T H E   C O N C E P T S   B E H I N D
A   S Y S T E M A T E Q U E

A Conceptual Model of Business Systems

Paul Lindgreen

Institute of Informatics
Copenhagen Business School
Denmark

Abstract: The paper introduces the word 'Systemateque' as a general term for a database similar to a data dictionary, but with a much more general scope. While a data dictionary is a database for support of the design and maintenance of data processing systems, a systemateque is the database of a more advanced and general CASE tool supporting all kinds of systems analysis and systems design activities.

A traditional way to establish computerized support of such activities is to extend a data dictionary system in an ad-hoc manner, for example by utilizing a common facility to record user-specific matters. Often this can be done in a rather liberal way, but also with limited support from the system, because basically it is constructed for another purpose. As a consequence, the focus will continue to be at the data processing system instead of on the environment - the business system.

The paper, therefore, approaches the problem the other way round: A study on the so-called "conceptual level" is performed of the concepts relevant for a systemeer performing business system analysis and design of changes to possible sub-systems within a business system. The concepts are related to those behind the popular "Structured Analysis" approach and to the traditional data dictionary concepts exposing a data dictionary as only a narrow and special example of a systemateque just as a data processing system is only a part of a business system.

1. Introduction

The notion of a data dictionary emerged when it was realized that computers and database principles could be used to record and keep track of the documentation produced during the design of data processing systems. A data dictionary (DD), simply, is the database of a CASE tool to support the design and maintenance of a data processing system.

Although the function of a data processing system (DPS) is just a small part of what goes on in the surrounding organization, the focus of most such CASE-systems has always been on DPS-implementation matters and on computer oriented aspects. The concepts covered by the majority of DD-based CASE-tools are centered around programs, files, records, fields and similar phenomena.

Most DD-systems has some kind of "escape"-facility that in principle permits the recording of all kinds of user-specific phenomena. Usually, this can be done in a rather liberal manner with practically no restrictions - neither as regard relationships between the phenomena nor attributes characterizing them. But, usually, the DD-systems provide no database schema for these user specific aspects, and there are no build-in functions for consistency check or for nice presentations of the recorded facts in reasonable reports, maps, screen surveys etc. At best these functions can be in-house programmed as "exit-routines", if the system allows for that.

However, in general an approach of this kind is wrong. Trying to incorporate more and more aspects into a too narrow or specialized frame - even if it is technically possible - will lead to bad solutions. It is a much more sound principle first to attack the problem in a general way. In most situations this will provide a wider an more flexible frame. Then, if it is useful for the application, it is possible to optimize one or more sub-areas within the frame with special functions.

The present paper attacks the problem in this general way: The aim is not a traditional DD, but a more comprehensive and general <u>systemateque</u>:

A systemateque is a database for CASE tools supporting the systems analysis and systems design process in general. The scope of a systemateque is not just a DPS, but the much more comprehensive business systems. A systemateque, thereby, automatically will incorporate the scope of the usual DD. It also will enable all the tradidional functions of DD systems. However, it may very well turn out that other sub-areas appear to be more important and that other kinds of functions than those known from DD system will be more useful.

The purpose here is not to design a certain systemateque for a specific CASE tool, but rather to perform a general <u>information analysis</u> of the area covered by systemateques - the domain of systems analysis and systems design. This area typically comprises organizations, factories, companies, institutions (public or private), shops, communities, projects etc. or any organizational or geographical part hereof, like production plants, inventories, sales departements, transport sections etc. Any of these kinds of business organizations can be regarded as a system in interaction with an environment. In this paper such systems will be called business systems.

When a busisness system is analyzed the facts about the system are collected step by step in an incremental way. To keep track of the facts they should be recorded in the systemateque. But in order to obtain proper CASE support it is necessary that these facts are of certain relevant kinds "known" by the CASE tool. By known is meant that the internal structure of the systemateque must reflect the types of phenomena typically observed in business systems. The systemateque must enable the recording of these facts, and the CASE tool must be able to check the consistence of them according to a set of rules valid for the relationships between the different kinds of business system concepts.

This requires a general model covering all variants of business systems from the above mentioned domain. Such a model defines a set of "generic" kinds of entities that appear in specific forms in each business system. The paper describes such an information model of business systems to be applied when a specific systemateque-based CASE tool is designed.

2. General assumptions and fundamental concepts

In general a model is a description expressing certain aspects of a given domain. But a model is not a complete description. Usually it expresses only a part of what can be said about the domain. In order to be understood the model must also apply a well known language. In our case the language could be similar to that applied by a systemeer performing usual systems analysis. That is so, even here where the analysis is on a higher abstraction level: Not a single, but all kinds of business systems are considered. But the problem is that the language used by systemeers is not a unique well-known language. Each systemeer uses his private dialect based on his own set of pet concepts.

It is not the point here to discuss such dialects. (This is relevant when a CASE tool with a systemateque is designed). But in order to understand the model the reader must be aware of a few fundamental concepts and also of the restricted view of the domain that is enforced by the model.

This section, therefore, outlines some model principles and part of a general conceptual foundation, that is needed in order to understand the elements of the model. (A more detailed discussion of these fundamental matters can be found in (Lin 89)). But neither the principles nor the fundamental concepts and the applied terminologi is the issue of this paper. The application of them in the business system model, however, is.

System and system-related concepts

The concept system is not uniquely defined in the litterature, but typically system can be found explained as:

"A collection of interrelated parts characterized by a boundary with respect to its environment" (Iiv 83)

or just as:

"A set of objects with a set of relations" (Lan 71).

Most people intuitively agree on such simple definitions. Apparantly they are broad enough to cover the meaning of usual linguistic constructs where 'system' is used.

But system is a much more difficult concept. If we look at what in practice are considered systems, and if we really think about it, it becomes obvious that some very important aspects of the system concepts are missing in the traditional definitions:

A system is not an absolute or objective phenomenon. There must be a system viewer who can see a purpose in regarding something as a system. This purpose can be expressed as at least one meaningful relationship between the collection of parts considered as a whole and the environment. I general system theory such a relationship is called a systemic property. A very important point is that a systemic property is not possessed by any of the individual parts. It is a property the system viewer associates with the parts conceived as a whole. One system viewer sees the system as having one set of systemic properties, while another viewer will see other properties with the same collection of parts.

Accordingly it is necessary to consider the following concepts:

A system is someones conception of a set of parts that are related such that they can be regarded as a whole conceived to have at least one particular systemic property in relation to the environment that is not possessed by any of the individual parts.

A system viewer is a person who conceives a set of related parts as a system.

A system domain is the set of related parts in some area that is conceived as a system by the system viewer.

The environment of a system are the conceived parts of the world outside the system domain that are necessary in order to describe the systemic properties of the system.

A business system, then, is a system conceived from domains of the kinds of enterprises listed in the introduction (organizations, companies, departements etc.). This still leaves business system a rather vague concept, but referring to some well-known concepts from general systems theory (see (Ack 71) and (Chl 81)) a business system is an open, active and purposive system. This means that activities are carried out in the system as a means for a purposeful interaction with the environment.

The systemic property of a business system is its function in relationship with the environment. This function is that it - either as respond to impressions from the environment or by its own initiative - carries out activities that are intended to satisfy or change the behaviour of the environment.

If on the other hand we look at what goes on inside a business system, the most prominent characteristic is that the activities are carried out by actors and that the actors must communicate in order for the system to behave in the purposeful manner expressed by its systemic function

Simplifications and restrictions

The model does only express certain aspects of business systems. There are some assumed restrictions or simplifications as regards what is considered relevant (possible, useful, reasonable?) to state something about. These restrictions are listed below:

First, the model leaves out all aspects of energy transformations, such as, for instance, the production or consumption of heat, the radiation of light, metabolism in living organisms and so on. More generally the model exclude all molecular aspects of physical, chemical and biological processes, for instance the hardening of concrete, the growth of a tomato plant or similar. This means that phenomena of these kinds are ignored in the expressions of the model. However, when appropriate models of such phenomena exist (from biology, chemistry etc.), the expressions of these models could be included in the model domain. An example of that could be the activity of removing the shutter boards from a concrete wall at a certain time, because a model of concrete hardening says it is safe enough to do so.

Secondly - because people, so far, know too litle about human behaviour in general - the model restrict itself to consider reproducible activities only. This means that the model ignores all kinds or aspects of human behaviour that cannot be described precisely enough to enable other persons to perform in the same way. Considering social interaction in general, this may look as an intolerable restriction, but it may not be so: In fact, most activities in business and enterprises are intended to be reproducible - simply in order to ensure their purposefulness. In public administration or civil service it may even be required by law, and the very basis for sensible accounting, inventory control, invoicing, use of computers, communication in traffic control etc. are well defined rules, which must be - and usually are - followed exactly.

Third, only activities with an explicit beginning and end are considered. Whenever an activity is carried out it must be possible to distinguish between the situation before the activity is started and the situation left after the activity is finished. This means, for example, that phenomena like "there is a growing concern about ..." or "as time went by, it became more and more difficult to ..." must be treated as a succession of discrete changes, each one caused by the execution of an explicit activity.

Fourth, only activities carried out by an explicite actor are considered. Possible activities that goes on by themselves are ignored. Actors may act as individuals or as members of working teams or together with technical devices (e.g. computers), but then it is the group or the symbiotic functioning parts that serve as the actor.

3. Concepts for the description of business systems - an informal introduction

In the following the domain of a business system is called the business domain, while 'system' is used short for 'business system'.

Actors

In a business domain changes occur as a consequence of what in the system is called activities. In the conception of the business domain as a system an activity is always carried out by an actor. In the language of the model the actor is said to be playing a role as agent for the type of activity in question.

According to the model a certain actor can be agent for many different types of activities, and a certain type of activity may have several actors, each one serving as a potential agent for the activity. At the same time many different actors may be active carrying out activities. One actor may also be active with more than one activity at the same time - even with several activities of the same type.

The most obvious kind of an actor is a single person, who is skilled to perform the types of activities the actor is agent for. But an actor may also be a device, a machine, a robot or a computer that somehow is programmed or set up to perform the duties of being agent. The two kinds of actors differ in that human actors may be and usually are conscious about the purpose of their activities and usually have an interest in fulfilling the purpose. Neither is the case for any kind of artefact actors. A human actor can be "a responsible agent". An artefact actor can not!

However, an actor may also be any grouping of these kinds of actor candidates, for example, a team of workers, a secretary working with a typing machine, or an interconnected set of computers serving as a packet switching network for data transmission. In general any sub-system of a business system which itself is a business system can be regarded as an actor performing certain types of activities. These activities, in fact, constitute the systemic function of the sub-system.

Note that the same person in a domain may be conceived as more than one actor. A person is a phenomenon in the business domain. An actor is a conception - a part of the system view!

Activities

In the model activities are always temporally discrete, which means that every activity has an explicit start and an explicit termination both controlled by the agent as a part of the execution. During the period between start and termination the state of the system is changed.

Since many activities in a system potentially can be active at any time, the (total) state of the system (or of the business domain) is very difficult - if not impossible - to consider. But each instance of a certain type of activity has a certain delimited (partial) part of the phenomena of the business domain as its set-off or pre-state and a corresponding part as its resulting state. Therefore, in the model the termination of each activity results in a new partial state in the business domain.

For a business system a characteristic property is that the creation of each such resulting partial state can be regarded as the purpose of the execution of the corresponding activity. The partial states resulting from the execution of activities may be constituted by the mere existence of so-called reagents - concrete physical objects (or delimited amounts of matter) - or by the current location of such reagents, or they may be abstract phenomena (for instance the result of a decision). Finally, they may be a combination of both abstract and concrete phenomena.

In general an activity is a structure of sub-activities which on a basic level are elementary acts. An elementary act is a sub-activity that not in a reasonable way can be described by further sub-activities.

The model consider the following kinds of elementary acts:

- a measurement of the value of a physical property. This includes the counting of all kinds of reagents in sets or of occurrences of events inside a given period

- a physical transformation of a single reagent or a set of reagents into another set of reagents

- a movement of an reagent from one location to another

- a decision

- representation of an instance of a certain type of information in a data set

- realization of information from a given data set.

An activity can be interpreted (and described) i two different ways - as a process or as an act:

Regarded as a **process** one adopt a view relatively close to the reality. In this way the activity is seen as one in principle arbitrarily fine temporal structure of sub-activities which in corresponding small steps change the business domain from the partial pre-state to the resulting post-state. The condition for a process-view according to the model is that the activity is confined to what a single agent performs in the period of activity execution.

Therefore, the intermediate sub-states appearing during the execution will be relevant for the agent only and are of no importance for any other actor in the system.

Alternatively it is possible to abstract from the temporal progress of sub-activities and regard the activity as an act. As such only the resulting partial state is of importance and in some cases the total duration of the activity as well. In this way it is also possible to consider sub-activities, but these sub-acts are not mutually time dependant. They just reflect a corresponding sub-division of the resulting state in sub-states, such that each sub-act contributes solely and entirely to the associated sub-state. The difference between a sub-state created by a sub-act and an internal sub-state created during a process is, that the first may be relevant for the execution of other activities in the system - the latter is not.

In general a business system can be regarded as having an <u>internal function</u> and an <u>external function</u>. The internal function is mainly concerned with the actors and their activities and their mutual communication. The external function concerns the different kinds of <u>impressions</u> from the environment, i.e. the changes of state in the domain caused by activities outside the system, and the <u>expressions</u> from the system, i.e. the changes in the domain of the environment caused by activities in the system. The model recommends the following language: The internal function of a business system is referred to as "the function <u>in</u> the system" while the external function is "the function <u>of</u> the system". The latter is equivalent with the systemic properties of the business system.

Events

Usually, the activities in a system are executed in an asyncronous manner. The actors possess the necessary ressources to carry out the execution of the activities in an autonomous way. However, the activities depend on each other in a complex dynamic way necessary in order for the system to behave as a whole in the specific systemic way. Typically, activities are dependant in a way where it has no sense executing a certain activity, before one or more other activities are terminated, because they together provide the necessary set-off state for the activity.

As a consequence the individual actors must signal to each other about the termination of their own activities, or actors must observe the performance of other actors, such that each activity execution temporally fits into the collective, purposeful pattern in a proper way.

In the model the dynamic dependencies between the activities are expressed by means of the concept <u>event</u>. An event is an abstraction of certain phenomena in the system or in the business domain or from the system that may be conceived as a potential <u>cause</u> for an actor to start the execution of an activity, such that the execution is purposeful under the given circumstances. Events connect the activities in a system in a dynamic network. In the language of the model an event may <u>trigger</u> an agent to start the execution of an instance of the corresponding activity type. Note, this is a part of the system view. The cause/effect abstraction (event -> trigger -> agent -> start activity) is relevant independant of how the signalling/observation is accomplished in practice in the business domain.

Basically there are the following kinds of events:

- Externally initiated events that occur when a physical change of state in the business domain is caused by an impression from the environment.
- Activity initiated events that occur whenever an activity is terminated
- Receipt initiated events that occur when an actor receives a certain type of reagent or as a part of a communication a certain type of data set (see below). (This kind of event is, in fact, a special case of an activity initiated event).
- Time initiated events that occur at a certain pre-defined point in time or repeatedly with a given frequency or after the elapse of a certain period of time after another event.
- Agent initiated events that occur when an agent decides so. Through mental awareness of the situation, the agent, who must be a human actor, invokes himself to act in a certain way. The invocation happens during some kind of "tolerance period" inside which the purposefulness of the act is unchanged, but otherwise it is beyond the model to express anything about the conditions for the event to occur.

The same event may trigger the execution of several different activities.

Communication

The actors not only signal about the proper times for the activations of activities. They also communicate with each other exchanging the information that is necessary for the execution of the activities in order that they can contribute in a purposeful way to the systemic function of the system.

In the model **information** is explained as formalized knowledge about partial states in the business domain. When the execution of an activity is terminated the executing agent is the only one in the system knowing about the resulting partial state. Of course, after that any other actor may perform observing activities (measurements) that provides them with information about the result. If that is done so in order for the system to function properly, these measuring activities belong to the system. But, usually, the information is provided by a formal communication between the actors in question.

A communication between two actors is a structure of separate activities, (which partially are of the kinds of basic acts: representation, movement and realization). The activities involve the two communicating actors as agents, but, usually, also one or more other actors. In a communication situation there is a sender who possesses the information and a receiver who shall have the information - either because the receiver needs the information himself, or because other actors in the system intend that the receiver should have it. Accordingly a communication can be triggered by the sender (the communication is an announcement), by the receiver (the communication is a question/answering sequence) or by a third part (the communication serves management or control).

During a communication the sender produces a data set representing information of a certain type according to conventions that are agreed upon between the sender and the receiver - a so-called communication protocol. A data set is a structure of physical phenomena that can be sensed and interpreted by the receiver according to the communication protocol, such that the represented information is realized. During the communication the involved data are transported and/or handled in various ways in the business domain until eventually they reach the receiver. The sender and the receiver may be the same actor, who communicates over time. This is possible if the data are stored properly and made available at a later time.

The transport/storage of a data set involves a separate structure of often rather complicated activities (movement, transmission, separation, transformation, conversion to other kinds of physical phenomena, reforming etc.). Each of these data handling activities can be carried out by the sender or by the receiver, but usually, one or more other actors are involved with the job.

According to the model the data handling activities can be considered activities on the same level like all other activities in the systen. But it is also possible - and often an advantage - to conceive them as a separate communication system.

Note, that according to the model (even with its limitations/simplifications) communication correspond to what really goes on in the business domain: It is actors who communicate! Not as by the so-called "structured analysis" methods where data is claimed to be flowing between processes. Apart from the fact that the data involved in a communication will not flow continously, but rather separated into data sets and handled in a discrete manner, there are always actors involved! The information is exchanged between the sender and the receiver, who both are actors. The involved data are also handled by actors.

Information and entities

The **information** necessary to carry out the activities in a purposeful manner is always about partial states in the business domain. Usually, the partial states are the result of activities carried out in the system, but certain states may have existed from a time before the "temporal scope" of the system. Some changes of state may also be caused by the environment, but in order for these to be of importance for the system, there has to be at least one activity carried out by an actor in the system for each such impression. During an ("accepting") activity the impressed state (or part of it) is realized by an actor in the system. (F.ex. goods are not just delivered to a company, they are also, somehow, received by a responsible person representing the company).

In practice, however, only a limited set of the properties of the partial states as they exist in the business domain is of interest for the system. (This, in particular, is where a distinction between system and system domain becomes important). A lot of properties are of absolutely no relevance - neither for the function **in** the system nor for the function **of** it. It may very well be that some of these irrelevant properties are of importance in other systems conceived from the same domain, but that will not make them relevant in the current one.

Therefore, it is necessary to consider a special system concept that precisely expresses those properties among the (infinitely?) many of a partial state that are relevant for the system. In the model these state interpretations (or conceptions) are called <u>entities</u>. Thus, in the model there is a clear distinction between on the one side the partial state, which is a part of the domain, and on the other side the corresponding entity. **An entity is a conception** and is a part of the system. An entity is an abstracted partial state characterized by a certain subset of the properties of the state. Note, that this is valid regardless of whether the state is physically concrete like a car or abstract like an order or an account.

It is an important aspect of the model that the execution of an activity of a certain type causes the creation of an instance of at least one entity type. Thereby a considerable part of the semantics of an entity type is in the type of creating activity.

This relationship between activity type and entity type and the fact that (usually) entities as seen by the model are created through the execution of activities, is a very important point, and it gives the model a expressional power compared to other (explicit or implicit) models known from the area of systemeering and "conceptual modelling". For a long time entities has been used as a relevant concept, but so far it has remained a rather vague concept. It has not been explained in any reasonable depth of understanding. Usually 'entity' has been treated just as a generic term for all kinds of phenomena in the real or imagined world (cf. the definition in the ISO report (ISO 82)). What it is that creates entities and contributes to the semantic of entities - in reality or in some model view - has been ignored or at least not given proper attention.

Apart from the creating activity an entity type is characterized by the specific set of system relevant properties of the corresponding partial state. Each property expresses a certain fact about the partial state that is of importance for the function in the system.

Properties of entities can be classified as either <u>role properties</u> or <u>attribute properties</u>:

A role property is a property of an entity that expresses a relationship with a number of entities in the system - all of a certain type called the <u>object entity type</u>. A role property is characterized by:

- a reference to the related object entity type. The role may be reflexive such that the object entity type is identical with the current one.
- a so-called <u>span</u> (or cardinality) that expresses the possible minimum and maximum number of entities of the related object entity type
- a reference to an <u>opposite directed role</u> that expresses the inverse relationship. This opposite role is a property of the object entity type.

An attribute property is a property that associates a discrete value with each entity posessing the property. The value further specifies the fact expressed by the the property. An attribute property is characterized by:

- a <u>value type</u> that defines the domain of potential values
- a possible <u>measuring unit</u> for the values (this is relevant for certain types of attributes with numeric values)
- a possible <u>accuracy</u> of the values. Where it is relevant it expresses the uncertainty of values associated with a given type of entity, for example, values determined by a measurement or estimated by an agent.

It is important to stress the distinction between a value and a possible piece of data representing the value. A value is abstract while data is concrete. For example, the numeric value expressing the number of submitted papers to CASE 89 is an abstract phenomenon and different from the possible set of physical symbols representing it on a piece of paper to the program commitee. However, a value may very well itself be a data set representing information from another system. Consider, for example, the value of the property 'address' of the entity 'Customer' in some sales system. This value is data representing information from another system, f.ex. a post distributing system.

4. A traditional approach or systemateque-based CASE systems?

None of the concepts from the described business system model should be mysterious or cause any surprise for a professional systemeer. All the concepts of the model reflect relevant phenomena, which everybody - familiar or not with the task of analyzing and describing a business system - can observe in every organization.

Still many people will regard the model as too complex - covering too many concepts. They prefer simpler or more naive models. Very often - in particular among practitioneers - the opinion is expressed that in general a simple model is much better than a more complex one. These people seem to put more emphasis on how easy the model is to understand and to apply than on the quality of the obtained results. Doing so they forget, that applying a simple model will often give a wrong picture, because too many important aspects are ignored.

A model will always give a restricted picture of its domain, but that doesn't mean that it necessarily must distort the domain or give a unrealistic limited picture of it. If the domains of the systems are complex, as most organizations are in practice, it is naive to believe, that one can understand and describe them by means of a simple model with only a few, uncomplicated concepts covering only a very restricted part of what really goes on.

The two most serious causes of distortion in systemeering work today are:

- the strong focus on data processing at the expence of other relevant functions in the organization. This is best exemplified by the limited scope of traditional data dictionaries.

- the worldwide, rather uncritical commitment to the strange model of reality enforced by the so-called "structured analysis" methods (Mar 78).

It is beyond the scope of this paper to discuss in detail the many fallacies of the structured analysis approaches, but on the background of the presented business system model a few points should be mentioned (see (Bub 87) and (Flo 86) for a more extensive discussion):

The scope of structured analysis (SA) is stated to be business systems, but there is no explicitly expressed model behind SA and some of the concepts are rather vague (see above mentioned references). Worse however, is that a number of important aspects of business systems are completely ignored:

The most obvious flaws - probably also the most serious ones - are that neither actors nor the dynamics of activities are considered. Activities seem to be regarded as "running" forever. There is no event concept, and there is no other way to describe the activation and termination of activities.

Communication, as well, cannot be described - neither as something discrete taking place only at certain times, nor as regard what is communicated. In SA there is only a rudimentary information concept. It is not possible to see, what the data "flowing" between processes really represent. It can only be assumed from the possible intuitive associations one may have with the names used for the data components. As another consequence it is not possible to specify precisely what the necessary information is for an activity in order to carry it out in a proper way.

There are other strange omissions and inconveniences with SA and its implicit model, but let us turn to the other common approach, the use of traditional data dictionaries:

Also here there is no explicitly formulated model behind, but the types of phenomena usually supported by DD-based systems can be broadly categorized as:

- Data structure oriented comprising phenomena like, for example, field, record, file, database, report, and screen image
- Data process oriented comprising, for example, system, program, module, and sub-routine
- Hardware/environment oriented like computer, terminal, network, and user.

If we compare these DD-concepts with the business system model, it is obvious that the view expressed by the DD-approach is equally restrictive as that enforced by SA:

Again information cannot be described properly. Only through the possible intuitive associations with names for records and fields. It is possible to describe data sets of various kinds, but not how they are used in communication. Computers, terminals and users could be regarded as actors, but not as communicating actors and not in any general sense. Which kinds of activities, for example, are users carrying out according to the DD concepts? Furthermore, like it is with SA, the kinds of activities considered are restricted to data processing. Also, in most cases the activities can only be described as processes, not on the higher abstraction level as acts. The static structure of activities can be described as reflected in the rather specific types of relationships: programs as parts of systems and subroutines or modules as parts of programs. But description of the dynamic aspects of the activities is not supported by data dictionaries.

There seems to be all reason to abandon the traditional approaches with their rather simplistic view of reality and turn to one with a better conceptual foundation. The systems analysis and systems description process is so demanding, that it must be supported by a powerful CASE tool. But a CASE tool alone will not help if it doesn't reflect the complexity of the domain. The domain of a systems analysis and design process, the organization, is so complex - even if we restrict us as described in section 2 - that the CASE tool should not be based on naive and unrealistic models.

The database of the CASE tool - the systemateque - must be designed to reflect all relevant aspects of the domain within the accepted limitations. If the realized facts about the organization cannot be accomodated in the database, then, first, it cannot constitute a proper description of the system (or provide the information for one), and, secondly, it cannot form the basis for proper CASE functions to support the analysis and design activities.

From the business system model described in section 3 an information model can be derived that specifies the various types of generic business system entities with specifications of their mutual relationships and attributes. This information model should form the basis for design of the systemateque when the available database implementation tools are known.

But inherent in the business system model are also criteria for the design of one or more <u>formal languages</u> enabling the specifications of conditional, computational, temporal and other "quantitative" structural facts about the business domain. Expressions of these kinds will refer to operands of the types reflected in the information model, but the consistency rules to apply on such expression require other formal means than the information model. This, however, is not considered in this paper.

5. Conclusion

What is stressed here, primarily is to base the development of a CASE tool for systems analysis and systems design on a model that really reflect the many aspects of the business domain. Such a model has been outlined in the paper. It expresses that the business domain may be conceived as a business system, shortly characterized in the following way:

People and/or artefacts conceived in the business system as **actors** carry out **activities** caused by the occurrence of various kinds of **events**. The execution of an activity changes the state in the domain to something, which in the system is conceived as an **entity**. Entities expresses the information that is relevant for a purposeful execution of the activities. In order to provide the information for the proper actors, they must **communicate**. This is done by means of **data** that is exchanged among the actors.

References:

(Ack 71): Russel L. Ackoff: "Towards a System of System Concepts". Management Science Vol.17. July 1971.

(Bub 87): Janis Bubenko jr.: "Problems and Unclear Issues with Business Activty and Data Flow Modelling". SYSLAB working paper no.21, University of Stockholm 1987.

(Chl 81) Peter Checkland: "Systems Thinking, Systems Practice". J.Wiley & Sons 1981.

(Flo 86): Christiane Floyd: "A Comparative Evaluation of Systems Design Methods". In: Olle, Sol, Verrijn Stuart (eds.): "Information Systems Design Methodologies - Improving the Practice" (CRIS III). North Holland Publ. Co. 1986.

(Iiv 83) J.Iivari: "Contributions to the Theoretical Foundations of Systemeering Research and the PIOCO Model"; Acta Universitatis Ouluensis Ser.A no.150. Univ. of Oulu 1983.

(ISO 82) J.J.van Griethuysen (ed.): "Concepts and Terminology for the Conceptual Schema and the Information Base". Publ. ISO/TC97/SV5 - N695, March 1982.

(Lan 71) Börje Langefors: Editorial notes to: Bubenko/Langefors/Sølvberg (eds.): "Computer Aided Information Systems Analysis and Design". Studentlitteratur, Lund 1971.

(Lin 87) Paul Lindgreen: "Entities from a Systems Point of View". In: S.Spaccapietra (ed.): "Proceedings from the 5th International Conference on Entity - Relationship Approach" North Holland Publ. Co. 1987.

(Lin 89) Paul Lindgreen: "A Foundation of General Concepts for the Area of Information and Business Systems". Paper submitted to the IFIP WG 8.1 working conference: "Information System Concepts - An In-depth Analysis", Namur Oct. 1989.

(Mar 78) Tom De Marco: "Structured Analysis and System Specification". Yourdon Press, New York 1978.