A Communication Oriented Approach to Conceptual Modelling of Information Systems

Jan L.G. Dietz

University of Limburg
Faculty Economics and Business Administration
P.O.Box 616, 6200 MD Maastricht, The Netherlands

Abstract

Informata are introduced as a subclass of information systems. The SMARTIE framework is presented which supports the conceptual modelling of informata. The core of the framework is an automaton, which can be viewed as a generalization and an extension of the finite state machine, but which also draws on results of language philosophical research. A model of a system in this framework is called a communication model, as opposed to the conventional process model. Several techniques supporting the modelling and the specification of systems in the SMARTIE framework are presented. To demonstrate the practical applicability of the modelling principles, a traffic control system is taken as an example.

1. Introduction

The term "information system" is widely used nowadays to denote quite diverse things. E.g. an inventory control system is called an information system, but the particular way in which a marketing department of a company organizes the fulfillment of its information needs is also called an information system. Because not everything presented in this paper applies to all information systems, a subclass is delimited, which we prefer to call informata (singular: informaton).
An *informaton* (from *infom*ation and auto*maton*) is an information generating system, the operation of which is discrete and (potentially) completely specified. Non-determinism is allowed. Discrete means that only discrete information items are taken as input, and that the number of items generated in any finite time interval

is finite. Although mostly informata are implemented using artefacts (computers), the incorporation of human beings is not excluded, provided their behaviour is prescribed and bounded to this prescription. For the remainder of the paper, the words "system" and "informaton" are considered to be synonym.

An informaton may be conceived to consist of a number of information producing units, which communicate by sending messages to each other. A model of a system in this respect is called a *process model*. A well-known process modelling technique is the DFD technique [5,12]. While this technique effectively supports the development of process models, it is not very well suited (and indeed is never meant to be) for the development of models at a higher conceptual level, at which the essential aspects of the communication between the units is abstracted from the particular way in which it is implemented. We will call this type of model *communication model* . The difference in abstraction level between a commmunication model and a process model of a system is comparable to the difference between a conceptual model and an internal model of a data base. The practical significance of a communication model is twofold.
Firstly, it describes the essential features of the communication relationships between units, such that there is precisely one communication model of every system, whereas there may be several process models corresponding to one communication model [3].
Secondly, it differentiates between pragmatic meanings of messages, as will be explained hereafter.

The communication between human beings by means of natural language is extensively studied in language philosophy, notably in [1] and [10]. One of the challenging outcomes of these studies for the field of informatics is that messages convey inseparably two things at the same time: the propositional *content* and the pragmatic *function* (effect, purpose). As an illustration, consider the next example messages:

1: "Mr. Smith wants to book a flight to Toronto"
2: "I will book a flight to Toronto for Mr. Smith"
3: "A flight to Toronto has been booked for Mr. Smith"
4: "Is there a booking of a flight to Toronto for Mr. Smith?"

The propositional content of all four messages is the same, viz. the booking of a flight to Toronto for Mr. Smith. However, their pragmatic functions are different. Message 1 conveys a *request*. The essence of a request is that the sender attempts to get the receiver to act in such a way that the proposition becomes true. Message 2 conveys a *promise*, the essence of which is that the sender commits himself to act in such a way as to make the proposition true. Message 3 conveys an *assertion*. The essence of an assertion is that the sender commits himself to the truth of the proposition. Message 4 conveys a *question*, the essence of which is that the sender attempts to elicit from the receiver as many information as is needed to conclude the truth or falseness of the proposition.
Several more pragmatic categories are distinguished in language philosophy.

However they are not relevant for our purposes, with the exception of one, viz. the *declaration* . Examples of declarations are "I herewith baptize you John" and the whistle-signal of the referee indicating the end of a soccer game. The difference between an assertion and a declaration is that an assertion is based on observing a situation while a declaration creates a situation itself. Messages of the declaration type illustrate pre-eminently that to a large extent the world is being made by language [1]. Searle distinguishes in this respect between 'brute facts' and 'institutional facts' [10].

Building on these language-philosophical basic outcomes we make two simplifications, which seem to be legitimated by our modelling purposes.
The first simplification concerns requests and promises. We assume the promise to grant a request implicit. In other words, the receiver of a request has not the option to refuse. The combination of a request and the corresponding promise is called an *order* . We envisage a collection of orders between two communicating actors, to which the sender can make changes. The receiver is compelled to execute these orders in due time.
The second simplification is that we assume the question related to an assertion implicit. The combination of a question and the corresponding assertion is called a *statement* . Furthermore, we envisage a collection of statements between two communicating actors, to which the sender can make changes. The receiver is able to inspect the contents of the collection at any time.
Although informata do not communicate in the way human beings do, one can take advantage of the language-philosophical analysis of communication acts in describing the communication between informata by distinguishing also between statements and orders.

In this paper a framework, called the SMARTIE framework, and some suitable techniques are presented for the development of communication models. In section 2 the basic characteristics of the framework are presented. Section 3 deals with the specification of the behaviour of a system, whereas the modelling of system structure is discussed in section 4. Section 5 contains a short evaluation of the presented material and some conclusions. As an illustrating example informaton throughout sections 3 and 4, a simple traffic control system is used. The next narrative description applies to it.
There is a simple road crossing having in each of the four directions traffic lights, as depicted in figure 1. The light signals in two opposite directions are identical.
In each direction a recurrent pattern of light signal changes can be observed : ...green - yellow - red - green - yellow .... These patterns are called cycles. Thus, there are only two different cycles in our example, called C1 and C2 (cf. figure 1). At any moment a cycle is in a particular phase (green, yellow or red). A more detailed description of a cycle as well as of the interdependencies between C1 and C2 is depicted in figure 2.
At some distance in front of a traffic light there is a traffic sensor. Every time a car passes such a sensor a signal is produced. If a car stops (for instance because the light is yellow or red) it covers the sensor, thus making it impossible for other cars to pass it.

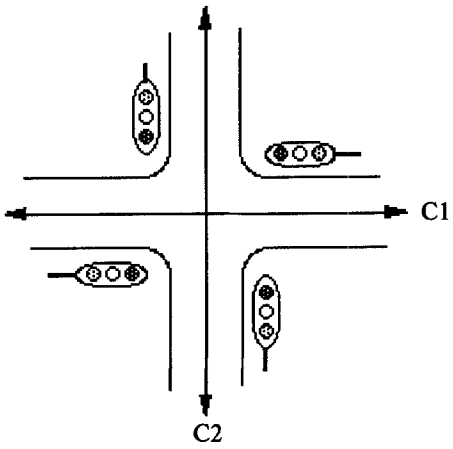


Figure 1. Picture of a simple road crossing with traffic lights

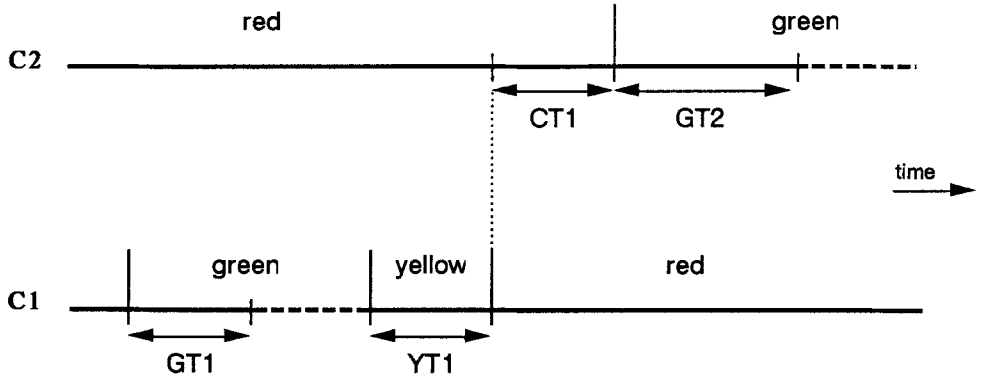


Figure 2. Cycles and their interdependecies

It takes a particular amount of time for a cycle to clear the crossing. From figure 2 it can be read that after cycle C1 goes into its red phase it takes CT1 (Clear Time) time units before cycle C2 goes into its green phase.
A green phase has a minimal duration of GT (GreenTime) time units. It is however prolonged always if there is no traffic asking to pass in the other cycle. The prolonged green phase is indicated in figure 2 by a dashed line.
Contrary to a red phase and a green phase, a yellow phase has a fixed duration. In figure 2 it is denoted by YT (YellowTime).

2. The SMARTIE framework

In this section a particular kind of automaton is described, called SMART automaton or smartie. A smartie can be viewed as a generalization and an extension of the finite state machine [7]. Smarties operate in a linear continuous time dimension, for which the set of real numbers is taken. The presentation in this paper is rather informal. For a formal and extended discussion the reader is referred to [3]. From now on, wherever the word "system" is used, a system modelled as a smartie is meant.
The operation of a system is easiest explained by considering the universe of all systems, called SU (System Universe). SU is finite and closed, so there are no relationships between the systems in SU and the environment.

The communication of statements is modelled using the concept of state. At every moment a system is in a particular state. A *state* is a set of propositions. The propositions p contained in a state are elements of the *state base* of the system, being the set of all propositions which may belong to a state of the system. The communication of a statement by a system to some system(s) is modelled as the generating of the statement $S(p)$ by the sending system and the subsequent change of the state of any system for which p belongs to the state base. A statement $S(p)$ is actual at time t if p is contained in the state at time t.
The communication of orders is modelled using the concept of agenda. At every moment a system has a particular agenda. An *agenda* is a set of pairs <p,t>. The propositions p contained in an agenda are elements of the *action base* of the system, being the set of all propositions which may belong to an agenda of the system. The communication of an order by a system to some system(s) is modelled as the generating of the order $O(p,t)$ by the sending system and the subsequent change of the agenda of any system for which p belongs to the action base. An order $O(p,t)$ is actual at the point in time t if the pair <p,t> is contained in the agenda at time t.

The set of propositions constituting the contents of the orders on the agenda of a system, which are actual simultaneously at some time t, is called the *action* for the system at time t. If there is a non-empty action, the system performs a *transition*, resulting in the production of a, possibly empty, finite set of statements, and a, possibly empty, finite set of orders.
The produced set of statements is called the *mutation* of the system. The contents of these statements are elements of the *mutation base*, which is the set of all propositions the system is able to generate as content of a statement. The generating of a statement instantly changes the state of any system for which the contained proposition belongs to the state base.
The produced set of orders is called the *reaction* of the system. The contents of these orders are elements of the *reaction base*, which is the set of all propositions the system is able to generate as content of an order. The generating of an order instantly changes the agenda of any system for which the contained proposition belongs to the action base.

The performance of a transition consists of the evaluation of a partial function, called the *transition base* of the system. A transition base can be denoted as a set of transitions <a,s,m,r>, where a is an action, s is a state, m is a mutation, and r is a reaction. For every pair <a,s> there is precisely one transition in the transition base. A transition <a,s,m,r> is said to be effectuated if the action of the system equals a and the state equals s. The effectuation yields a mutation m and a reaction r.

Below, a formal definition of the behaviour of a system is presented. In this definition, the powerset of a set X is denoted as $\wp X$, and the set of positive real numbers is denoted as $\Re^+$.

definition 1

The *behaviour* of a system is defined by a tuple < S, M, A, R, T >, where:

S : a set of propositions, called the *state base* ;
M : a set of propositions, called the *mutation base* ;
A : a set of propositions, called the *action base* ;
R : a set of propositions, called the *reaction base* ;
T : a partial function, called the *transition base* :
$T \in \wp A * \wp S \rightarrow \wp(R * \Re^+) * \wp M$;

Sometimes it appears more convenient to define T as < TR, TM >, where:

$TR \in \wp A * \wp S \rightarrow \wp(R * \Re^+)$, called the *reaction function*;
$TM \in \wp A * \wp S \rightarrow \wp M$, called the *mutation function*;

(end definition 1)

Note that S,M,A and R may intersect in any way, as is illustrated in figure 3.

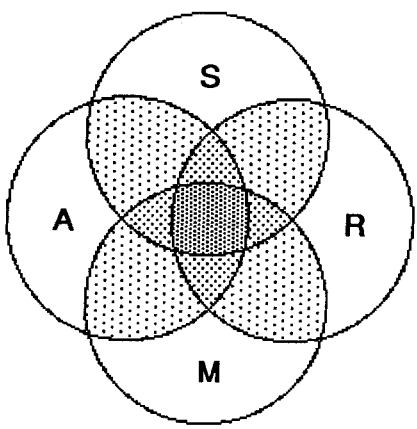


Figure 3. Illustration of the possible intersections of S,M,A and R

The performance of a transition is called an *event*. A sequence in time of events is called a *process* [3]. A process can be described fully by only two process variables. At a point in time t these variables are defined as the agenda and the state at time t. All other variables, which may be of interest, can be defined in terms of these two.

Below, a formal definition of the process of a smartie is provided. In this definition, the set of real numbers is denoted as $\Re$, and the set of natural numbers as $\aleph$.

definition 2

A *process* of a smartie is defined by a tuple < S, M, A, R, T, I, E > , where S, M, A, R and T are defined in conformity with definition 1, and where

I = < IT, IS, IA >, called the *initial conditions*, with

IT : the *initiation time* , i.e. the start time of observing the process;
$IT \in \Re$;

IS : the *initial state*, i.e. the state at time IT;
$IS \in \wp S$;

IA : the *initial agenda* , i.e. the agenda produced internally before IT;
$IA \in \wp(A * \Re)$;

E = < ES, EO > , called the *external influences*, with

ES : the *external statements*;
$ES \in \wp(S * \Re)$;

EO: the *external orders*;
$EO \in \wp(A * \Re)$;

A *process* is described by a pair of process variables $< \phi, \sigma >$, where:

$\phi \in \Re \rightarrow \wp(A * \Re)$; $\phi(t)$ is the *agenda* at time t ;

$\sigma \in \Re \rightarrow \wp S$; $\sigma(t)$ is the *state* at time t;

In order to define these variables, the next additional variables are introduced :

$\tau \in \aleph \rightarrow \Re$; the points in time τ_n are the only moments at which an event occurs;
τ_n is a shorthand notation for $\tau(n)$;

$\mu \in \mathfrak{R} \rightarrow \wp M$; $\mu(t)$ is the *mutation* at time t;

$\alpha \in \mathfrak{R} \rightarrow \wp A$; $\alpha(t)$ is the *action* at time t;

$\rho \in \mathfrak{R} \rightarrow \wp(R * \mathfrak{R})$; $\rho(t)$ is the *reaction* at time t;

The process variables can now be defined as follows:

$\tau_0 = IT$;

$\tau_{n+1} = \min \{ t \mid t > \tau_n \wedge \exists x : <x,t> \in ES \cup \phi(\tau_n) \}$;

$\phi(\tau_0) = IA \cup EO$;

$\phi(\tau_{n+1}) = \phi(\tau_n) \Delta \{ <x,t> \mid <x,t> \in \rho(\tau_{n+1}) \wedge x \in A \}$;

$\sigma(\tau_0) = IS$;

$\sigma(\tau_{n+1}) = \sigma(\tau_n) \Delta (\{ x \mid <x,\tau_{n+1}> \in ES \} \cup (\mu(\tau_{n+1}) \cap S))$;

$\mu(\tau_{n+1}) = TM(\alpha(\tau_{n+1}), \sigma(\tau_n))$;

$\alpha(\tau_{n+1}) = \{ x \mid <x,\tau_{n+1}> \in \phi(\tau_{n+1}) \}$;

$\rho(\tau_{n+1}) = \{ <x,t> \mid <x,t-\tau_{n+1}> \in TR(\alpha(\tau_{n+1}), \sigma(\tau_n)) \}$;

For $\tau_n < t < \tau_{n+1}$ holds: $\sigma(t) = \sigma(\tau_n)$; $\phi(t) = \phi(\tau_n)$; $\alpha(t) = \mu(t) = \rho(t) = \emptyset$;

(end definition 2)

The next event time τ is the next point in time at which an external statement or an external order or an internal order becomes actual.
The agenda ϕ is changed at a time τ by taking the symmetric set difference of the current agenda and the reaction ρ at time τ , as far as the elements of ρ belong to the action base. Initially the agenda contains the internally produced initial agenda and the external orders.
(Note. Actually, a change of the agenda is either the addition or the removal of an order. Consequently it would be necessary to distinguish between produced additions and produced removals, and to choose appropriate operations to deal with them. However, the symmetric set difference operation (denoted by Δ) appears to be a more convenient and elegant operation. The advantage of it is that adding and removing orders in order to arrive at the new agenda are performed in one go. The symmetric set difference of sets A and B is defined as: $A \Delta B = (A \setminus B) \cup (B \setminus A)$. End note.)
The state σ is changed at a time τ by taking the symmetric set difference of the current state and the union of the external mutation and the internal mutation, the

latter as far as the propositions belong to the state base. Initially, the state is equal to IS.
The mutation μ at a time τ is the result of the application of the function TM to the actual action and the current state. At a time t between two successive points in time τ, $\mu(t)$ is empty.
The action α at a time τ is the set of propositions, which are the contents of the actual orders. At a time t between two successive points in time τ, $\alpha(t)$ is empty.
The reaction ρ at a time τ is the set of orders produced at that time. At a time t between two successive points in time τ, $\rho(t)$ is empty.
Because of the distinction between the communication of statements and the communication of orders, we accordingly distinguish between two kinds of influencing between systems, called conditioning and directing.

A system 1 is said to *condition* a system 2 if M1 and S2 do have a non-empty intersection. If this is the case, then every statement belonging to this intersection, produced by system 1, will change the state of system 2 instantly.

A system 1 is said to *direct* a system 2 if R1 and A2 do have a non-empty intersection. If this is the case, then every order belonging to this intersection, produced by system 1, will change the agenda of system 2 instantly.

The distinctive difference between conditioning and directing is that directing implies the triggering of a system to perform a transition, whereas conditioning does not do this. A system takes notice of communicated statements, i.e. of a state change, at some later point in time, namely when it is triggered and consequently needs to inspect its state.

3. Behaviour specification

The *specification* of the behaviour of a system consists of the specification of its five defining components: the state base S, the mutation base M, the order base A, the reaction base R, and the transition base T. The components S,M,A and R are specified by means of a proposition table, the component T is specified by means of a transition table.

3.1. The proposition table

First order logic appears to be a suited vehicle for expressing propositions, although there are alternative ways of specification. The advantage of logic is that it is well-defined and almost universally known. The major barrier to the use of logic seems to be the traditional Peano-Russell notation. Fortunately, friendlier and in

some ways better notational forms are emerging (cf. [11]). The introduction of logic in this paper is rather informal. So a basic knowledge of first order languages (cf e.g. [9]) is assumed. A thorough discussion of the specification of systems can be found in [6].
A proposition is represented in first order logic by a ground atomic formula, or *atom* for short. By means of the usual logical operators (non-elementary) formulas can be composed out of atoms. Examples of atoms are:

flight_booking(toronto, mrs. adams, 890721)
cust_order(smith, bicycle, 20)

If the constants in the argument list of an atom are replaced by variables, the atom becomes an *atom type*. Atom types thus represent proposition types. A substitution of the variables of an atom type by constants yields an atom. The resulting atom is called an *instantiation* of the atom type. Analogously we distinguish between formulas and *formula types*.

Propositions are defined by means of a *proposition table*. Such a table contains an enumeration of atom types, and an indication as to which base(s) the propositions represented by their instantiations belong. Because the action base, the reaction base, the state base and the mutation base of a system may overlap in any way, several base indications may apply to the same atom type. Furthermore, a proposition table contains a narrative description of the meaning of the propositions.

Figure 4 shows the proposition table for the traffic control example. In the column "base" the base(s) are indicated to which the instantiations of the proposition types belong: A refers to the action base, S refers to the state base etc..

PROPOSITION TABLE		SYSTEM : traffic control
base	**atom type**	**semantics**
S	clear_time(C,CT)	the time neede to clear cycle C is CT
S	green_time(C,GT)	the (standard) green time for cycle C is GT
A	let_pass(C)	a car has passed a traffic sensor in cycle C
S,M,A,R	phase (C,CLR)	the phase of cycle C is CLR (red, yellow or green)
S	yellow_time(C,YT)	the yellow time for cycle C is YT

Figure 4. Proposition table of the traffic control system

3.2. The transition table

The transition base T is specified by means of a *transition table*. Such a table comprises a set of *production rules*, each consisting of a data part and a facta part. (Note. 'Data' means 'what is given' and 'facta' means 'what is made or done'.)
The data part is further subdivided into an action part and a state part, both containing a formula type. The atom types, which figure in the action part represent proposition types, the instantiations of which are elements of the action base. The atom types, which figure in the state part represent proposition types, the instantiations of which are elements of the state base. A time reference may be added to an atom type in the state part, denoted by a negative number between rectangular brackets.
The facta part of a production rule also consists of two parts: the mutation part and the reaction part, both also containing a formula type. The atom types, which figure in the mutation part represent proposition types, the instantiations of which are elements of the mutation base. The atom types, which figure in the reaction part represent proposition types, the instantiations of which are elements of the reaction base. To every atom type in the reaction part a positive time delay is added between rectangular brackets.
The substitution of the variables in all atom types of a production rule by constants is called an *instantiation* of the rule. This takes place when the rule is *executed*, as a consequence of a triggering of the system. In order to guarantee that the atom types in the facta part can be instantiated, it is necessary to require that every variable in the facta part also figures in the data part.

The meaning of a production rule is based on the truth values of the constituent parts. The truth value of the action part is logically derived from the action at the time of execution.The truth value of the state part is logically derived from the state immediately before the execution. If a time reference is added to an atom type, a true instantiation yields the point in time at which it was added to the state. The truth value of the data part is the conjunction of the truth values of the order part and the state part.
The meaning of a production rule can now be defined as follows: if and only if the truth value of the data part for a given instantiation is true, the truth of the facta part is enforced.
The triggering of a system generally causes a number of parallel executions of production rule instantiations.
Firstly, there may be a number of true instantiations of the same rule. This may have two reasons. One is that there are two or more simultaneous actual orders of the same type. For instance, there may be two simultaneous orders of the type "let_pass(C1)", and even also at the same time one or two orders of the type "let_pass(C2)". The other reason is that the state part may contain variables, which do not figure in the action part. In that case, every substitution of these variables yielding a true state part, results into a separate instantiation of the rule.
Secondly, there may be true instantiations of two or more rules. For instance, there may be a "let_pass(C)" order and a "phase(C,CLR)" order at the same time.

Figure 5 shows the transition table for the traffic control system (for the sake of simplicity, only half of it is shown; by exchanging C1 and C2 the other half is got). A conjunction of formulas is denoted by a "," and a disjunction by a ";". Negation, expressing the removal of a proposition, is denoted by "¬". For the denotation of constants and variables, the Prolog convention (cf. [2]) is adopted. The specification is considered to be self-explaining. Note that the occurence of an order "let_pass(C1)" leads to the execution of three rules, only one of which can succeed.

TRANSITION TABLE		PROCESSOR : traffic control	
DATA		FACTA	
action	state	reaction	mutation
let_pass(C1)	phase (C1, red), phase(C2,green)[-T], green_time(C2,GT2)	phase(C2,yellow) [+max(ε,GT2-T)]	
let_pass(C1)	phase(C1,red), phase(C2,red)[-T], clear_time(C1,CT1), green_time(C2,GT2)	phase(C2,yellow) [-T+CT1+GT2]	
let_pass(C1)	phase(C1,yellow)[-T], yellow_time(C1,YT1), clear_time(C1,CT1), green_time(C2,GT2)	phase(C2,yellow) [-T+YT1+CT1+GT2]	
phase(C2,yellow)	phase(C2,green), phase(C1,red), yellow_time(C2,YT2)	phase(C2,red)[+YT2]	¬phase(C2,green), phase(C2,yellow)
phase(C2,red)	phase(C2,yellow), phase(C1,red), clear_time(C2,CT2)	phase(C1,green)[+CT2	¬phase(C2,yellow), phase(C2,red)
phase(C1,green)	phase(C1,red)		¬phase(C1,red), phase(C1,green)

Figure 5. Transition table of the traffic control system

As an example we will elucidate the meaning of an instantiation of the first rule. It says that if a "let_pass" order is actual in cycle C1, and if at that time C1 is in its red phase and cycle C2 is in its green phase, then an order is generated which will try to change the phase of C2 to yellow. The moment at which this order will be actual depends on whether the green phase of C2 is prolonged or not. If it is, the order will be actual immediately (i.e. after the smallest possible amount of time ε). If it is not prolonged, it will be actual after the standard greentime of C2.

4. Modelling system structure

As we have seen, a system 1 and a system 2 communicate if there is a non-empty intersection of A1 and R2 or of A2 and R1, or if there is a non-empty intersection of S1 and M2 or of S2 and M1. The first case is called directing and the second one is called conditioning.
The communication relationships between systems can be made more intelligible if a set of systems is modelled as a smartienet. A *smartienet* is a network consisting of three kinds of components: processors, banks and channels. Four kinds of links are distinguished between these components. A smartienet provides a mechanical interpretation of the communication between systems.
The smartienet representation of a system consists of a processor (the kernel of the system) and a number of connected banks and channels. Figure 6 exhibits the symbolic representations of the components of a smartienet. A graphical representation of a smartienet using these symbols is called a *Communication Structure Diagram (CSD)*.

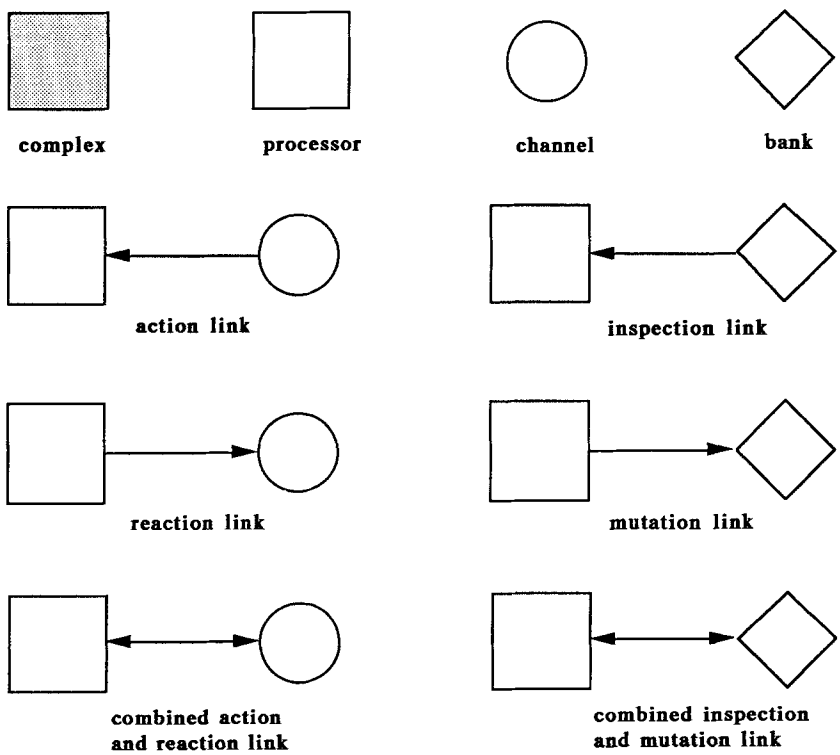


Figure 6. The components of a smartienet and the corresponding graphical symbols

Processors are executers of transitions. A processor represents the transition mechanism of a system. The operation of a processor therefore is defined by the system's *transition base*.

Banks serve to communicate statements. To this end, they are able to store statements (to be precise: the statements' contents are stored). The set of stored statements at some moment, is called the *contents* of the bank at that moment. The contents of a bank is updated and inspected by processors. A bank is defined by its *storage base* , which is the set of all propositions it is able to store as content of statements. The storage bases of the banks in a smartienet are disjoint.

Channels serve to communicate orders. To this end, channels are able to store orders (to be precise: pairs <p,t> are stored, where p is the order content and t the order time). The set of stored orders at some moment, is called the *contents* of the channel at that moment. The contents of a channel is updated and inspected by processors. When an order becomes actual the contained proposition is 'emitted'. A channel is defined by its *emission base*, which is the set of all propositions it is able to emit as content of orders. The emission bases of the channels in a smartienet are disjoint.

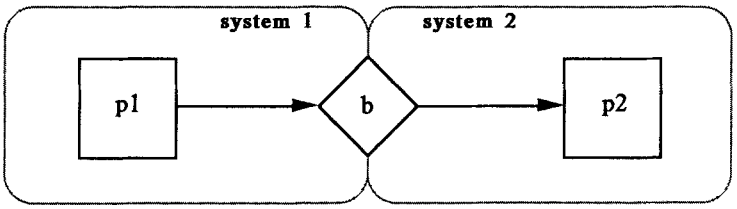
The being disjoint of the storage bases, the emission bases and the transition bases (cf. section 3) illustrates the conceptual quality of the smartienet. The important thing at the conceptual level is to discover and to show the essentially different kinds of communication among systems. Whether one chooses, for good reasons, to implement a transition base by means of multiple identical information processors, or to allow duplicate messages and data files for the implementation of channels and banks, is irrelevant from the conceptual point of view.

The reaction base of a system is equal to the union of the emission bases of its reaction channels. The action base of a system is equal to the union of the emission bases of its action channels. The action for a system at some moment consists of the union of the propositions emitted simultaneously by its action channels.
It is possible that a channel is action channel and reaction channel of a system as well. This case is called *self-directing* : the system is able to cause its own future transitions. In this way, for example, periodic activities can be modelled.

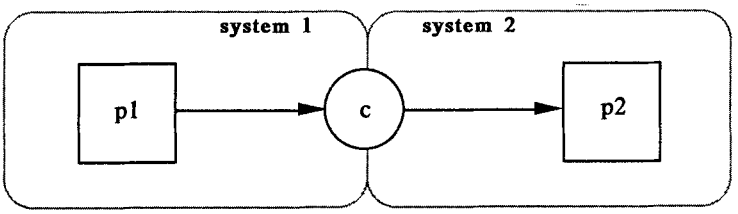
The mutation base of a system is equal to the union of the storage bases of its mutation banks. The state base of a system is equal to the union of the storage bases of its inspection banks. The state of a system at some moment consists of the union of the contents of its inspection banks.
It is possible that a bank is inspection bank and mutation bank of a system as well. This case is called *self-conditioning* : the system is able to inspect statements, which are produced by itself. (Note. Self-conditioning is the classical concept of the (internal) state of a system. End note.)

The CSD's of the conditioning and directing relationships between systems are pictured in figures 7. Processor 1 and processor 2 are the kernels of respectively a

system 1 and a system 2. Channel c is called a *reaction channel* of system 1 and an *action channel* of system 2. Bank b is called a *mutation bank* of system 1 and an *inspection bank* of system 2.



conditioning processor p1 conditions processor p2 through the interface bank b



directing: processor p1 directs processor p2 through the interface channel c

Figure 7. The conditioning and directing relationships between systems

Figure 8 shows the CSD of the traffic control world. A world is defined as a system plus its environment. The environment of a system consists of the systems with which an interface exists. Two environmental systems are identified : "traffic" and "traffic control supervisor". The latter should be understood as the responsible person or organizational function. The system "traffic" directs the system "traffic control" by means of "let_pass" orders. The traffic control supervisor conditions traffic control by stating global control parameters like e.g. the duration of the yellow period. Traffic control conditions the traffic with respect to the phase each of the cycles is in. Apparently, it also uses this information for its own operation. The traffic control system directs itself through the local channel "phase changes". The bank "parameters" contains statements of the type clear_time(C,CT), green_time(C,GT) and yellow_time(C,YT). The bank "phase" contains statements of the type phase(C,CLR).
The channel "let_pass" contains orders of the type let_pass(C) and the channel "phase changes" contains orders of the type phase (C,CLR).

The conceptual level of the SMARTIE modelling approach is apparent from the diagram of figure 8. For instance, the conditioning relationship from "traffic control" to "traffic" and to itself through the bank "phase" does not say anything, even not for the logical level, about how appropriate information flows, processes

and stores should be designed in order to realize this relationship. A similar remark holds for the directing relationship between these two systems through the channel "let_pass". Of course, one is inclined to imagine respectively traffic lights and sensors in the road surface. However, it is important to realize that such imaginations are not suggested in any way by the smartie model or by the graphical expression of its structure in the form of the CSD in figure 8.

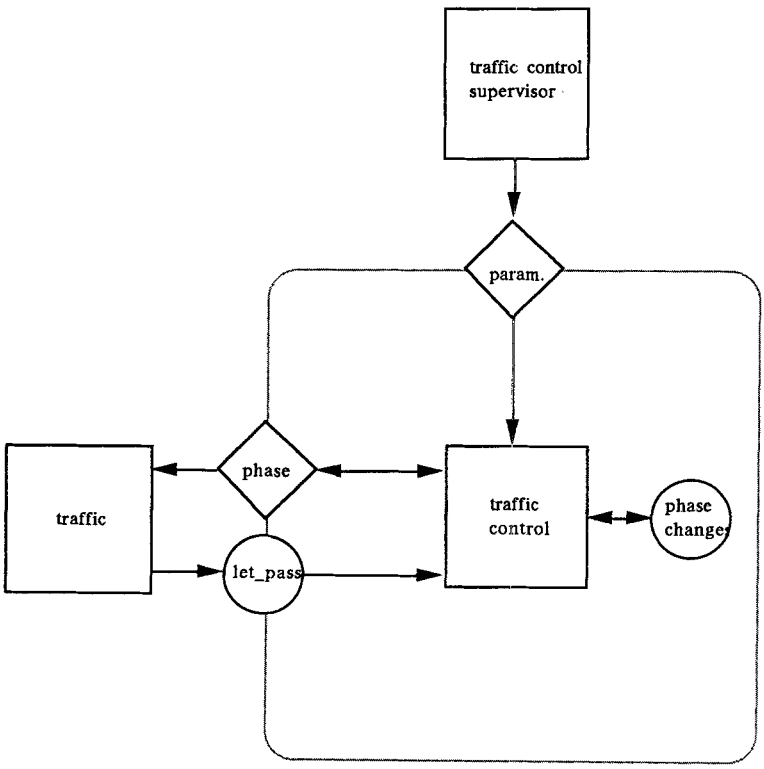


Figure 8. CSD of the traffic control world

5. Evaluation and conclusions

We have presented, be it in a concise way, the SMARTIE approach to systems modelling, and we have elaborated to some extent the specification of the behaviour of an informaton and the modelling of its structure.
It has been demonstrated that the modelling approach produces system models, which can rightly be called conceptual according to the definition of communication models provided in section 1. The clear distinction between conditioning and

directing as two essentially different kinds of communication between systems appears to lead in an almost natural way to true conceptual modelling. Discussions with many professional systems analists and designers affirm this.

A convincing way of introducing a new method or approach is to compare it with existing ones, especially well-known ones, and to reveal in doing so its strength and weakness. Comparing two things however can only make sense if they are both well defined and consequently well understood. With regard to modelling approaches this should to our opinion mean that the semantics of the key concepts and constructs are formally defined. It is therefore very unfortunate that the most well-known and widespread modelling approaches c.q. development methods, like Structured Analysis and Structured Design [5,12] and JSD [8], lack such a formal definition. An attempt to compare the SMARTIE approach with these and with others objectively, as we have undertoken in [4], very soon comprises subjective elements, just because one can not escape the necessity to interpret narrative descriptions.
The choice of a graphical representation technique for a model of whatever type is ultimately of course very unimportant. If for instance one dislikes the CSD symbols one is free to replace them by other symbols. The only important thing is how well the semantics of a diagram are defined. Whatever attractive properties the DFD technique may possess, it does not possess well defined semantics.

The SMART automaton presented offers a clear definition of system dynamics: a system can only be triggered to become active by the occurrence of an order, and this will always be the case if the proposition contained belongs to the action base of the system. Never can a statement cause the performance of a transition. A precise definition and specification of system dynamics is of utmost importance for real-time systems. So much the worse is it to see that the most widely used technique for modelling dynamics in this area is an extended version of the DFD technique. The extension consists mainly of the addition of control flows [5,12]. The sad thing is that in both publications the distinction between triggering information and non-triggering information (our distinction between directing and conditioning) does not correspond with the distinction between control flows and data flows, as one, apparantly naively(?), would expect.

The development of a practical method for the modelling and specification of systems is subject of current research. Future research topics include the formal derivation of system properties, and the development of suitable tools. Presently, only a prototype of a behaviour simulator in Prolog is available.

References

1. Austin, J.L., *How to do things with words*,
Harvard University Press, Cambridge MA, 1962.

2. Bratko, I., *Prolog Programming for Artificial Intelligence*, Addison-Wesley Publ. Comp., 1986.

3. Dietz, J.L.G., A communication oriented approach to conceptual systems modelling, in: *Proceedings of the Int. Working Conf. on Dynamic Modelling of Information Systems*, Noordwijkerhout, The Netherlands, april 1990.

4. Dietz J.L.G., Houben, G-J., *A new system concept for conceptual systems modelling*, Research Paper, Eindhoven University, Netherlands, Dept. of Informatics, 1987.

5. Hatley, D.J., Pirbhai, I.A., *Strategies for Real-Time System Specification*, Dorset House Publishing, New York, 1987.

6. Hee, K.M. van, Houben, G-J., Dietz, J.L.G., Modelling of discrete dynamic systems; framework and examples, in: *Information Systems*, vol 14, no. 4, 1989.

7. Hopcroft, J.E., Ullman, J.D., *Introduction to automata theory, languages, and computation*, Addison-Wesley Publ. Comp., Inc., 1979.

8. Jackson, M., *System development*, Prentice-Hall International, 1983.

9. Lloyd, J.W., *Foundations of Logic Programming*, Springer Verlag, New York, 1984.

10. Searle, J.R., *Speech Acts*, Cambridge University Press, Cambridge, 1969.

11. Sowa, J.F., *Conceptual structures: Information Processing in Mind and Machine*, Addison-Wesley Publ. Comp., 1984.

12. Ward, P.T., Mellor, S.J., *Structured Development for Real-Time Systems*, Prentice-Hall Inc., 1985.