paper. Advanced discussions on these aspects can be found respectively in [DATE.83], [DATE.86], [TSIC.82] and [ABRI.74], [REIT.86], [STAC.85], [HULL.87b].

III.1 SEMANTIC MODELS

Models referred to as semantic models correspond to a major improvement in the field of conceptual modelling, making possible a reasoning on real world entities and relationships between them, instead of building data structures. On the other hand they provide abstraction mechanisms which have widely proved their usefulness in conceptual modelling.

Three main papers [CHEN.76], [SCHM.75] and [SMIT.77b] have outlined two important ideas in conceptual modelling signaling the emergence of semantic data models. **Data independence** and **abstraction forms** for capturing additional semantics in the modelling process are the generic properties of semantic models. A large variety of semantic models have been proposed since the mid-1970s: E/R model [CHEN.76], SHM [SMIT.77a,77b], SHM$^+$ [BROD.82], SDM [HAMM.81], TAXIS [MYLO.80], DAPLEX [SHIP.81], RM/T [CODD.79], SAM* [SU.83] , THM [SCHI.83]... A suvey of 16 prominent semantic models is presented in [HULL.87a]. Others surveys and comparisons of semantic models may be found in [PECK.88] and [SCHR.84].

Data independence

The first idea addressed by semantic models is that of data independence: the conceptual schema should be free from the physical structure of the database. This makes possible a change at the data physical level without involving any modification of the conceptual schema. This principle is a major improvement over classical data models (file and database models) which offer little to support data independence. In addition, the conceptual model is closer to the human perception of the universe of discourse. This is the main objective of the well known Entity-Relationship model introduced by Peter Chen [CHEN.76]. P. Chen adopts the view that the real world consists of entities and relationships among them. Entities and relationships are characterized by properties. The three of them (entities, relationships and properties) are classified into types.

Support of abstraction forms

Abstraction involves the ability to emphasize details essential to the problem and to suppress all irrelevant details [BROD.86b]. Abstraction is essential in conceptual modelling due to the complexity of current information systems.
A large variety of surveys of the area of semantic models [HULL.87b], [PECK.88], [SCHR.84] converge on the idea that all semantic models make use of abstraction mechanisms among which classification, aggregation, grouping and generalization are fundamental ones. Four main advantages of abstraction forms can be outlined:

- (i) At first, abstraction forms allow to organize elements of the conceptual schema in a **hierarchical way;** they replace the flat structure resulting of the unique record construct provided by classical models [KENT.79]. For instance, the repeated use of aggregation allows the building of a hierarchy of objects. A link in this hierarchy expresses a "PART_OF" relationship between two objects. Generalization is another way to organize objects into a hierarchy in which a link represents an "IS_A" relationship between two objects. Used in a complementary fashion, aggregation and generalization allow to produce two hierarchies which can be thought of as two independent or orthogonal views of the same reality.

- (ii) Abstraction forms may be considered as **semantic relationships** which increase the semantic power of classical data models and improve the semantic richness of conceptual schemas. The four abstraction forms have their root in Artificial Intelligence with semantic nets [PECK.88], [SCHR.84] in which nodes are connected by four types of links: "PART_OF", "IS_A", "MEMBER_OF" and "INSTANCE_OF". Each form of relationship has a precise semantics such as the inheritance property for the "IS_A" relationship.

- (iii) Abstraction forms are specially adequate for the **design of complex objects** which are nowadays required by emerging data-intensive applications including CAD/CAM, office information systems and AI applications. In this context, aggregation and grouping (also called association) are very helpful to define higher-level objects from existing objects. An aggregate object is a complex object built up from component objects of different types. An object defined by grouping is also a complex object (also called set or group) which may be viewed as a set of objects of the same type. These abstraction forms allow to gradually make visible the structure of a complex object and how the individual components of the object relate to it and to each other. The semantic model SDM [HAMM.81] introduces the terms *base class* and *nonbase class* to distinguish classes which are defined independently of all other classes from classes which are structurally related to others classes by means of aggregation and grouping.

- (iv) Abstraction forms may be considered as **modelling facilities**, as they allow to reason at different levels of abstraction; using the generalization form, for instance, the modeler can emphasize similarities among objects, abstracting their differences at the generic level and thus define specific properties at the specialized level. They support reuse in the sense that the modeler may derive new objects from existing ones. Using aggregation for instance, it is possible to define a new object as a composition of existing objects (aggregation used in a bottom up approach). In the same way, generalization may be used to construct new objects by refining a generic one. Smith and Smith [SMIT.77a] discuss a systematic approach to database design in which aggregation is used either in a bottom-up fashion to capture relationships through aggregate objects or in a top-down fashion to structure a complex object through its decomposition into constituent objects.

Limitations of semantic models and need for further researches

Semantic models have emerged from the need of more expressive conceptual data models. They have fully achieved the original research objectives: economy of expression, modelling flexibility and modelling efficiency [PECK.88]. However they put the emphasis on the static aspect of systems, i.e their structure. In order to provide an advanced modelling support for both structure and behavior of a system, semantic models should be investigated in two complementary ways:

- **Increasing semantics of relationships with primitive operations**. As objects are connected through various kinds of relationships, the insertion, deletion or modification of one

object may impact other objects. A precise definition of primitive operations with their specific constraints for each kind of relationship (aggregation, generalization...) should increase their semantics. These primitive operation definitions should be clearly conveyed to the user of the model whereas they must be explicitly specified by them with desired constraints in most current models. The temporal hierarchic model (THM) [SCHI.83] is an attempt in this direction. THM defines primitive operations and proposes a set of dynamic axioms and side-effect axioms using dynamic and temporal logic in order to help maintaining the database consistency. In the same way, [SMIT.77b] defines relational invariants that can be considered as conditions on a relation to represent aggregation and generalization abstractions.

- **Dynamic modelling**. Systems have both structural and behavioral properties. Behavior refers to state transitions and dynamic properties (operations and their relationships). Semantic models should provide concepts for defining high level operations in a conceptual schema. Some semantic models have already proposed such extensions. SHM$^+$ and TAXIS for example, have included transaction and action concepts for handling dynamic modelling. In these approaches, abstraction forms such as aggregation and generalization can be applied on actions.

III.2 BEHAVIORAL MODELS

The purpose of behavioral models is to provide a conceptual formalism for expressing how and when changes occur to entities and relate with one another. Their main contributions in conceptual modelling concern concepts for handling dynamic aspects of systems, formalisms and techniques for reducing ambiguity, inconsistency and incompleteness in behavior specification and concepts for dealing with temporal aspects.

III.2.1 Operational approach versus declarative approach of dynamic modelling

There exists a great variety of behavioral models, which differ both in the concepts for modelling dynamic aspects of a system and in the related languages. We adopt the classification proposed by A. Olive in [OLIV.86] based on the approach taken by each model to deal with dynamic modelling. A. Olive suggests to distinguish operational and declarative models. In the operational approach, the system behavior is described through processes which are explicitely invoked, whereas in the declarative approach, the system behavior is described as a collection of rules which are invoked through object accesses.The two approaches are not exclusive; for instance in the OBLOG language [SERN.88], both approaches are mixed through CSP processes and safety and liveness rules. The authors state that the declarative way is preferable for expressing what little is forbidden whereas the procedural way seems preferable for expressing what little is allowed in the evolution of the system.

The operational approach

This approach has been strongly influenced by database notions such as the one of transaction. Examples of models which follow this approach are REMORA [ROLL.82], SHM$^+$ [BROD.82], TAXIS [MYLO.80] and OBCM [WAND.89]. They are based on two main assumptions :

(i) Changes in the computerized system corresponding to changes in the real world are defined by means of events and operations. An occurrence of a real-world external event triggers the execution of a set of operations (or transaction) which reflects the effects of the event on the

computerized system and produces a state transition. This approach makes a clear distinction between states and state changes of the system.

(ii) Operations as well as queries and integrity constraints can most often only access the current state of the database.

The term operational approach has been chosen by A. Olive with the purpose to express that this approach mixes the "logic" and "control" components of the modelized system i.e. the "what" the system should do and "how" it does it. Thus a dynamic specification statement has the following general form:

When <event occurrence identified>
if all <conditions> met
then < transaction execution>
on <objects>

Events express when a state change on entities or in the environment must happen; operations define how the entities are modified. Conditions must be defined in a way which guarantees that the transition is permitted. The transaction (set of operations) must produce a new valid state. An event is considered as a control mechanism of operation executions on entities.

For behavior modelling purposes, this approach suggests to consider the system at execution time and to define which actions to trigger when a certain event happens. Such an approach involves the specification of all permitted transitions in the conceptual schema.

It is well fitted to the modelling of business systems dynamics. It eases the mapping from the conceptual schema to a database solution which consists of data and transactions on data.

The declarative approach

The declarative approach aims at providing a complete specification of a system behavior expressing only its logical component. Contrarily to the operational approach, it emphasizes the specification of behavior in terms of rules without taking into account the control of these rules. The specification of the control component is entirely left to a subsequent phase of the system development. This approach has been largely influenced by knowledge representation techniques especially through logical formalisms. Examples of declarative conceptual models are CIAM [GUST.82], DADES [OLIV.82], ERAE [DUBO.86], DCM [OLIV.89] , CPL [DIGN.89] and OBLOG [SERN.89].

A declarative specification is mainly composed of:

- facts which may be elementary (they correspond to events) or derivable,
- derivation rules which express what must have happened for a particular fact to hold at a particular time,
- set of constraints which determine valid facts at a given time.

The declarative approach suggests to consider a complete history of the system, and to constraint state and event occurrences. For instance in the OBLOG language, a safety rule which constrains the occurrence of an event on an object has the following form:

{condition} <EVENT>

where {condition} concerns object attributes and past events that have occurred in the object life cycle. In OBLOG, other rules (called derivation rules) allow to derive attribute values after a sequence of events.

Comparative comments on the two approaches

In the following, we briefly comment upon the two approaches in terms of readability of the conceptual schema, facility in constructing the conceptual schema, implementation aspects and strenghts of the declarative approach in conceptual modelling activity.

(i) A conceptual schema constructed with a declarative approach is probably more modular and easier to read in the sense that all events and conditions affecting an entity are defined in a single place. In the operational approach, understanding the behavior of an entity requires looking up all the operations which may affect this entity and analysing them.

(ii) On the other hand, the construction of the conceptual schema is more difficult in the declarative way because of the distribution of events in the conceptual schema. As an event may affect several entities, it should appear in all of them. It may be possible to forget in the modelling process some entities affected by this event. In the operational approach, when an event is defined, all the entities affected by it are considered at the same time.

(iii) It seems that the operational approach is simpler to implement than the declarative one. However some current works seek to demonstrate how a declarative conceptual schema can be translated into an executable specification. The TEMPORA project [TEMP.91] recognizes the role of business rules and visibly maintains these rules throughout the software development process, from requirements specifications through to an executable implementation. The rule based conceptual specification is supported at the database level by an extension of the relational model with temporal semantics and an execution mechanism that provides active database functionality.

(iv) The declarative approach is usually related to languages which have the ability to easily describe by means of rules the appropriate knowledge on some system. One of the major advantages of declarative statements is the separation of business rules declaration from their implementation in application programs, consequently any change to business rules can be reflected in the information system in a more straightforward manner. The conceptual rule language CRL [LOUC.90] allows for example to easily interpret and specify business rules in terms of the objects that exist in the organization.

III.2.2 Techniques and formalisms for validating system behavior specification

Elimination of ambiguity, inconsistency and incompleteness in a conceptual schema is difficult. These problems are discussed in greater detail in [MEYE.85] and [DAVI.79]. Most of the behavioral models make use of tools for checking (sometimes in a automated way) the specification. We outline some of these tools in the following.

Formal conceptual languages

Formality and deductive power of declarative conceptual languages related to the declarative modelling approach provide suitable support for dealing with some specification errors. The languages ERAE [DUBO.86], [DUBO.87], [HAGEL.88] and OBLOG [SERN.88] are good illustrations of how intensively use deduction to validate the requirements.

In ERAE, two techniques are used to detect deficiencies in specifications; the first one consists of deriving new statements from the available ones and to propose them for validation to the end-users; the other one consists of checking the consistency of the requirements when adding a new statement in the specification.

OBLOG uses a logical calculus which provides tools for proving properties of the described society of objects; such properties address inheritance hierarchies and deadlock situations.

Techniques based upon state transitions
These techniques are derived from finite state machines. They consider the system specification as an hypothetical machine that generates a state change in response to an input. These techniques are supported by notations such as state transition diagrams and state transition matrices. Petri-nets are probably the most known technique of this class; they are particularily suitted to describe an intended system behavior where precise process synchronization is required. A survey and comparison of these techniques is given in [DAVI.88].

III.2.3 Dealing with temporal aspects

Time is a very important subject in conceptual modelling because it is intimately tied to the description of dynamic aspects of the world. Temporal models have most often been defined within declarative behavioral models. This results from the nature of the behavioral approaches that suggest to consider the complete history of the system, and to constraint states and events happenings. Conceptual modelling has addressed the problem of time modelling with two main objectives:

- to define time models,
- to use temporal languages in the behavior modelling of systems.

The "time" introduced by L. Anderson [ANDE.82], the set approach of "temporal domains" proposed in [CLIF.87] and "temporal abstractions" defined in [BOLO.83] are signifiant examples of works related to time models. In time modelling, two distinct approaches have been proposed, one based on time points [DERM.82] and the other one founded on the notion of "time periods" [ALLE.81], [ALLE.84], [VILA.82].

[LOUC.91], [HAGE.88] and [ELMA.90] have proposed some extensions of the Entity / Relationship model in which time is introduced as a distinguished entity class. The THM model [SCHI.83] is an extension of a semantic model with time concepts.

Complementarily to temporal concepts, new formal languages with temporal operators are used to constraint the event occurrences and the chronological order of states. The following operators seem to be adopted by various formal languages used in conceptual modelling : ERAE [DUBO.86], TEMPORA [TEMP.91], OBLOG [SERN.89] and CPL [DIGN.89]:

(O f) means that f is true in the next state,
(V f) means that f is true or will become true in some future state,
(⌊ f) means that f is true and will remain true in all future states,
(f U y) means that f will remain true until y becomes true, or forever if y never becomes true.

There are also four corresponding operators referring to the past.

In conclusion, we have argued that researches in conceptual modelling during the last decade have been focussed on the definition of conceptual models with suitable concepts, having well defined semantics and which, taken together, provide a powerful mean of expression. We have also indicated the limitations of these models in particular in terms of integration of static and dynamic aspects modelling, formality of foundations and ease of use. One emerging evolution of conceptual models and languages seems to combine the main contributions of the approaches described above with the advantages of the object-oriented paradigm. This topic is discussed in the next section.

IV. THE INFLUENCE OF THE OBJECT-ORIENTED APPROACH ON CONCEPTUAL MODELLING

In the past, programming languages, artificial intelligence and databases areas have contributed independently to the development of conceptual modelling. Further advances in conceptual modelling require the integration of the concepts, tools and techniques that were developed in these fields for system specification. Database design typically emphasized static properties, extracting all structural properties from all programs that would access the database, resulting in a schema to represent these properties. This approach radically differs from the programming languages view that emphasizes dynamics. It is now clear that both structural and behavioral properties of application objects must be not only designed but also integrated. The object-oriented approach places equal emphasis on structure and behavior modelling in order to achieve an integrated result. The new emerging **object paradigm** might be an **unifying approach** which can contribute to conceptual modelling evolution.

IV.1 THE EMERGING FIELD OF OBJECT-ORIENTED MODELLING

Push for object-oriented techniques and concepts

Conceptual models and languages should be influenced in the future (as they were in the past) by the research areas of databases, programming languages and artificial intelligence. In these three areas, the object-oriented approach has matured enough during the past years. In programming languages, since SIMULA-67, a large variety of languages have been developed either as extensions of existing languages (C++, Objective C, LOOPS...) or as new object-oriented programming languages (Eiffel, Smalltalk...). Introduction of frames [MINS.75] in the field of artificial intelligence has lead to the development of frame-based knowledge representation languages such as KEE, SRL and KRS. Lastly, object-oriented database systems [LECL.88], [KIM.90] are characterized by richer data models (definition of a database schema in terms of objects, support for definition and management of complex objects and class hierarchy...). All researches and developments in these areas converge to similar concepts: object encapsulation and information hiding, object hierarchy and inheritance, object communication through messages. A detailed presentation of these generic concepts can be found in [MEYE.88]. We believe that object-orientation is influencing conceptual modelling with new modelling concepts (such as object life cycle), new modelling techniques (such as reuse) and new models of conceptual modelling process (such as the fountain model). However, most of the underlying principles of object orientation address the design process, it is necessary to refine their semantics i.e make them relevant in the conceptual modelling area.

Object-oriented modelling approaches

The emergence of object-oriented concepts in the earlier phases of systems development has resulted in a number of specific approaches with a multiplicity of ideas [HEND.90]. Approaches diverge on how to perform object-oriented modelling. To summarize, we consider three major ones:

- **The functional approach** recommends a primary specification based upon system functions and dataflows which is then transferred to an object view at the design stage. This approach has lead to studies on transformation of dataflows analysis models into object-oriented design [ALAB.88], [BAIL.89], [WARD.89]. The functional approach is adopted by several authors who consider object orientation as only relevant for a part of the system life cycle. Thus, object-oriented design required to be preceeded with a methodology for requirements engineering. Booch [BOOC.87] for instance, recommends the JSD methodology [JACK.83] for providing a system specification as an input of his object-oriented design methodology. Another fashion to combine the functional approach and the object-oriented approach is to mix the two approaches [BAIL.89], [WARD.89] at the requirement analysis phase. An in depth analysis of such "mixed" approaches is made in [WIER.91]; the author concludes that the two paradigms are incompatible and should not be used in a single modelling effort due to their too different modularization principles and diagrammatic techniques.

- **The data approach** is strongly influenced by classical conceptual models. It bases the identification and description of objects either on an entity / relationship model [HOZA.89], [SHLA.88], [SHLA.89] or on classical semantic models [RUMB.91], [COAD.90]. It emphasizes the structural view of objects. In addition, this approach often leads to place side by side several conceptual schemas without expliciting relationships between them. For instance the object modelling technique (OMT) [RUM.91] consists of three kinds of models : the object model (which describes the static structure in terms of objects and relationships), the dynamic model (which describes the control structure in terms of events and states) and the functional model (which describes the computational structure in terms of values and functions). In this approach, the object concept is very close to the one of entity.

- **The object-centered approach** makes use of object-oriented concepts since the beginning of the modelling process. The resulting conceptual schema is described as a community (also called society or collection) of objects that can interact with each other. The use of the same concepts for both requirements analysis and system design is seen by many [LADD.89], [SEID.89] as one of the strenghts of an object-oriented approach.

The object concept is considered as a **modelling concept** which can be used for representing any kind of relevant element of a system. Thus, this approach makes necessary an object classification for facilitating object identification. The OBLOG model [COST.89] distinguishes active objects (which have the capability of initiating events on others objects) and passive objects (which evolve only by sharing events with active objects). J. Iivari [IIVA.91] proposes another classification which includes both persistent objects (entity types), transient objects (events), user objects (actors or units) and so on.

In this approach, the object concept is also a **knowledge unit** which describes an object in three perspectives:

* the static perspective (data or structure perspective) is centered on objects attributes and structural relationships among objects,

* the behavior perspective (or dynamic perspective) is specified by the object life cycle. It includes the definition of events which may occur in the object life,

* the process perspective focusses on the dynamic and temporal relationships between objects, typically this perspective deals with triggering of operations when some event occurs and with synchronization rules between events.
A few papers address the integration of these perspectives [RAMA.91], [SERN.91]. We believe however that this approach is the only one which could contribute to a major improvement of conceptual modelling practices; thus we concentrate in the following on this approach.

IV.2 REQUIREMENTS AND RESEARCH TOPICS FOR OBJECT-ORIENTED MODELLING

We stress in the following the features required by a centered object-oriented approach for constituting a step forward in conceptual modelling.

Study of semantic relationships between objects
Some authors argue that relationships violate the encapsulation principle and that they should be attached to a single object. It is the authors' view that this approach leads to hide complex situations and dependencies which could imply program failures. We believe that conceptual modelling must emphasize the analysis and description of relationships among objects to show how the information they contain depends on two or more objects [RUMB.87]. Following this line, some oriented modelling approaches have proposed refinements to solutions investigated within semantic models:

- **Aggregation refinement**. Several authors [ESSI.91], [BRUN.91] propose to make a distinction between aggregates with components that are existentially dependent on the aggregate (this kind of aggregation is named composition in [BRUN.91], containment in [ESSI.91], and has-component relationship in [KAPP.90]) and aggregates that have existentially independent parts (this kind of relationship is named assembly in [ESSI.91], relationship in [KAPP.90] and reference in [BRUN.91]).

- **Generalization / specialization refinement**. The "IS-A" relationship is used in most object-oriented models. Since its introduction in semantic models [SMIT.77b], several interpretations have been proposed. In conceptual modelling, two important forms of generalization are relevant. The most usual form is used to express an **"IS-A" relationship** between a generic class and a specialized class. According to this view, models interpret generalization in terms of a subset relationship between object classes : "class A IS_A class B" means that every instance of A is also an instance of B. Viewing the classes EMPLOYEE and STUDENT as one generic class PERSON means that any employee and any student is a person. Some authors have introduced a **notion of role** which is used when an object belongs to the specialized class for a limited period of its life cycle. For instance, a person may be considered in the role of referee for a certain time in his life but not during his entire life. These two forms of generalization are introduced and used in a complementary way in [ESSI.91], [ELMA.90], [PERN.90a] and in [KAPP.90].
The IS_A relationship is always related to an **inheritance mechanism** which states that properties of the generic class are inherited by the specialized classes. In some models, inherited properties can be overriden at the specialized class level. Overriding is a common practice in object-oriented programming languages. It can occur on a feature (attribute or method) of a generic class, even without preserving its semantics. We believe that this practice can lead to conceptual confusion and we argue for overriding mechanisms which preserve the semantics of the original feature. In

this perspective, a restrictive form of overriding is proposed in some object-oriented models; for instance [LOOM.87] proposes to use overriding by augmentation i.e the new method differs from the generic one by adding some particular action affecting specific attributes of the specialized class. [SERN.90] proposes a formalization of the IS_A relationship based on object morphisms [EHRI.89], [EHRI.90] with a strict ("monotonic") form of overriding: an overriden operation (named event) may have further effects, but the effects on the inherited attributes must remain the same.

Adaptation of the localization principle

Semantic models have contributed to the practice of abstraction by defining mechanisms such as classification, aggregation, generalization and association. The object-oriented approach provides a new useful abstraction form which encourages the description of objects in isolation from others. This abstraction principle called localization [BROD.86b] contributes to the achievement of modularity in conceptual specifications. Modularity is one of the key programming concepts not yet well established in conceptual modelling. The principle allows the modeler to reason locally on a single object, making abstraction of its relationships with other objects. A large variety of object-oriented models supports this principle by means of object structure and object life cycle concepts. Typically the **object structure** consists of a set of attributes and the state of an object is defined by attribute values; attributes can be related to static integrity constraints which restrict the set of possible attribute values of objects. The **object life cycle** consists of a set of states, a set of events that cause state changes and a set of transition constraints that restricts the succession of event occurrences. The object life cycle of an object makes the behavior of an object explicit and localized. This approach must be distinguished from classical behavioral models where the life cycle of an object is spread throughout the conceptual schema in terms of operation conditions.
A great number of object-oriented models suggests to use the concept of object life cycle [BRUN.91], [KAPP.90], [ESSI.91], [SERN.91]; most of these models propose diagrammatic techniques to depict it. However, the current models differ on the event notion. In [KAPP.90] an event (named activity) is considered as an operation which may be primitive or not. Non primitive activities are complex operations which may be depicted according to a functional decomposition. In [SER.91] an event is an atomic operation which causes a attribute state change of only one object. Research efforts remain to be done to clarify the notions of event and object life cycle, to understand the conceptual implications of the localization principle and its contribution to reuse of conceptual objects librairies.

Modelling of object communication

Objects interact dynamically as well as statically. Modelling dynamic relationships among objects in an object-oriented approach for constructing a conceptual schema is a current research topic. The object life cycle addresses the problem of the events that change the state of only one object. A more difficult problem occurs when an event involves state changes of several objects. It is the authors' belief that the message passing mechanism used in classical object-oriented programming languages is not powerful enough to conceptually express the causal interactions between objects. It leads to embed dynamic and temporal (synchronization of operations on objects, propagation of operations on objects) relationships in operation specifications. Object-oriented conceptual models must propose richer mechanisms for modelling communication between objects. The **triggering mechanism** is explored in [WIRF.90] and used in [KAPP.90] and [BRUN.91] to initiate simultaneous state changes of several objects. Another way to model synchronization is to let **events be shared by several objects**; this approach is proposed for example in OBLOG [COST.89]. Complementary to the triggering mechanism, [KAPP.90] introduces another mechanism (called hierarchical synchronization) for **scheduling the life cycle** of an object with the

life cycles of its components. Determining an appropriate set of mechanisms to describe, at the conceptual level, object communication is a current research topic.

Normalization of object-oriented conceptual schemas

The normalization theory related to the relational model has proved its usefulness in designing databases [CODD.72]. The normalization guidelines can be regarded not only as a discipline by which the designer can capture a part of the semantics of the real world but also as a means to recognize undesirable relations and to convert them into a more desirable form. We believe that a suitable object-oriented model should provide such guidelines in order to support the design of "well-defined" conceptual schemas. Let's notice that models [WIRF.89], [HEIT.87] dealing with object-oriented design start to propose criteria (such as minimizing the complexity of a client / server graph) to evaluate the quality of a specification and optimization rules (such as increase as possible the number of clients per object) for helping the designer in constructing "well-defined" software architecture.

Object-oriented models for conceptual modelling have not yet delt with the normalization principle. None of these models provides means to prove that the conceptual schema carries out desirable properties such as completeness, nonredundancy...Furthermore by using these models, uniqueness is not achieved (there exist several ways to represent the same thing) and it is impossible to evaluate and compare the different results. A research effort should be done in defining "quality criteria" for an object-oriented conceptual schema and guidelines for achieving a suitable result.

Object grouping mechanisms

This research topic addresses the definition of objects "packaging" mechanisms. The purpose is to define an intermediate unit between an entire conceptual schema and the basic building block of object allowing to view the conceptual schema contents at different levels of detail. Grouping mechanisms can be defined with distinct purposes:

- to facilitate a layered system analysis; they are then, near to decomposition mechanisms and top down approaches used in traditional structured analysis. In [CHAM.89], a technique (named ensemble) is explored for creating and analysing objects in an ordered fashion. Ensembles like objects can be described by attributes, can interact with other objects and other ensembles. A major difference between ensembles and objects is that an ensemble has internal parallelism, while an object is a finite state machine.

- to allow reuse at the analysis / design levels. Some authors argue that the class concept (although used for code reuse) is a too fine unit for achieving specification reuse. Proposition for good unit of reuse are the concepts of framework [WIRF.90], [BECK.91] and module [ADIB.90].

- to allow to hide details of a set of classes that are irrelevant outside the group. [RUMB.91] introduces the notion of module as a logical grouping construct which captures one perspective or view of a situation; however the bounderies of a module are arbitrary. The OOA method [COAD.90] allows, once the conceptual schema is built, to present it in five major layers; each layer corresponds to some aspects (attributes or services or hierarchy of objects) of all objects within the conceptual schema.

Conceptual modelling for distributed systems

In order to take into account the future computer applications based on distributed architectures, conceptual modelling must address the problem of cooperation of information systems. Two approaches can be explored at the conceptual level:

- the integrated approach which consists of a unique schema to which all subsystems have to conform. This approach is related to distributed database systems [CERI.84] and seems to raise two major problems: it may be complex and even impossible to maintain consistency in distributed

systems when the number of local systems is important; secondly, this approach may be extremely expensive and time consuming due to the fact that requests on local systems must also pass through the global schema [GARD.90],

- the federated approach [HEIM.85] in which each local system has its own schema and is responsible for controlling its interactions with other systems by deciding which information to import and export. This approach is related to the idea of open systems [HEWI.86].

V. PERSPECTIVES IN CONCEPTUAL MODELLING PROCESS

To improve the conceptual modelling activity in the system's life cycle, it is necessary not only to substantially refine modelling formalisms, but also to produce better tools which make use of automated knowledge engineering techniques.

Due to the creative nature of the conceptual modelling activity, the current CASE tools lack in providing intelligent support for this activity. Today's CASE tools only concentrate on supporting the management of the conceptual schema. They assist the designer in drawing the conceptual schema in a diagrammatic form, in storing its contents in a repository and in documenting it. They do not help him in constructing the conceptual schema itself. They do not support the conceptual modelling tasks such as knowledge acquisition or conceptualization. The main areas where improvements are in need are:

- support to capture user requirements and to produce a conceptual schema, abstracting and conceptualization from them,
- validation support for checking whether the conceptual schema is consistent and whether it correctly expresses the requirements informally stated by the users community,
- understanding and formalizing the conceptual modelling process in order to support it in an automated way.

V.1 KNOWLEDGE ACQUISITION SUPPORT

Natural language based knowledge acquisition

Conceptual modelling requires knowledge acquisition about some application domain. Such knowledge is often expressed in natural language, in particular when interwieving people working with the application. One approach to start the modelling process is to generate a conceptual schema from the description of the application domain with natural language sentences. Various techniques are experienced for abstracting from narrative texts and a few prototypes have been developed following this approach. AMADEUS [BLAC.87] is one example which aims at combining graphics and natural language for conceptual modelling. Other examples are OICSI [CAUV.88] and ACME [KERS.86] which are conceptual modelling expert systems.

Knowledge acquisition by forms

Forms are widely used for communication of information in many organizations. Existing documents constitute an important input of the modelling process. Due to the fact that most forms are structured in similar ways, the process of deriving knowledge from forms can be partly automated. Thus, the idea comes to extract the knowledge from examples of forms and to translate

it into elements of the conceptual schema. This approach has been investigated in [MANN.88], [CHOO.88] and extended by [TALL.90].

Knowledge acquisition by reusing specifications

Another way to acquire knowledge about an application domain is to reuse specifications of existing information systems of that domain. In systems development reuse is limited to routines and run-time libraries. However, to increase the effectiveness of software reuse, an extended view of reusability is needed where reusable libraries store both reusable elements and their associated development information, such as design structures, design transformations and documentation [DAHR.88]. It seems that the degree of reuse is dependent of the approach used in designing reusable components. Several projects have emphasized the advantage of using the encapsulation and inheritance mechanisms of the object-oriented paradigm. An interesting methodology to guide the application designer in the design of applications based on reusable objects is presented in [PERN.90b]. The methodology supports object-oriented requirements specifications, it drives the application designer to derive requirements specifications through guidelines based on the knowledge of a given application domain derived from earlier similar applications. These guidelines are stored in the repository with reusable classes. A prototype (REquirements Collection And Specification Tool: RECAST) based on this approach has been realized within the Ithaca project [FUGI90], [PROE.89]. Others authors have proposed repositories to store reusable components [JOHN.90].

Another way based on analogy as a means for reusing specifications from a CASE repository has been experienced by Maiden and Sutcliffe [MAID.91], [SUTC.91]. They tackle the retrieval problem and the customisation problem of reusable conceptual components and propose an Intelligent Reuse Advisor based on cognitive models of specifications reuse and analogous reasoning. Similarly [GROS.91] has shown the existence of generic modelling patterns that can be reused and tailored to a specific project. This approach is similar to the one developed by Renbenstein et Waters [RENB.91] for providing an automated assistance for requirements acquisition based on the notion of "cliché".

Knowledge acquisition by reverse engineering

Lastly, application domain knowledge may be found in existing information systems already automated. The purpose of reverse engineering is to recapture the information lost in the process of coding application with a database or programming language. Reverse engineering is defined in [CHIK.90] as a process of analysing a subject system to identify the system's components and their relationships and to create representations of the system in another form or at a higher level of abstraction. Reverse engineering can be applied all over the development stages; starting from the existing implementation, it leads to recapture or recreate the design and then to decipher the requirements actually implemented in the system. Reverse engineering techniques add to the examination of the existing system domain knowledge, user input and fuzzy reasoning in order to obtain a semantic conceptual schema [CHIK.90], [BIGG.89]. The problem of abstracting from information system structures to a conceptual schema level has been addressed in the Abstract Conceptual Schema (ACS) project which has produced theoritical mappings from Cobol to ACS and in the MAP project which abstracts from the relational level to the conceptual schema level. Reverse engineering methods and tools combined with CASE environments will provide another means for acquiring more knowledge upon users requirements.

V.2 VALIDATION SUPPORT

Validation by paraphrasing the conceptual schema

During the development of a conceptual schema, it is necessary to perform schema validation to see if the conceptual schema correctly describes the application domain. Paraphrasing the conceptual schema in natural language makes easier to understand the semantics and consequently to perform validation. This validation technique may be useful if the conceptual model is formal and if users are not familar with that conceptual formalism. Paraphrasing conceptual schema to natural sentences is an important research topic. Techniques that could be adapted to this area have been proposed in the database field, see for example [MUCK.85] and [LUK.86]).

Validation by diagnosing the conceptual schema

The modelling activity achieves from the domain informal knowledge a conceptual schema which must be defined and specified by the use of formal modelling concepts. Formality and deductive power of the conceptual model allow some automatic diagnosis on the schema. Usually, schema diagnosis is based on different kinds of rules: syntax rules, semantic rules, quality detecting rules (based on general methodological aspects or on expert's knowledge). [WOHE.88] and [CAUV.88] provide some examples of rules that are part of the knowledge base of an expert system.

Knowledge about the application domain can be used in diagnosis of conceptual schemas to judge whether an application specification is compatible with the domain. Domain dependent rules imply the use of some general approach to represent and handle domain dependent knowledge. This approach is investigated in [WOHE.88].

V.3 PROCESS SUPPORT

The modelling process is intended to monitor the different cooperative tasks employed in delivering the conceptual schema. Monitoring is required to organize, control and select the appropriate modelling tasks in a certain situation during the process. In most CASE tools the modelling process is entirely predefined, that is the designer can enter upon a task only if the previous one is itself completed. The modelling process is viewed as a set of tasks executed according to a predefined linear sequence. A CASE tool which really supports monitoring of modelling tasks should be able to automatically propose the designer the better action to perform in each situation reached during the process. Development of such tools involves the definition of new mechanisms such as tracing mechanisms and explanation mechanisms. This requires also the definition of new process models. Tentatives for over-coming the limitations of the waterfall model can be classified [DOWS.87] in activity-oriented models, product-oriented models and decision-oriented models. The spiral model [BOEH.88] revised by [IIVA.91] and the fountain model [HEND.90] are examples of the former ones. View points [FINK.90] and design product graphs [RUEH.88] follow the product-oriented view of process modelling. Examples of the latter models are the DAIDA one [JARK.90] and Potts's one [POTT.88], [POTT.89]. Research effort must be done in understanding and formalizing the conceptual modelling process in order to support it an automated way.

VI. CONCLUSION

In this paper, we have discussed the main trends and perspectives in the area of conceptual modelling. Conceptual modelling has much advanced in the past several years. The push for object-oriented techniques and concepts is reaching this area involving the emergence of object-oriented modelling approaches with a multiplicity of ideas. On the basis of a reflection on the history of the evolution of conceptual modelling on the one hand, and on the requirements of conceptual modelling for future information systems, on the other hand, we addressed several research topics

which have not been well explored. The paper outlined the contrast between the great evolution of conceptual models for specifying the contents of a conceptual schema and the little attention payed to the conceptual modelling process. Development of advanced and automated support for controlling the process requires its formalization. In this perspective, the research of process models is a major topic that researchers might take up.

REFERENCES

[ABRI.74] : J.R. ABRIAL, "Data Semantics", In J.W. Klimbie and K.L. Koffeman (eds.), Data Management Systems, North-Holland, Amsterdam, The Netherlands, 1974.

[ADIB.90] : M. ADIBA, F. BRISSAUD, P. DECHAMBOUX, B. DEFUNDE, F. EXERTIER, J. GIRAUDIN, S. HOUSSIN, C. LENNER, "Base Model for ARISTOTE", ARISTOTE Report - RAP006, Grenoble, September 1990 (in French).

[ALAB.88] : B. ALABISO, "Transformation of data flow analysis models to object-oriented design", In Proc. of OOPSLA'88, San Diego, California, 1988.

[ALLE.81] : J.F. ALLEN, "An Initial-based Representation of Temporal Knowledge", 7th Int. Conf. on Artificial Intelligence, Vancover, Canada, 1981.

[ALLE.84] : J.F. ALLEN, "Towards a General Theory of Action and Time", Artificial Intelligence, Vol 83, No 2, 1984.

[ANDE.82] : T.L. ANDERSON, "Modelling Time at the Conceptual Level", Improving Database Usuability and Responsiveness, P. Scheuermann (Ed.), Academic Press, 1982.

[ANSI.75] : "Study Group on Data Base Management Systems: Interim Report 75-02-08", In ACM SIGMOD Newsletter, FDT, Vol 7, No 2, 1975.

[BECK.91] : K. BECKER, F. BODART, "Reusable Object-Oriented Specifications for Decision Support Systems", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[BIGG.89] : T.J. BIGGERSTAFF, "Design Recovery for Maintenance and reuse", Computer, July 1989.

[BAIL89] : S.C. BAILIN, "An Object-Oriented Requirements Specification Method", Comm. ACM Vol 32, No 5, 1989.

[BLAC.87] : W.J. BLACK, "Acquisition of conceptual Data Models from Natural Language Descriptions", In 3rd Conference of the European Chapter of ACM", Copenhague, Danemark, 1987.

[BOEH.88] : B.W. BOEHM.88], "A Spiral Model of Software Development and Enhancement", Computer Journal, Vol 21, 1988.

[BOLO.83] : A. BOLOUR, L.J. DEKEYSER, "Abstractions in Temporal Information", In Information Systems, Vol 8, No 1, 1983.

[BOOC.87] : G. BOOCH, "Software Engineering with ADA", Benjamin/Cummings, California, 1987.

[BORG.85] : A. BORGIBA, S. GREENSPAN, J. MYLOPOULOS, "Knowledge Representation as the Basis for Requirements Specification", IEEE Computer, April 1985.

[BROD.82] : M.L. BRODIE, E. SILVA, "Active and Passive Component Modelling: ACM/PCM", in IFIP WG8.1 Working Conference on Information System Design Methodologies : a Comparative Review, T.W. Olle, H.G. Sol, A.A. Verryn-Stuart (eds), North-Holland (pub), 1982.

[BROD.86a] : M. L. BRODIE, J. MYLOPOULOS, J. W. SCHMIDT, "On Conceptual Modelling: Perpectives from Artificial Intelligence, Databases and Programming Languages", M.L. Brodie, J. Mylopoulos and J.W. Schmidt (eds), Springer-Verlag (pub), Harrisonburg, Virginia, 1986.

[BROD.86b] : M.L. BRODIE, "On the Development of Data Models", In "On Conceptual Modelling - Perpectives from Artificial Intelligence, Databases and Programming Languages", M.L. Brodie, J. Mylopoulos and J.W. Schmidt (eds), Springer-Verlag(pub), Harrisonburg, Virginia, 1986.

[BRUN.91] : J. BRUNET, "Modeling the World with Semantic Objects", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[BUBE.86] : J.A. BUBENKO, "Information System Methodologies - A Research Review", SYSLAB Report 40, Department of Computer and Systems Science, University of Stockholm, Sweden, 1986.

[CAUV.88] : C. CAUVET, C. PROIX, C. ROLLAND, "Information Systems Design: an Expert System Approach", In Artificial Intelligence in Databases and Information Systems, R.A. Meersman, Z. Shi, H. Kung (eds), North-Holland(pub), 1988.

[CERI.84] : S. CERI, G. PELAGATTI, "Distributed Databases - Principles and Systems", McGraw-Hill, New York, 1984

[CHAM.89] : D. De CHAMPEAUX, W. OLTHOFF, "Towards an Object-Oriented Analysis Technique", In Proc. of the Pacific North-west Software Quality Conference, September 1989.

[CHEN.76] : P.P.S. CHEN, "The Entity-Relationship Model: Toward a unified view of data", ACM Transactions on Database Systems, Vol 1, No 1, March 1976.

[CHIK.90] : E.J. CHIKOFSKY, J.H. CROSS, "Reverse Engineering and Design Recovery: A Taxonomy", IEEE Software, January 1990.

[CHOO.88] : J. CHOOBINEH, M. MANNINO, J. NUNAMAKER, B. KONSYNSKY, "An Expert Database Design System Based on Analysis of Forms", IEEE Trans. on Software Engineering, Vol 14, No 2, 1988.

[CLIF.87] : J. CLIFFORD, A. RAU, "A Simple General Structure for Temporal Domains", Temporal Aspects in Information Systems, Rolland (Ed.), North-Holland, 1987.

[COAD.90] : P. COAD, E. YOURDON, "Object-Oriented Analysis", Yourdon Press, Prentice Hall, Englewood Cliffs, NJ, 1980.

[CODD.72] : E.F. CODD, "Further Normalization of the Data Base Relational Model", In Data Base Systems, Courant Computer Science Symposia Series, Vol 6, Englewood Cliffs N.J. Prentice-Hall, 1972.

[CODD.79] : E.F. CODD, "Extending the Database Relational Model to Capture More Meaning", ACM Trans. on Database Systems, Vol 4, No 4, December 1979.

[COST.89] : J.F. COSTA, A. SERNADAS, C. SERNADAS, "OBL-89, User's Manual, Version 2.3", Instituto Superior Técnico, Lisbon, May 1989.

[DAHR.88] : V. DAHR, M. JARKE, "Dependency Directed Learning in System Maintenance Support", IEEE TSE, Vol 14, No 2, 1988.

[DATE.83] : C.J. DATE, "An Introduction to Database Systems", Vol 2, Addison-Wesley Publishing Company, 1983.

[DATE.86] : C.J. DATE, "An Introduction to Database Systems", 4th Ed., Vol 1 Addison-Wesley Publishing Company, 1986.

[DAVI.79] : A.M. DAVIS, T. RAUSCHER, "Formal Techniques and Automatic Processing to Ensure Correctness in Requirements Specifications", In Proc. of the IEEE Specifications of Reliable Software Conference, Cambridge, Mass. IEEE Press, Wash., D.C. 1979.

[DAVI.88] : A.M. DAVIS, E.H. BERSOFF, E.R. COMER, "A Strategy for Comparing Alternative Software Development Life Cycle Models", IEEE Transactions on Software Engineering, October 1988.

[DERM.82] : D. McDERMOTT, "A Temporal Logic for reasoning about processes and plans", Cognitive Science, Vol 6, 1982.

[DIGN.89] : F. DIGNUM, "A Language for Modelling Knowledge Bases", PhD Thesis, Department of Mathematics and Computer Science, Vriji Universiteit, 1989.

[DUBO.86] : E. DUBOIS, J. HAGELSTEIN, E. LAHOU, A. RIFAUT, F. WILLIAMS, "A Data Model for Requirements Engineering", 2nd Int. Conf. on Data Engineering, Los Angeles, 1986.

[DUBO.87] : E. DUBOIS, J. HAGELSTEIN, "Reasoning on Formal Requirements: A Lift Control System", 4th Int. Workshop on Software Specification and Design, Monterey, USA, 1987.

[DUBO.89] : E. DUBOIS, J. HAGELSTEIN, A. RIFAUT, "Formal Requirements Engineering with ERAE", Philips Journal of Research, Vol 43, No 4, 1989.

[EHRI.89] : H.D. EHRICH, A. SERNADAS, C. SERNADAS, "Objects, Object Types and Object Identity", In Categorical Methods in Computer Science with Aspects from Topology, H. Ehrig et al (Eds), LNCS 393, Springer Verlag, 1989.

[EHRI.90] : H.D. EHRICH, A. SERNADAS, C. SERNADAS, "From Data Types to Object Types", In Journal of Information Processing and Cybernetics, EIK 26, 1990.

[ELMA.90] : R. ELMASRY, I. EL-ASSAL, V. KOURAMAJIAN, "Semantics of Temporal Data in an Extended ER Model", 9th Int. Conference of Entity-Relationship Approach, Lausanne, October 1990.

[ESSI.91] : L.J.B. ESSINK, W.J. ERHART, "Object-Modelling and System Dynamics in the Conceptualization Stages of Information Systems Development", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[FINK.90] : A. FINKELSTEIN, J. KRAMER, M. GOEDICKE, "Viewpoint Oriented Software Development", Proc. of Conf. on "Le Genie Logiciel et ses Applications", Toulouse, 1990.

[FUGI.90] : M.G. FUGINI, B. PERNICI, "RECAST: A Tool for Reusing Requirements", Conf. CAISE'90, Stockholm, May, 1990.

[GARD.90] : G. GARDARIN, P. VALDURIEZ, "SGBD avancées, Bases de données objets, déductives, réparties", Eyrolles, 1990 (in French).

[GROS.91] : G. GROSZ, C. ROLLAND, "Why and How Should We Hide Conceptual Models ?", Proc. of the Conference on Software Engineering and Knowledge Engineering (SEKE'91), Skokie, Illinois, Etats Unis, Juin 1991.

[GUST.82] : M.R. GUSTAFSSON, T. KARLSSON, J.A. BUBENKO, "A Declarative Approach to Conceptual Information Modelling", in IFIP WG8.1 Working Conference on Information System Design Methodologies : a Comparative Review, T.W. Olle, H.G. Sol, A.A. Verryn-Stuart (eds), North-Holland (pub), 1982.

[HAGE.88] : J. HAGELSTEIN, "Declarative Approach to Information Systems Requirements", in Knowledge-Based Systems, Vol 1, No 4, September 1988.

[HAMM.81] : M.M. HAMMER, D.J. McLEOD, "Database Description with SDM: A Semantic Database Model", ACM Trans. on Database Systems, Vol 6, No 3, 1981.

[**HEIM.85]** : D. HEIMBIGNER, D. McLEOD, "A Federated Architecture for Information Management", ACM Trans. on Office Information Systems, Vol 3, No 3, September 1985.

[HEIT.87] : M. HEITZ, "HOOD, une méthode de conception hiérarchisée orientée objets pour le développement de gros logiciels techniques et temps réel", In Journées ADA France (BIGRE N5), 1987.

[HEND.90] : B. HENDERSON-SELLERS, J.M. EDWARDS, "The Object-Oriented Systems Life Cycle", Communications of the ACM, September 1990.

[HEWI.86] : C. HEWITT, P.De JONG, "Open Systems", In "On Conceptual Modelling - Perpectives from Artificial Intelligence, Databases and Programming Languages", M.L. Brodie, J. Mylopoulos and J.W. Schmidt (eds), Springer-Verlag (pub), Harrisonburg, Virginia, 1986.

[HOZA.89] : B.J. HOZA, M.K. SMITH, S.R. TOCKEY, "An Introduction to Object-Oriented Analysis", In Proc. of the 5th Structured Techniques Association Conference, Chicago, May 1989.

[HULL.87a] : R. HULL,"A Survey on Theoritical Research on Typed Complex Database Objects", In Databases, Ed. J. Paredaens, Academic Press, London, 1987.

[HULL.87b] : R. HULL, R. KING, "Semantic Database Modelling: Survey, Applications and Research Issues", ACM Computing Surveys, Vol 19, No 3, 1987.

[ISO.82] : "Concepts and Terminology for the Conceptual Schema and the Information Base", Report No 695, ISO/TC9/SC5/WG3, 1982.

[IIVA.91] : J. IIVARI, "Object-Oriented Design of Information Systems: The Design Process", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[JACK.83] : M.A. JACKSON, "System Development", Prentice Hall International, 1983.

[JARK.90] : M. JARKE, M. JEUSFELD, T. ROSE, "A Software Process Data Model for Knowledge Engineering in Information Systems", Information Systems, Vol 15, No 1, 1990.

[JOHN.90] : W.L. JOHNSON, M. FEATHER, "Building an Evolution Transformation Library", Proc. 12th Int. Conf. on Software Engineering, Nice, March 1990.

[KAPP.90] : G. KAPPEL, M. SCHREFL, "Using an Object-Oriented Diagram Technique for the Design of Information Systems", In Proc. of the Int. Working Conf. on Dynamic Modelling of Information Systems, Noordwijkerhout, The Netherlands, April 1990.

[KENT.79] : W. KENT, "Limitations of Record-based Informations Models", ACM Trans. Databases Systems 4, 1, March 1979.

[KERS.86] : M.L. KERSTEN, H. WEIGAND, D. DIGNUM, J. BOOM, "A Conceptual Modelling Expert System", In 5th International Conference on the ER Approach", Spaccapietra (Ed), Dijon, 1986.

[**KIM.90]** : W. KIM, et al., "Architecture of the ORION Next-Generation Database Design", IEEE Trans. on Knowledge and Data Engineering, March 1990.

[LADD.89] : R.M. LADDEN, "A Survey of Issues to be considered in the Development of an Object-Oriented Development Methodology for ADA", ADA Letters, 2, 1989.

[LECL.88] : C. LECLUSE, P. RICHARD, F. VELEZ, "O2, an Object-Oriented Data Model", In Proc. ACM SIGMOD, Int. Conf. on Management of Data, Chicago, June 1988.

[LOOM.87] : M.E.S. LOOMIS, A.V. SHAH, J.E. RUMBAUGH, "An Object Modeling Technique for Conceptual Design", European Conference on Object-Oriented Programming, Paris, June 1987.

[LOUC.90] : P. LOUCOPOULOS, P. McBRIEN, U. PERSSON, F. SCHUMMAKER, P. VASEY, "TEMPORA - Integrating Database Technology, Rule Based Systems and Temporal Reasoning for Effective Software", In Proc. of ESPRIT'90 Conference, Brussels, November 1990.

[LOUC.91] : P. LOUCOPOULOS, B. THEODOULIDIS, D. PANTAZIS, "Business Rules Modelling: Conceptual Modelling and Object-oriented Specifications", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[LUK.86] : W.S. LUK, S. KLOSTER, "ELFS : English Language from SQL", ACM Transactions on Database Systems, Vol 11, No 4, 1986.

[MAID.91] : N.A.M. MAIDEN, A.G. SUTCLIFFE, "Specification Reuse by Analogy", Software Engineering Journal 6(1), pp 3 - 15, 1991.

[MANN.88] : M.V. MANNINO, V.P. TSENG, "Inferring Database Requirements from Examples in Form", In 7th Int. Conf. on Entity-Relationship Approach, Batini (ed.), North-Holland, 1988.

[MEYE.85] : B. MEYER, "On Formalisms in Specifications", IEEE Software, 1985.

[MEYE.88] : B. MEYER "Object-Oriented Software Construction", Pub. Prentice-Hall International (UK) Ltd, Hernel Hempstead 1988.

[MINS.75] : M. MINSKY, "A Framework for Representing Knowledge", In P. Winston (ed.), The Psychology of Computer Vision, McGraw-Hill, New York, 1975.

[MUCK.85] : E.M. MUCKTEIN, M.G. DATAVOSKY, "Semantic Interpretation of a Database Query Language", Data and Knowledge Engineering, Vol 1, No 1, 1985.

[MYLO.80] : J. MYLOPOULOS, P.A. BERNSTEIN, H.K.T. WONG, "A Language Facility for Designing Databases Intensive Applications", ACM Trans. on Database Systems, Vol 15, No 2, 1980.

[OLIV.82] : A. OLIVE, "DADES: A Methodology for Specification and Design of Information Systems" in IFIP WG8.1 Working Conference on Information System Design Methodologies : a Comparative Review, T.W. Olle, H.G. Sol, A.A. Verryn-Stuart (eds), North-Holland (pub), 1982.

[OLIV.86] : A. OLIVE, "A Comparative of the Operational and Deductive Approaches to Conceptual Information Systems Modelling", Information Processing 86, Elsevier Science Pub., North-Holland, 1986.

[OLIV.89] : A. OLIVE, "On the Design and Implementation of Information Systems from Deductive Conceptual Models", 15th Int. Conf. on VLDB, Amsterdam, 1989.

[OLLE.88] : T.W. OLLE, J. HAGELSTEIN, I.G. MACDONALD, C. ROLLAND, H.K. SOL, F.J.M. VAN ASSCHE, A.A. VERRIJN-STUART, "Information Systems Methodologies, a Framework for Understanding, Addison Wesley, Workingham, England, 1988.

[PECK.88] : J. PECKHAM, F. MARYANSKI, "Semantic Data Models", ACM Computing Surveys, Vol 20, No 3, September 1988.

[PERN.90a] : B. PERNICI, "Objects with Roles", ACM/IEEE Conference on Office Information Systems, Boston, MA, April 1990.

[PERN.90b] : B. PERNICI, "Class Design and Meta-Design", In D. Tsichritzis, "Object Management", Centre Universitaire d'informatique, Internal Report, Univ. of Geneva, July 1990.

[POTT.88] : C. POTTS, G. BRUNS, "Recording the Reasons for Design Decisions", Proc. 10th Int. Conf. on Software Engineering, 1988.

[POTT.89] : C. POTTS, "A Generic Model for representing Design Methods, Proc. 11th International Conference on Software Engineering, 1989.

[PROE.89] : A.K. PROEFROCK, D.C. TSICHRITZIS, G. MULLER, M. ADER, "ITHACA: An Integrated Toolkit for Highly Advanced Computer Applications", In "Object-Oriented Development", Ed. D.C. Tsichritzis, Centre Universitaire d'informatique, Univ. of Geneva, July 1989.

[RAMA.91] : G.J. RAMACKERS, A.A. VERRIJN-STUART, "Integrating Information System Perspectives with Objects", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[REIT.82] : R. REITER, "Towards a Logical Reconstruction of Relational Database Theory", In "On Conceptual Modelling - Perpectives from Artificial Intelligence, Databases and Programming Languages", M.L. Brodie, J. Mylopoulos and J.W. Schmidt (eds), Springer-Verlag, Harrisonburg, Virginia, 1986.

[REUB.91] : H.B. REUBENSTEIN, R.C. WATERS, "The requirements Apprentice: Automated Assistance for Requirements Acquisition", IEEE Transactions on Software Engineering, Vol 17, No 3, 1991.

[ROLL.82] : C. ROLLAND, C. RICHARD, "The REMORA Methodology for Information Systems Design and Management", in IFIP WG8.1 Working Conference on Information System

Design Methodologies : a Comparative Review, T.W. Olle, H.G. Sol, A.A. Verryn-Stuart (eds), North-Holland (pub), 1982.

[RUEH.88] : M. RUEHER, D. LADRET, B. LEGEARD, "Capturing Software Process Through the Generated Objects", Proc. of the 4th Software Process Workshop, 1988.

[RUMB.87] : J. RUMBAUGH, "Relations as Semantic Constructs in an Object-Oriented Language", Proc. of OOPSLA'87 Conference, SIGPLAN Notices, Vol 22, No 12, 1987.

[RUMB.91] : J. RUMBAUGH, M. BLAHA, W. PREMERLANI, F. EDDY, W. LORENSEN, "Object-Oriented Modeling and Design", Prentice-Hall, Englewood Cliffs, NJ, 1991.

[SCHI.83] : U. SCHIEL, "An abstract Introduction to the Temporal-Hierarchic Data Model (THM)", in Proc. 9th Int. Conf. on Very Large Data Bases, Florence, 1983.

[SCHM.75] : H.A. SCHMID, J.R. SWENSON, "On the semantics of the relational model", In Proc. of the ACM SIGMOD Conference, pp 211-223, 1975.

[SCHM.89] : J.W. SCHMIDT, C. THANOS, "Foundations of Knowledge Base Management, Contributions from Logic, Databases and Artificial Intelligence", Cambridge University Press, 1989.

[SCHR.84] : M. SCHREFL, A.M. TJOA, R.R. WAGNER, "Comparison-criteria for Semantic Data Models", IEEE Transactions on Software Engineering, 1984.

[SEID.89] : E. SEIDEWITZ, "General Object-Oriented Software Development: Background and Experience", syst. and Softw. 9, 1989.

[SERN.88] : C. SERNADAS, J. FIADEIRO, A. SERNADAS, IFIP Conference on "Object-Oriented Conceptual Modelling from Law", In "Artificial Intelligence in Databases and Information Systems", Ed. R. A. Meersman, Z. Shi and C.H. Kung, North-Holland, Guangzhou, China, 1988.

[SERN.89] : A. SERNADAS, J. FIEDERO, C. SERNADAS, H.D. EHRICH, "The Basic Building Block of Information Systems", In Proc. of the IFIP WG 8.1 Working Conference on Information Systems Concepts: An In-depth Analysis,Namur, Belgium, October 1989.

[SERN.90] : A. SERNADAS, H.D. ERHICH, "What is an Object after all?", In Proc. of Conf. on Object-Oriented Databases, Windermere, U.K., July 1990.

[SERN.91] : C. SERNADAS, P. RESENDE, P. GOUVEIA, A. SERNADAS, "In-the-large Object-Oriented Design of Information Systems", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[SHIP.81] : D.W. SHIPMAN, "The Functional Data Model and Data Language DAPLEX", ACM Trans.on Database Systems, Vol 6, No 1, 1981.

[SHLA.88] : S. SHLAER, S.J. MELLOR, "An Object-Oriented Systems Analysis, Modelling the World in Data", Yourdon Press, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[SHLA.89] : S. SHLAER, S.J. MELLOR, "An Object-Oriented Approach to Domain Analysis", Software Engineering Notes, Vol 14, No 5, 1989.

[SMIT.77a] : J.M. SCHMITH, D.C.P. SCHMITH, "Database Abstractions: Aggregation", Comm. of the ACM, Vol 20, No 6, June 1977.

[SMIT.77b] : J.M. SCHMITH, D.C.P. SCHMITH, "Database Abstractions: Aggregation and Generalization", ACM Trans. Database Systems 2, 2, pp 105-133, 1977.

[STAC.85] : R.A. STACHOWITZ,"A Formal Framework for describing and Classifying Semantic Data Models", Information Systems, Vol 10, No 1, 1985.

[SU.83] : S.Y.W. SU, "SAM*: A Semantic Association Model for Corporate and Scientific-Statistical Databases", Inf. Sci. 29, 1983.

[SUTC.91] : A.G. SUTCLIFFE, "Object-Oriented Analysis : The Abstract Question", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[SYSL.90] : SYSLAB Research Plan 1990-93, Department of Computer and Systems Science, University of Stockholm, Sweden, 1990.

[TALL.90] : B. TALLDAL, B. WANGLER, "Extracting a conceptual model from examples of filled in Forms", Proc. of Int. Conf. COMAD, pp327-350, N. Prrakash (Ed), New Delhi, India, 1990.

[TEMP.91] : TEMPORA Esprit 2 Project, Concepts Manual, 1991.

[TSIC.82] : D. TSICHRITZIS, F. LOCHOVSKY, "Data Models", Prentice-Hall, Englewood Cliffs, N.J., 1982.

[VILA.82] : M.B. VILAIN, "A System for Reasoning about Time", In Proc. AAAI 82, Pittsburgh, USA, 1982.

[WAND.89] : Y. WAND, R. WEBER, "An Ontological Evaluation of Systems Analysis and Design Methods", In Proc. of the IFIP WG 8.1 Working Conference on Information Systems Concepts: An In-depth Analysis, Analysis of some Fundamental Information Systems Concepts, Namur, Belgium, October 1989.

[WARD.89] : P. WARD, "How to Integrate Object Orientation with Structured Analysis and Design", IEEE Computer, March 1989.

[WIER.91] : R.J. WIERINGA, "Object-Oriented Analysis, Structured Analysis, and Jackson System Development", In Proc. of the IFIP TC8/WG8.1 Working Conference on the Object-

Oriented Approach in Information Systems, F. Van Assche, B. Moulin, C. Rolland (Eds), North-Holland (pub), Quebec city, Canada, October 1991.

[WIRF.89] : R. WIRFS-BROCK, B. WILKERSON, "Object-Oriented Design: A Responsability-Driven Approach", In Proc. of OOPSLA'89 Conference, SIGPLAN Not. (ACM) 24, 10, October 1989.

[WIRF.90] : R.J. WIRFS-BROCK, R.E. JOHNSON, "Surveying Current Research in Object-Oriented Design", Comm. of the ACM, Vol 33, No 9, September 1990.

[WOHE.88] : R. WOHED, "Diagnosis of Conceptual Schemas", In Artificial Intelligence in Databases and Information Systems, R.A. Meersman, Z. Shi, C.H. Kung (eds), North-Holland, Canton, China, 1988.