A modelling method using Movie SimCon/ExSpect

Eindhoven University of Technology

Department of Mathematics and Computing Science

A modelling method using MOVIE SimCon/ExSpect

by

T.H.W.Beelen W.J.J.Stut P.A.C. Verkoulen

92/26

Computing Science Note 92/26
Eindhoven, December 1992

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. F. van Neerven
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.

A modelling method using MOVIE and SimCon/ExSpect

T.H.W. Beelen[1] W.J.J. Stut Jr.[2] P.A.C. Verkoulen[1]

[1] Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
NL 5600 MB Eindhoven
the Netherlands
E-mail: {theo,verkoulen}@win.tue.nl

[2] Central Development and Support Group Hospital Information System
BAZIS
P.O. Box 901
NL 2300 AX Leiden
the Netherlands

Abstract

Data and behaviour aspects of information systems are often modelled separately. In this paper a modelling method is introduced that starts up the integration of two specific techniques that deal with these different aspects. MOVIE, developed at BAZIS, is a *semantic data modelling* technique that models the static structure of an information system at a conceptual level. SimCon/ExSpect, developed at Eindhoven University of Technology is an executable specification framework that describes both static and dynamic aspects of a system. In this paper the transformation of MOVIE into SimCon/ExSpect is introduced using a number of transformation rules. A considerable part of the transformation can be done automatically, i.e. by a computer. However, in order to get a meaningful and executable specification the role of an human expert is indispensable.
In the approach described in this paper, we cover the modelling process from the early conceptual phase to the generation of an executable prototype that can be used for validating the design.

1 Introduction

In the field of information system development *data modelling* has been the main topic of research. Data modelling basically concentrates on the organisation of a database, i.e. modelling data and their mutual relationships. For that purpose, relational, hierarchical, and network models gained wide acceptance. In order to provide more powerful abstractions for the specification of database schemas, (formal) semantic data models have been presented (e.g. the Entity-Relationship Model [Chen, 1976], NIAM [Nijssen and Halpin, 1989] and MOVIE [Stut jr. *et al.*, 1992]).

Less attention has been paid to modelling the *behaviour* of information systems. Traditionally, behaviour modelling has mainly been done for reactive and real-time systems. For that purpose a number of modelling techniques have been introduced, such as CCS [Milner, 1980] and (high-level) Petri Nets [Reisig, 1985]. As the specification of behaviour is also important for data-intensive applications, an increasing interest in behaviour modelling of information systems can be noticed. For more details about several data and behaviour modelling techniques, a good overview can be found in [Brinkkemper, 1990].

In order to promote consistency and comprehensibility between a data model on the one hand and a behaviour model on the other hand, much research is conducted towards the *integration* of existing data and behaviour modelling techniques. An example of such research is [Sernadas *et al.*, 1991].

In this paper, a new modelling method is presented which integrates data and behaviour modelling at a conceptual level. The method is based on MOVIE, a new formalism extending semantic data modelling, and the integrated SimCon/ExSpect framework [Houben and Verkoulen, 1991; Hee and Verkoulen, 1992], a formalism extending Coloured Petri Nets. An overview of the method is given in Figure 1.

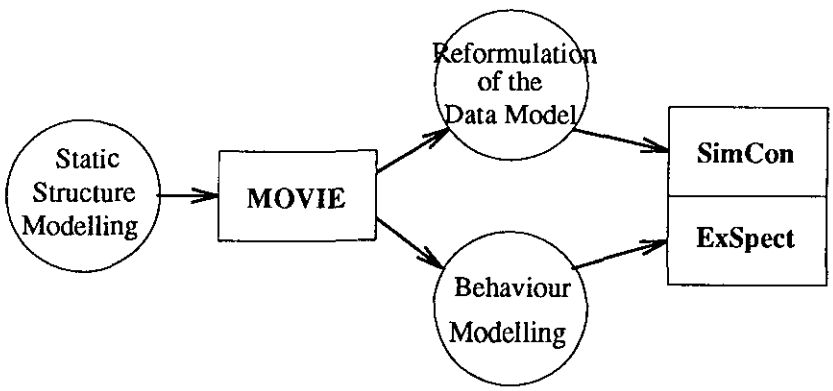


Figure 1: Overview of the modelling method.

As a first step MOVIE is used to specify the *static structure* of the system. A MOVIE model not only includes the data objects and their mutual relationships, but also allows to specify the so-called *active objects* that act on the data objects, their so-called *activity relationships*, as well as the objects that influence the active objects.

In the second step, the MOVIE model is transformed into a SimCon/ExSpect specification. This transformation addresses both the specification of behaviour in terms of *transitions* and *places*, and the reformulation of MOVIE's data relationships. The MOVIE's data rela-

tionships will be transformed to the SimCon part of the framework mentioned above. SimCon allows to describe the system at a rather abstract level, without bothering about implementation aspects in an early stage. The integrated SimCon/ExSpect framework provides an executable specification.

This paper focusses on the transformation of MOVIE into SimCon/ExSpect. The remainder of this paper is organised as follows: first a brief introduction in the formalisms MOVIE, SimCon/ExSpect is given in Section 2. Next, the transformation is presented in terms of transformation rules in Section 3. Finally, conclusions are stated in Section 4. For an extensive discussion of MOVIE we refer to [Stut jr., 1992]. A formal and more detailed description of the transformation can be found in [Beelen, 1992].

2 Modelling Techniques

2.1 MOVIE

MOVIE (Modelling Object Views) is a formal semantic modelling technique that models the static structure of a system. This structure is described by objects and relationships between these objects. MOVIE distinguishes active and passive objects, as well as activity and passivity relationships. Activity relationships model activities of active objects. Passivity relationships model static properties of both active and passive objects. Passive objects distinguish from active objects by autonomous behaviour. An object connected to a relationship denotes that this object plays a (unique) role w.r.t. this relationship. Each activity relationship has exactly one active object that plays the role of *agent*. The agent object controls the activity represented by the activity relationship.

A MOVIE model is decomposed in *fragments*. A fragment models the *view* on the system of one single object or relationship. This object or relationship is called the *central element* of the fragment. Formal consistency rules take care of the syntactical correctness of the model. Furthermore, structure rules restrict the fragment to those objects and relationships that model only the relevant part of the system w.r.t. the central element.

Objects or relationships that have an identical view on the system belong to the same *type*. A complete MOVIE model consists of a set of representative fragments, one fragment for each (object or relationship) type.

Figure 2 shows the fragment of the object type Secretary. The (active) object secretary of type Secretary calls a visitor of type Patient from a waitingRoom of type Room. An other activity done by secretary is inspecting the patient's idCard of type IdCard. The secretary wears a badge also of type IdCard. Note that each rectangle models one single object of a certain type and each diamond-shaped box models one single relationship between individual objects.

Furthermore, MOVIE provides some *advanced features* that enable a more concise notation. In particular, MOVIE provides aggregation of isomorphic structures, multiple object types, partial fragments, (multiple) inheritance, re-use of large structures and abstraction.

2.2 SimCon/ExSpect

The ExSpect (Executable Specification Tool) part of the integrated SimCon/ExSpect framework is a specification formalism for distributed systems. The description of the interaction structure is based on Coloured Petri nets [Jensen, 1991], extended with the concept of time.

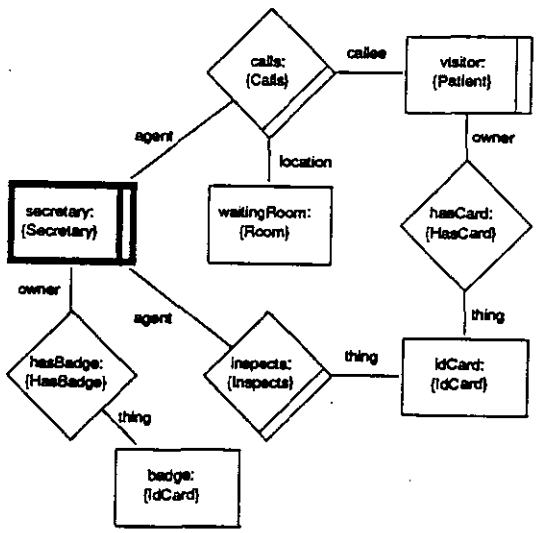


Figure 2: Fragment of the secretary

The ExSpect model, describing the net-structure and system components, consists of two kinds of basic components: *processors* and *channels* (corresponding with *transitions* and *places* in Petri net terminology). Channels may contain (typed) *tokens*. Tokens are produced and consumed by processors (firing). A token has a *time-stamp* which indicates the time before which the token cannot be consumed. The basic components are extended by three "derived" components: *stores, systems* and *pins*. A store is a channel that contains exactly one token. A system is an aggregated structure of processors, subsystems, channels stores and pins. A system can be specified as a hierarchy of subsystems. A pin, specified in a subsystem, can be considered as a reference to a channel of the system lying above the subsystem.

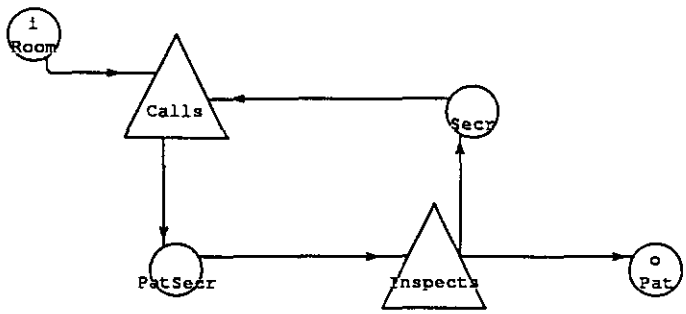


Figure 3: Graphical representation of the secretary activities

Figure 3 shows the graphical representation of the specification of the secretary activities. Processor Calls is connected to input channel Secr, input pin Room and output channel PatSecr. As soon as a "Patient-token" arrives in channel Room and a "Secretary-token" is present in channel Secr, the activity Calls can be executed. As a result of this execution, a

"Patient-token" is produced in channel PatSecr. This token represents the patient called by the secretary and the secretary who actually called the patient.

The global state space of a system , i.e. the token sets contained in the ExSpect channels is modelled by the SimCon part of the SimCon/ExSpect framework. SimCon (Simple Integrated Model for Complex Object Networks) describes the static structure in terms of *simple objects*. A simple object represents an entity of the system. For this representation a number of aspects of a simple object are relevant: an object identity, a number of attribute values, and a number of relationships with other simple objects. As usual, simple objects with similar properties constitute a *simple type*. The model also includes the notion of *inheritance*. Attributes model properties of simple objects. Attributes are defined using the type system of ExSpect[1].

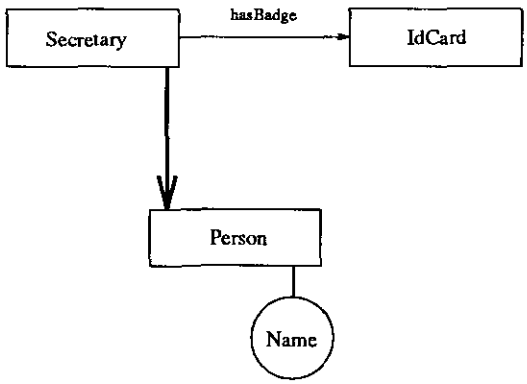


Figure 4: Example of a Simple Schema

Like in many models, *schemas* are used to define types. In Figure 4, a *simple schema* specifies the simple types Secretary, Badge and Person, and for each type, attributes (e.g. Name) and relationships (e.g. hasBadge) with other simple types are specified. Relationships between simple types are multi-valued. Moreover, the inheritance relationships are specified in the schema by means of a boldface arrow (a Secretary is a Person). A simple schema induces a *universe* of allowed simple objects.

Additional support for specifying complex objects including dynamic aspects like location and time is given by means of *container objects*. A container object is a set of simple objects. Besides that, a number of other aspects is relevant: A *location* states where a container object resides in the system. A *time-stamp* models the time at which an object is available; the simple objects from a container object are all at the same time at the same place. Just like with simple objects, container objects with similar properties constitute a *container type*.

In the integrated SimCon/ExSpect framework, a container type is associated with each channel. This means that the tokens that reside in that channel are objects of that particular container type. A part of the simple schema is thus associated with each channel. In order to be able to define container types, the schema concept is extended. An *object schema* not only includes a simple schema, but also a set of container types together with their structures, a set of location names, and a set of time-stamps. In the same way as for a simple schema, an

[1]This type system is standard. It has some basic types (e.g. num for integers) and some type constructors (e.g. power-set, tuples).

object schema induces a *universe* of allowed container objects.

Furthermore, the SimCon/ExSpect framework contains a set of tools: a graphical editor to design a specification, a type checker to check the consistency of a designed specification, an analysis tool and an interpreter to execute a specification [Broek and Verkoulen, 1992].

3 Transforming MOVIE into SimCon/ExSpect

The modelling process as described in Section 1 is presented as an interactive process between an automatic translator (computer) and an integrator (human expert w.r.t. the modelled system)[2]. The translator generates ExSpect-code by applying a number of transformation rules to a MOVIE model. The translator is not able to generate a complete specification of the modelled system, because a MOVIE model contains insufficient information. It is the task of the integrator to add missing information (e.g. the order of activities). In fact, this addition is a further specification of the system to be developed. So the integrator is in fact the modeller who is detailing the first conceptual specification.

This section informally introduces the transformation rules. Each transformation rule is explained in the following way. First, the rule is motivated. Next, an informal description of the transformation rule is given. The transformation rule is illustrated by applying the rule on the secretary fragment in Figure 2. Finally, alternatives of the transformation rule, the role of the integrator w.r.t. the transformation rule and semantical aspects are discussed. At the transformation we assume that the MOVIE model is consistent and complete (i.e. the model contains a fragment for each object and relationship type that may occur in the set of fragments).

3.1 Behaviour Modelling

Types

Both MOVIE and ExSpect classify objects that have the same properties by means of object types. MOVIE distinguishes (active and passive) object types and (activity and passivity) relationship types.

Rule 3.1 The abstract object types of a MOVIE model are directly translated to ExSpect types[3]. □

The object types Secretary and IdCard (see Figure 2), for example, are translated to the ExSpect types Secretary and IdCard respectively. They can be converted by the integrator to basic ExSpect types as follows:

```
type Secretary from str;
type IdCard from num;
```

Note that only object types are translated to ExSpect, not the objects. The objects themselves are transformed into a set of tokens of the ExSpect specification. Tokens have to be added to the specification in order to realize a meaningful execution. The tokens of the

[2] You could also call him a *modeller*, as he has to add information, thus making a specification more specific.

[3] In section 3.2, we will see a transformation directly in terms of SimCon. Such transformation is on a higher level of abstraction.

ExSpect specification cannot be derived from the MOVIE model; A MOVIE fragment only models the view of one representative object of an object type! The integrator has to take care of the assignment of tokens to channels.

Note also that the relationship types are not directly transformed to ExSpect types. The transformation of activity relationships (and their types) is explained below. The transformation of passivity relationship types is discussed in Section 3.2.

Activity Relationships and Related Objects

Activity relationships model the activities of the system. An activity may change the global state of the system. Since state changes in ExSpect are established by processors, activity relationships will be transformed to processors. The transformation of an activity relationship depends on the way the activity relationship appears in a fragment of the MOVIE model. Three different appearances have to be distinguished:

1. The activity relationship is the central element of the fragment (e.g. activity relationship Calls in Figure 5).
2. The activity relationship is a primary relationship in the fragment of its agent object (e.g. activity relationship Calls in Figure 2).
3. Neither the activity relationship, nor its agent object is the central element of the fragment.

Each appearance of an activity relationship is transformed differently.

The first appearance of an activity relationship can be considered as the definition of the activity relationship. We assume that the defining fragment of an activity relationship defines the direction of the edge between the central element (activity relationship) and an adjacent object.

Rule 3.2 Each activity relationship fragment is transformed to a processor heading:

- The name of the processor is chosen equal to the relationship type.
- A formal input and output channel are introduced (by default: $Agent_{in}$ and $Agent_{out}$). The agent object type is assigned to both channels. The meaning of these channels will become clear in the paragraph dealing with the decomposition in subsystems.
- A formal input and output channel are introduced (by default: C_{in} and C_{out}). To each of these channels an abstract SimCon container type is assigned (by default: CT_{in} and CT_{out}). The abstract SimCon container types will be composed from the types of the objects that play a role w.r.t. the activity relationship. The directed edge between an object and the activity relationship defines to which container type the type of this object will be added. If an object is connected to the activity relationship by an edge pointing to the relationship, the object's type is added to the container type of the input channel. Otherwise, if an object is connected to the activity relationship by an edge pointing to the object, the object's type is added to the container type of the output channel.

□

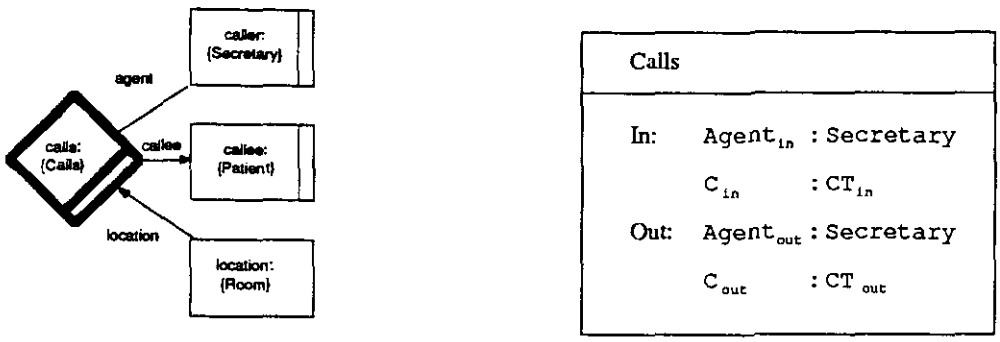


Figure 5: Translation of an activity fragment to a processor heading.

The primary objects of the activity relationship fragment (see left side of Figure 5), i.e. the objects adjacent to the activity relationship, are included in the formal parameters of the processor heading; formal input channels $Agent_{in}$ of type Secretary and C_{in} of type CT_{in} and formal output channels, $Agent_{out}$ of type Secretary and C_{out} of type CT_{out} are defined. The container types (CT_{in} and CT_{out}) are defined as indicated by Figure 6; simple type Room is contained in CT_{in} and simple type Patient is contained in CT_{out}.

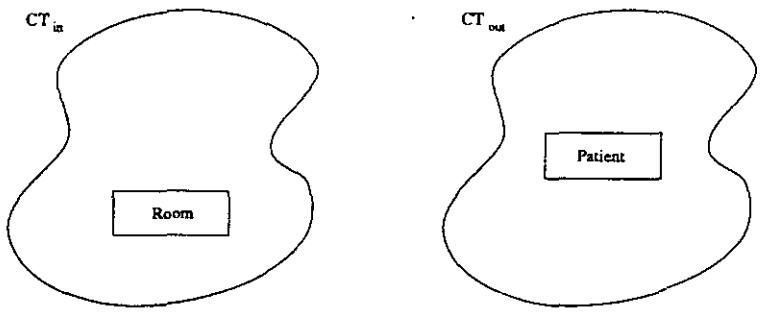


Figure 6: SimCon container types assigned to input and output channels

Alternatively, the types of non-agent objects each can be modelled in a separate channel (see [Beelen, 1992]). However, the use of complex container types provides a more convenient integration with the transformation of the passivity relationships to SimCon (see Section 3.2). Furthermore, since a processor consumes arbitrary tokens from each input channel, the use of separate channels makes it more difficult to consume *related* tokens (representing related objects).

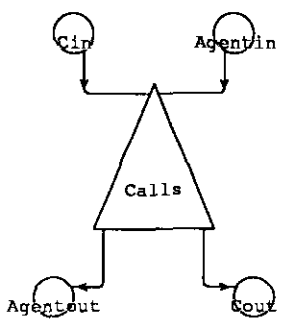
The transformation described above must be considered as a basic transformation of the objects that play a role w.r.t. an activity relationship. The semantics of an activity relationship may require that the type of an object (besides the agent object) which is connected to the activity relationship by a uni-directed edge, is added to both an input and an output container type. Furthermore, the integrator may decide to split an input (or output) container type into several input (or output) container types and to assign these types to several new formal channels (see also the paragraph dealing with the decomposition in subsystems).

Rule 3.3 A primary activity relationship in the fragment of its agent object is transformed to an installation of a processor(heading) defined by Rule 3.2:

- An actual input and output channel are introduced (by default: $Agent_{in}$ and $Agent_{out}$). The agent object type of the activity relationship is assigned to both channels.
- An actual input and output channel are introduced (by default: C_{in} and C_{out}). An abstract SimCon container type (by default: CT_{in} and CT_{out} respectively), constructed with types of adjacent objects to the activity relationship in the same way as is described in Rule 3.2, is assigned to each channel.
- The predefined processor representing the activity relationship is installed: each formal channel of the processor is connected to an actual channel of a corresponding type. Because of the consistency of the MOVIE model, there will be matching (container) types.

□

The activity relationship **Calls** appearing in Figure 2 is installed as is indicated by Figure 7. Input channels (Agent_{in} and C_{in}) and output channels (Agent_{out} and C_{out}) are introduced. Each of these actual channels is connected to the formal channel with the same name (and thus the same type).



Channel	Type
$\mathtt{Agent}_{in}$	: Secretary
$\mathtt{C}_{in}$	: $\mathtt{CT}_{in}$
$\mathtt{Agent}_{out}$	: Secretary
$\mathtt{C}_{out}$	: $\mathtt{CT}_{out}$

Figure 7: Processor installation of the activity **Calls**.

The third appearance of an activity relationship can be considered as a passivity relationship: such an activity relationship models an activity that was executed in the past. The transformation of passivity relationships is discussed in Section 3.2.

So far, no attention has been paid to the specification of the processor bodies. First, a difference of modelling activities in MOVIE and ExSpect has to be noticed: MOVIE merely models the existence of an activity and the objects that play a role w.r.t. the activity. The semantics of an activity is informally reflected by the activity relationship's name. ExSpect, on the other hand, specifies an activity by program code for manipulating the objects involved. The integrator has to translate the semantics of an activity relationship into ExSpect code (i.e. the processor body).

Decomposition into Subsystems

Due to the notion of fragments, the MOVIE model has a modular structure. In particular, the activities executed by objects of a certain type are modelled within the fragment of that object type. In order to retain the modularity, the decision has been made to decompose the ExSpect model into subsystems. More specific: the activities controlled by the same active object will be specified in the same subsystem. This subsystem will be called the "Activity Handler" of the corresponding active object type.

Rule 3.4 A subsystem ('ActivityHandler') is introduced for each active object type. □

When transforming an activity relationship, the corresponding processor is installed in the ActivityHandler-system of the agent object type. For example, the ExSpect structures obtained by Rule 3.3, representing the activities (Calls and Inspects) of the Secretary as modelled in Figure 2, are modelled in the subsystem Secretary-ActivityHandler as is indicated in Figure 8.

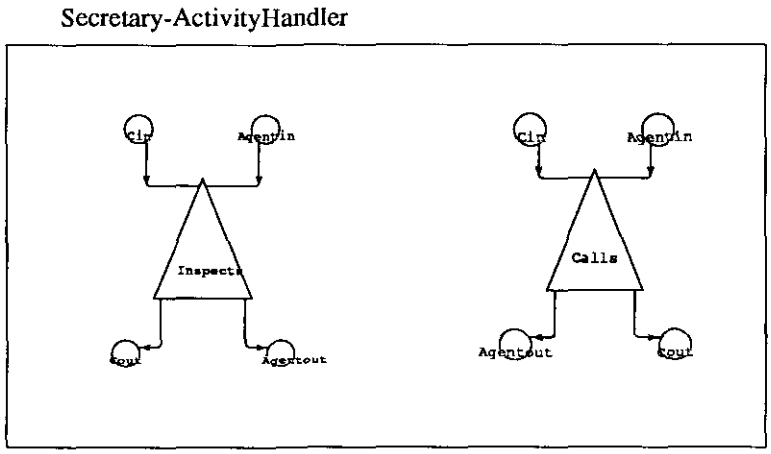


Figure 8: Subsystem representing the activities of the Secretary.

A MOVIE model does not address dynamic aspects, although this might be suggested by the directed edges. Therefore, a subsystem obtained by an automated transformation from MOVIE, does not express the order in which the activities are carried out. These dynamic aspects have to be added explicitly by the integrator, based on the semantics of the activity relationships and his or her knowledge of the system modelled.

In order to express a sequence of the activities of an object, the integrator has to put the small ExSpect structures obtained by Rule 3.3 in the correct order. To this end the agent channels are taken as a starting point: if activity A immediately precedes activity B, the output agent channel of the processor representing A, is identified with the input agent channel of the processor representing B.

When linking the other input and output channels of these processors (the "container channels" introduced in Rule 3.2 and Rule 3.3), the channels may have to be split into several container channels in order to specify dynamic aspects (such as synchronisation) correctly. This change requires a modification of both a processor definition and a processor installation.

When two or more active objects play a role w.r.t. one single activity relationship, generally communication between subsystems is indicated. In that case there are two aspects of concern.

Firstly, objects that have to be exchanged cannot be derived from the MOVIE model automatically. Candidate objects to be exchanged are those objects that play a role w.r.t. the activity relationship in question. The integrator has to make the decision which objects have to be exchanged.

Secondly, communication between subsystems takes place by means of a shared channel. Each of both subsystems has a reference (pin) to this common channel. Hence, channels generated by Rule 3.3 have to be replaced by a channel/pin-combination when the object represented by the channel is a part of the communication.

Furthermore, untyped pins (c.q. channels) may have to be added to a system in order to synchronise the activities of separate systems.

3.2 Transformation into SimCon Data Model

Passive Properties and Related Objects

The term passive properties indicates the set of passivity relationships and activity relationships used passively. The passive properties determine the state space of the ExSpect specification. The state space contains the description of the tokens in *channels* and *stores*. Tokens generally tabulate related objects, i.e. individual objects and their properties. The token of a channel can only be accessed by a processor when a connection with the store is established. Data that are globally needed through the system, can be specified in *functions* or *constants*. Each processor has access to functions and constants. The transformation of passive properties will only define the *types* of channels, stores, functions and constants. The integrator has to add the tokens and has to define the function bodies.

Alternatively, in [Beelen, 1992] a passivity relationship type is considered as a bag (multi-set) of the types of the objects that are adjacent to the relationship. Two transformation rules describe different ExSpect-implementations of such a bag. On the one hand, all properties of one single object type can be stored and, on the other hand, each single relationship type can be stored in an ExSpect component. The contribution of the integrator to this transformation of passive properties is significant: choice of ExSpect component, choice of implementation of a bag of object types. Furthermore, the transformation of passive properties strongly depends on the semantics of the relationships. A more abstract and convenient way of transforming passivity relationship types to ExSpect is to transform them to SimCon relationships.

The transformation of the passive properties and the objects that play a role w.r.t these relationships to a SimCon *simple schema* can be described by the following two rules:

Rule 3.5 Each object type is modelled as a simple type □

Rule 3.6 Each MOVIE relationship is modelled as a SimCon relationship. The latter connects the simple types that represent the object types that play a role w.r.t. the former. □

The passivity relationships hasBadge and hasCard relevant to the Secretary (see Figure 2) are translated to the SimCon simple schema as is shown in Figure 9. Objects playing a role w.r.t. these relationships are modelled as simple types.

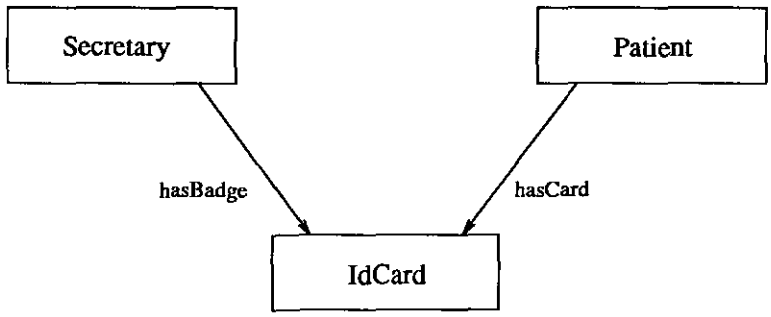


Figure 9: SimCon simple schema of the secretary's properties.

The transformation of passive properties and objects that play a role w.r.t. these relationships to a SimCon object schema is partly done by transformation Rule 3.2 and Rule 3.3.

Due to Rule 3.5, Rule 3.1 seems to be redundant. However, Rule 3.1 is not discarded in order to be able to refer to ExSpect types in Rule 3.2 and Rule 3.3. Note that some relationships in the MOVIE model may model attributes of objects. The integrator may decide to model the objects that play a role w.r.t. these relationships as attributes in the SimCon model (e.g. has relationships). Also note that SimCon only models binary relationships. A MOVIE relationship with a degree higher than two has to be objectified (i.e. modelled as an object type) as a simple type in the SimCon model. In this case, the roles of the objectified relationship can be used to access the simple types corresponding to the adjacent objects. As an alternative of objectifying the relationship, the relationship can be split into binary relationships between the adjacent objects.

The simple schema obtained by Rule 3.5 and Rule 3.6 has to be distributed to ExSpect channels and stores (and functions and constants). The complete SimCon model can be specified in one single complex store. The token of this store specifies the entire state space and can be considered as the central database. In that case, the token is manipulated by the processors which represent the activities. An advantage of specifying the database in one single store is that data are not distributed. A disadvantage is the complexity of the token and functions to manipulate the token. The integrated SimCon/ExSpect framework has to take care of the partitioning of the simple schema into smaller structures, such that each partial structure can be assigned to a channel or store. These channels and stores have to be correctly distributed over the subsystems. Partial structures also can be added to an existing container type defined by Rule 3.2 and Rule 3.3.

The specification of an object and its properties in a channel (instead of a store) may cause problems in the transformation of so called *prq-paths* of which the objects are involved in different activity relationships. A prq-path is a forward path in the graph of a MOVIE fragment containing two objects (p and q) and one passivity relationship (r). A prq-path indicates the access path to be followed to obtain a property of object p. A prq-path is shown in Figure 2 by the following path: visitor-hasCard-idCard.

Based on Rule 3.2, the processors representing the activity relationships Calls or Inspects have channels of type Patient and IdCard. A solution for linking these channels can be established in three different ways:

1. by an additional processor
2. by the processor representing the activity in which object p is involved
3. by the processor representing the activity in which object q is involved

Case 1 requires both an additional processor and an additional channel. Case 1 keeps the flow of objects through the (sub)system clear to the user. The disadvantage of an additional processor is a decreasing performance of an execution caused by the increasing number of processors.

When the derivation step is not established by an additional processor, it has to be established by one of the processors representing the activity relationships adjacent to the prq-path (Calls and Inspects in Figure 2). The type of the channel that connects these two processors is determined by the processor that derives the property. In case 2, the type of the connecting channel equals to the type of object q (type IdCard in the secretary case); in case 3, the type of the connecting channel equals to the type of object p (type Patient in the secretary case). The semantics of the property and its interpretation by the integrator determine which case will be chosen to establish the derivation step.

4 Conclusion

The transformation rules as described in Section 3 of this paper define the global structure of the specification of the modelled system. A considerable part of an arbitrary MOVIE model can be translated to the integrated SimCon/ExSpect framework by an automatic translator (i.e. computer tool). The definition of the following components can be established by an automatic translator:

- object types;
- processors (heading and installation);
- (sub) systems;
- abstract (complex) types for stores and channels.

The transformation rules provide maintenance of objects and relationships that are related. The rules also provide a ExSpect specification decomposed in subsystems. The modelling process seems to be convenient according to the transformation rules. However, the completeness of the transformation rules can hardly be proved. The informal role of the integrator w.r.t. the modelling process takes care of the completeness. In order to complete the components generated by the automatic translator to an executable and meaningful specification, the role of the integrator is indispensable. Among others, the integrator has to take care of the following aspects:

- ordering of processors;
- communication between processors (and subsystems);
- functional aspects (processor and function bodies).

According to [Beelen, 1992], the advanced features of MOVIE can also be transformed in a straightforward way.

This paper focussed on the syntactical description of the transformation process from MOVIE into the integrated SimCon/ExSpect framework. An other important aspect that has not been considered so far, is the consistency of the underlying semantics of both models. Future research w.r.t. an integration of MOVIE and SimCon/ExSpect might focus on semantical aspects of the integration. Also optimisation of generated ExSpect code might be discussed. This future research may then bring code generation within reach.

References

[Assche *et al.*, 1991] F.J.M. van Assche, B. Moulin, and C. Rolland, editors. *The Object Oriented Approach in Information Systems*, IFIP TC8 Working Conference, Quebec, Canada, 1991. North-Holland.

[Beelen, 1992] T.H.W. Beelen. An integration of MOVIE and ExSpect. Master's thesis, Eindhoven University of Technology, June 1992.

[Brinkkemper, 1990] S. Brinkkemper. *Formalisation of Information Systems Modelling.* PhD thesis, University of Nijmegen, 1990.

[Broek and Verkoulen, 1992] E.M.M.A. van den Broek and P.A.C. Verkoulen. A Tool for Integrated Modelling of Static and Dynamic Aspects of Systems. In K. Lyytinen and V-P. Tahvanainen, editors, *Next Generation CASE Tools*, pages 75–98. IOS Press, 1992.

[Chen, 1976] P.P. Chen. The Entity-Relationship Model: Towards a Unified View of Data. *ACM Transactions on Database Systems*, 1:9–36, January 1976.

[Hee and Verkoulen, 1992] K.M. van Hee and P.A.C. Verkoulen. Data, Process and Behaviour Modelling in an Integrated Specification Framework. In H.G. Sol and R.L. Crosslin, editors, *Proceedings of the Second International Conference on Dynamic Modelling of Information Systems*, Washington, D.C., USA, March 1992. North-Holland.

[Houben and Verkoulen, 1991] G.J. Houben and P.A.C. Verkoulen. An Integrated Approach to Modelling Structural and Behavioural Aspects of Complex Objects. In J. Göers, A. Heuer, and G. Saake, editors, *Third International Workshop on Foundations of Models and Languages for Data and Objects*, volume 91/3 of *Informatik Bericht*, pages 47–64, Aigen, Austria, September 1991. Technische Universität Clausthal.

[Jensen, 1991] K. Jensen. Coloured Petri Nets: A High Level Language for System Design and Analysis. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 342–416. Springer-Verlag, New York, 1991.

[Milner, 1980] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, New York, 1980.

[Nijssen and Halpin, 1989] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: A Fact Oriented Approach.* Prentice-Hall, 1989.

[Reisig, 1985] W. Reisig. *Petri Nets: an Introduction.* Prentice-Hall, 1985.

[Sernadas *et al.*, 1991] C. Sernadas, P. Resende, P. Gouveia, and A. Sernadas. In-the-large object-oriented design of information systems. In [Assche *et al.*, 1991], 1991.

[Stut jr. *et al.*, 1992] W.J.J. Stut jr., M.R. van Steen, L.P.J. Groenewegen, and A.R. Bakker. The MOVIE Modelling Technique. In J.L.G. Dietz, editor, *Computing Science in the Netherlands*, pages 304–315, 1992.

[Stut jr., 1992] W.J.J. Stut jr. *Constructing Large Conceptual Models with MOVIE.* PhD thesis, BAZIS Leiden, 1992.

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.
91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.

91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.
91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.

92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten J.A.Bergstra S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.
92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.

92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.