The Conceptual Task Model: a Specification Technique between Requirements Engineering and Program Development (Extended abstract)

S. Brinkkemper° *and A.H.M. ter Hofstede*°

* Software Engineering Research Centre, P.O. Box 424,
3500 AK Utrecht, the Netherlands;
° Department of Information Systems, University of Nijmegen,
Toernooiveld, 6525 ED Nijmegen, the Netherlands

ABSTRACT

In current practice of information system development, as well as in its support tools, there exists a gap between the informal requirements engineering activities and the more formal program development stage. To overcome this, a specification technique, called the Conceptual Task Model (CTM), is introduced, that is related explicitly to the results of the global requirements specification, i.e. process models and data models, and that can be input to code generation. The CTM technique is based on and defined in terms of Predicate\transition nets. CTM integrates the specification of the data manipulation function with control structures and local and global data models. The possibilities for the automated support of CTM are discussed. Finally, the precise relation with the process model and some other theoretical issues are presented.

KEYWORDS:

Process model, data model, conceptual task model, predicate\transition nets, CASE-tool.

1. INTRODUCTION

The requirements engineering phase and the program development phase, as they are commonly distinguished in the information system development life cycle, do not fit to each other properly with respect to the intermediate specification of the process view of the system. Output of requirements engineering should be a formal, complete, precisely defined problem specification, from which during program development code is derived manually or generated automatically.

The requirements engineering techniques used are unfortunately of an informal and global nature in order to capture the system in a concise and comprehensible way. Data flow diagrams and Entity-Relationship diagrams in some or other notation describe the process view and the data view of the system respectively. The data models are used to generate the data definition part (DDL) of the application software. Regarding the data manipulation part (DML), the processes at the bottom level of the data flow hierarchy, the so-called *tasks* [Brinkkemper 89a], are detailed by means of pseudo coding techniques, such as mini-specs [Yourdon 79] or action diagrams [Martin 85]. Since these contain informal statements, programmers usually need additional specifications and of course, pseudo code can never be input to code generation. In practice this leads to requirements engineering specifications only being helpful to define the scope and subject matter of the project, but a transformation of the process specification to programs is not made.

The crucial problem is therefore in the specification of the tasks, the processes at the bottom level of the process model. The tasks are processing data, that in its turn is specified in the data model. The tasks are refinements of the system processes and so their decomposition and contents will result in the modules and logic of the ultimate code.

We here want to introduce a new specification technique in which parts of the requirements engineering can be specified and that can be input to code generation. We impose on such a specification technique the following requirements:

1. The technique should enable fluent transfer between the phases and steps. Cross-references between models should be explicit.

2. The technique should be complete with respect to control flow, i.e. triggers, decisions, dynamic constraints and iteration.
3. The technique should produce unambiguous models that can straightforwardly be input to code generation or programming.
4. The technique should have a sound formal theoretical basis to enable the verification of theoretical statements and the formulation of properties on models that underlie all sorts of validation analysis.
5. The technique should be diagrammatic in order to ensure fast comprehension of the models during all kinds of written and verbal communication.
6. The technique should be complete with respect to data manipulation: retrieval of (derived) data as well as updates of the data.

There are a lot of methods proposed for the specification of processes, although not especially intended to be used for task modelling. We mention here ACM/PCM [Brodie 82], REMORA [Rolland 82], IML [Richter 82], Structure Charts [Yourdon 79], Process algebra [Bergstra 86], JSD [Jackson 83], EXSPECT [van Hee 88] and Petri-nets [Reisig 85]. We have reviewed most of them on their applicability for task modelling by assessing the requirements above. Those existing techniques do to a large extent not satisfy all the requirements (see [Ter Hofstede 89]).

Task specification, as we propose it here using the Conceptual Task Model (CTM), continues with the results of the global process specification, for instance denoted in data flow diagrams, and the completed data models. The manipulation of the data is defined in terms of small parts of the data model, for which we use here NIAM [Nijssen 89] and RIDL [Meersman 82], but any combination of data modelling technique and data modelling language, such as for instance relational tables and SQL, could be used. The work here can be seen as an elaboration of ideas in of the work of Genrich [Genrich 87], Kung and Sölvberg [Kung 86] and Richter and Durchholz [Richter 82].

In the following chapter we will introduce the CTM formally, formulate some properties and give an example of a CTM-net. The implementation of a CTM support tool for its use in system development is discussed in chapter 3. Chapter 4 contains some theoretical issues, such as the formal correspondence of the task model with the process model. We conclude with some summarising remarks and options for further research. This work is an extended abstract of [Ter Hofstede 89], which in its turn is an extension of the research reported in [Brinkkemper 89a].

2. THE CONCEPTUAL TASK MODEL

In this section first the Conceptual Task Model (CTM) will be defined in terms of Predicate\transition nets and an example of a CTM-net will be presented. Then the CTM will be defined formally and the example will be related to the formal definition. Based on the formal definition we can formulate some properties a correct CTM-net must have.

2.1 PrT-net basis of the CTM

One way to introduce the CTM is to base it upon the formalism of Predicate/transition nets (PrT-nets). The advantage of this proceeding is that the semantics of the CTM is then (partly) defined through the semantics of PrT-nets.

PrT-nets are introduced by Genrich and others in a series of articles, starting with [Genrich 79], and at the moment concluded by [Genrich 87]. In short, PrT-nets are interpreted, inscribed high-level Petri nets, where inscriptions consist of variables for individuals (as opposed to the non-individual token of Petri nets) and truth-valued expressions, preferably in first-order predicate logic. For a detailed treatment of PrT-nets we refer to [Genrich 87].

A CTM-net is a PrT-net where

- Instead of the formalism of first-order logical formulas and their structures, the conceptual data modelling language NIAM in combination with the corresponding data manipulation language RIDL is used as supporting structure. Functions and expressions, which can be seen as special kinds of RIDL functions, are interpreted in this structure.
- A distinction is made between task places and information places. A conceptual schema in NIAM is related to both kinds of places. Each place of the PrT-net is either a task place or an information place. The conceptual schema of an information place determines the information

structure of the tuples that can enter that place. The conceptual schema of a task place describes that part of the Universe of Discourse consisting of all the individuals of the tuples that can enter that place.

- An additional typing is related to each task place. When the arity of a task place *P* is *n*, a typing <T1,T2,...,Tn> is associated with *P* such that for every tuple <P1,P2,...,Pn> that can enter *P* we have that Pi is of type Ti (for all 1≤i≤n). The typing of a task place is a linear representation of the two-dimensional conceptual schema associated with that task place.
- Arrows may not be labeled with linear combinations of tuples, but only with single tuples.

We adopt three simplifying notational conventions. The first convention is that if we have *n* (*n* > 1) disjoint conditions (C1,C2,...,Cn), possibly combined with *m* (*m* ≥ 0) other conditions (Q1,Q2,...,Qm), then instead of having *n* separate transitions for each condition, we introduce one combined transition containing all conditions, as shown in fig. 2.1. Output arrows coming from a transition containing condition Ci are now attached to the little box containing Ci inside the combined transition.

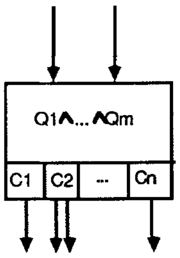


Figure 2.1 Combined transition for C1,C2,...,Cn

The second and third notational conventions are shown in fig. 2.2. These concern database I/O, which is bi-directional in the Predicate\transition formalism.

2.2 Example

In fig. 2.3, an example of a CTM-net is shown. This CTM-net calculates the rental proceeds of a film, which is defined as zero for new films and for rentable films as the number of tapes that contain that film times the rental price for that film.

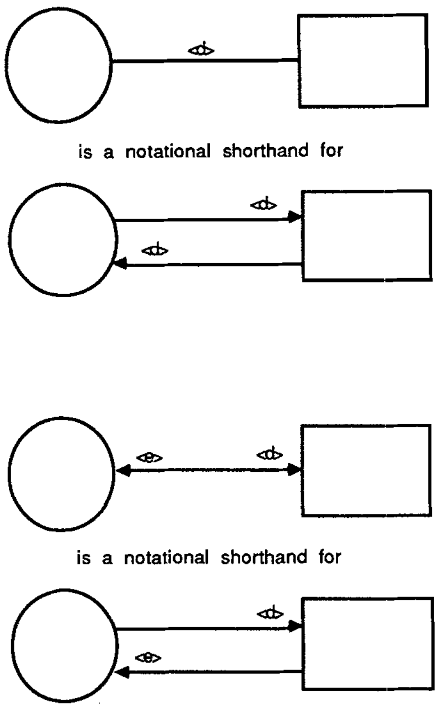


Figure 2.2 Double arrow convention for the CTM

In transition *T1* it is checked whether the film is new or not. Information of the information place *Information concerning films and tapes* is necessary to check this. If the film is new, a token consisting of that film is placed in the input place *P2* of transition *T2*. Transition *T2* then adds the current default value for new films (zero) to the tuple. If the film is not new, which is equivalent to the film being rentable, transition *T3* is enabled. Transition *T3* calculates the number of tapes that contain the processed film. Transition *T4* then searches for the rental price of the film and performs the multiplication of the number of tapes *n* and that rental price. Transition *T3* as well as transition *T4* need information from the information place *Information concerning films and tapes*. At the end of the calculation, task place *P5* will contain the film and its rental proceeds.

Near every place of the CTM-net, the corresponding conceptual schema is shown. These conceptual schemas contain information about the tuples that

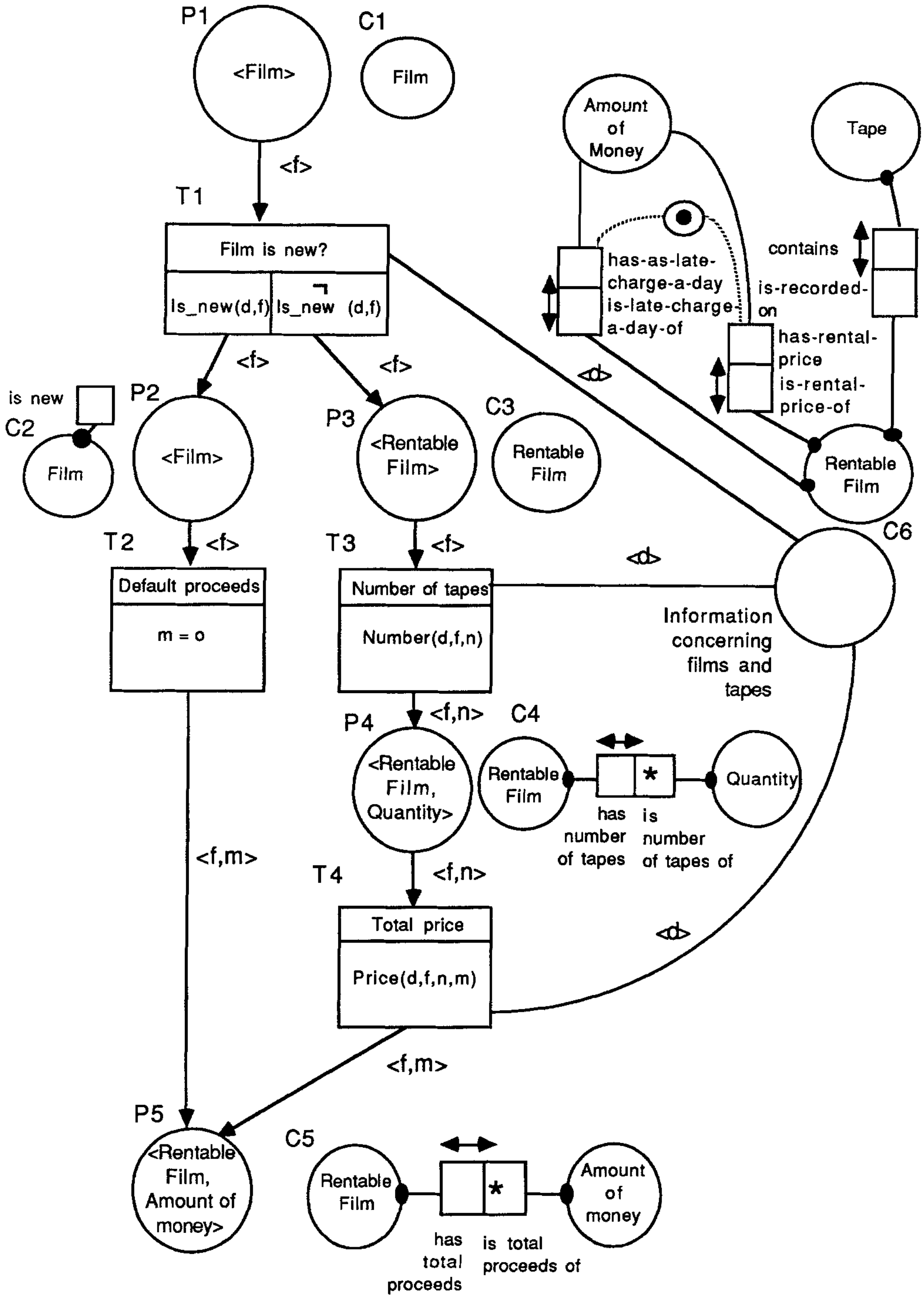


Figure 2.3 CTM-net for rental proceeds calculation

can enter the place to which the conceptual schema belongs. Conceptual schema *C2* for instance, asserts that every film that enters task place *P2* is new. The star in the conceptual schemas *C5* and *C6* denotes that the role, in which box the star is placed, is redundant.

The RIDL functions used in the transitions of fig. 2.3 are shown in fig. 2.4. In the function headings, we also showed the database on which these functions operate. In the function definitions we left this relation implicit.

```
PREDICATE Is_new (DATABASE d; FILM f);
BEGIN
        f IS NOT IN Rentable-film
END;

PREDICATE Number (DATABASE d; RENTABLE FILM f, QUANTITY n);
BEGIN
        n = NUMBER-OF Tape contains Rentable-film f
END;

PREDICATE Price (DATABASE d; RENTABLE FILM f, QUANTITY n, AMOUNT OF MONEY m);
BEGIN
        m = n * Amount-of-money is-rental-price-of Rentable-film f
END.
```

Figure 2.4. RIDL-queries belonging to CTM-net for rental proceeds calculation

Of course the CTM-net of fig. 2.3 is not the only possible solution to model the rental proceeds calculation. In fact it is a rather elaborate solution since it is possible to combine all the RIDL expressions in one transition. The disadvantage of that proceeding however is, that assertions about local information remain implicit. Another option is e.g. to have transition *T1* put a tuple *<f,0>* directly into task place *P5* if the film *f* is new. This obscures however the meaning of the zero. Guide-lines for modelling a task as a CTM-net will be reported soon.

2.3 Formal definition

After this introduction the definition of the Conceptual Task Model can be given in a formal way.

Definition 2.1 A CTM-net is a 12-tuple (Π, T, P, Σ, Ψ, Z, I, Φ, Λ, Θ, X, Ω), where

Π is a non-empty finite set of places,

T is a non-empty finite set of transitions (not combined transitions in the sense of fig. 2.1),

P is a finite set of parameterised RIDL expressions,

Σ is a non-empty finite set of conceptual schemas,

Ψ is a finite set of linear typings (a linear typing is a tuple of arbitrary length consisting of entity types),

Z is a non-empty finite set of variables,

$I \subseteq \Pi$ is a set of information places; $\Gamma = \Pi/I$ (by definition) is the set of task places,

$\Phi \subseteq \Pi \times T \cup T \times \Pi$ is a non-empty set of arrows, denoting that a place is input for or output of a transition,

$\Lambda \in \wp\wp(Z)^{\Phi}$ is a function from the set of arrows Φ to the set $\wp\wp(Z)$ of tuples of arbitrary length of variables chosen from Z, denoting the labeling of the arrows with a tuple of variables,

$\Theta \in P^{T}$ is a function from the set of transitions T to the set of parameterised RIDL expressions P, denoting which RIDL query belongs to which transition,

$X \in \Sigma^{\Pi}$ is a function from the set of places Π to the set of conceptual schemas Σ, denoting which conceptual schema belongs to which place,

$\Omega \in \Psi^{\Gamma}$ is a function from the set of task places Γ to the set of linear typings Ψ, denoting which typing belongs to which task place.

Now we relate the CTM-net of fig. 2.3 to this definition of a CTM-net. We will give examples of elements of each of the constituents of the 12-tuple:

P1,P2,..,P5 are elements of Π;

T1a, T1b and *T2* are elements of T (*T1a* and *T1b* are transitions that would become visible if we would unfold transition *T1* according to the notational shorthand of figure 2.1);

Number(d,f,n) is in P;

C1,C2,...,C6 are the elements of Σ;

<Rentable Film, Quantity> is in Ψ;

f, m and *n* are elements of Z;

Information concerning films and tapes is the only element of I, *P1* is an element of Γ;

(P1,T1) is an element of Φ, denoting the arrow going from task place *P1* to transition *T1*;

((T3,P4),<f,n>) is an element of Λ, denoting the labeling of the arrow going from transition *T3* to task place *P4* with the tuple *<f,n>*;

(T4,Price(d,f,n,m)) is contained in Θ;

(P5,C5) is an element of X;

(P5,<Rentable Film, Amount of Money>) is an element of Ω.

2.4 Properties

To formulate the properties a CTM-net must have, we introduce some auxiliary functions and predicates in an informal way. Most of these functions and predicates cannot be given here in a formal way, since we do not have a formal definition of RIDL and NIAM at hand.

The function *entity* operates on a conceptual schema and yields the set of entity types occurring in that conceptual schema,

Type_in_expression (r, v, e) is true if and only if the formal parameter *v* is supposed to be of type *e* in expression *r*,

The function *merge* operates on a set of conceptual schemas and yields the integration of these schemas,

The predicate *part_of* defines a binary relation between conceptual schemas and is true if and only if the first conceptual schema is part of the second conceptual schema,

The function *domain* operates on a RIDL expression and yields the domain (this is a conceptual schema) of that expression.

Among others, the following properties must hold for the 12-tuple:

Property 2.1

$$\forall p \in I \ \forall t \in T \ [\ (p,t) \in \Phi \Leftrightarrow (t,p) \in \Phi\ ]$$

This property states that an information place is never only input for nor only output of a transition.

Property 2.2

$$\forall p \in \Pi \ \exists t \in T \ [\ (p,t) \in \Phi \vee (t,p) \in \Phi\ ]$$

Every place is input for or output of a transition. From this property and the first property one can derive that every information place is input for at least one transition and also that every information place is output of at least one transition. It must not be forgotten however, that the set of information places may be empty.

Property 2.3

$$\forall p \in \Pi / I\ [\ \cup\, \Omega(p) = \text{entity}\ (X(p))\ ]$$

This property states that the set (not multi-set!) of entity types occurring in the typing of a task place equals the set of entity types occurring in the conceptual schema of that task place.

Property 2.4

$$\forall t \in T\ \forall v \in Z\ \forall e\ [\ \text{type\_in\_expression}\ (\Theta\ (t), v, e) \Rightarrow$$
$$\forall p \in \Pi / I\ \forall i\ [\ (((p,t) \in \Phi \wedge (\Lambda(p,t))_i = v) \Rightarrow (\Omega(p))_i = e\ ) \wedge$$
$$(((t,p) \in \Phi \wedge (\Lambda(t,p))_i = v\ ) \Rightarrow (\Omega(p))_i = e\ )]\ ]$$

This complex looking property simply states that the type of a formal parameter as can be derived from the typing of the task place to which it belongs should agree with the way this formal parameter is used (i.e. of which type it is supposed to be) in the expression of the transition to which it is a local variable.

Property 2.5

$$\forall t \in T\ [\text{part\_of}(\text{domain}(\Theta(t)), \text{merge}(\{c \mid c \in \Sigma \mid \exists\, p \in I$$
$$[(p,t) \in \Phi \wedge X(p) = c]\}))]$$

A RIDL expression in a transition should operate on the conceptual schemas of the information places connected to that transition, i.e. the domain of the RIDL expression *Θ(t)* of the transition *t* is part of the union of the conceptual schemas *c* of the information places *p* connected to *t*. This is a simple formulation of the type checking of queries and is derived from the more important rule that queries are formulated in terms of the data model. Of course more than property 2.5 can be stated about the relation of the query and the data model, but that is beyond the scope of this work. We only formulate the following simple corollary.

Corollary 2.6. The entity types in a RIDL expression are a subset of the entity types of the information places:

$\forall t \in T$ [entity(domain(Θ(t))) $\subseteq$ entity(merge($\{c \in \Sigma \mid \exists\, p \in I\ [(p,t) \in \Phi \wedge X(p) = c]\}$))]

Take the RIDL expression *Price* in fig. 2.4 as an example. This query has the types *Rentable Film* and *Amount of money* as entity types, which occur both in the conceptual schemas *C6* of fig. 2.3.

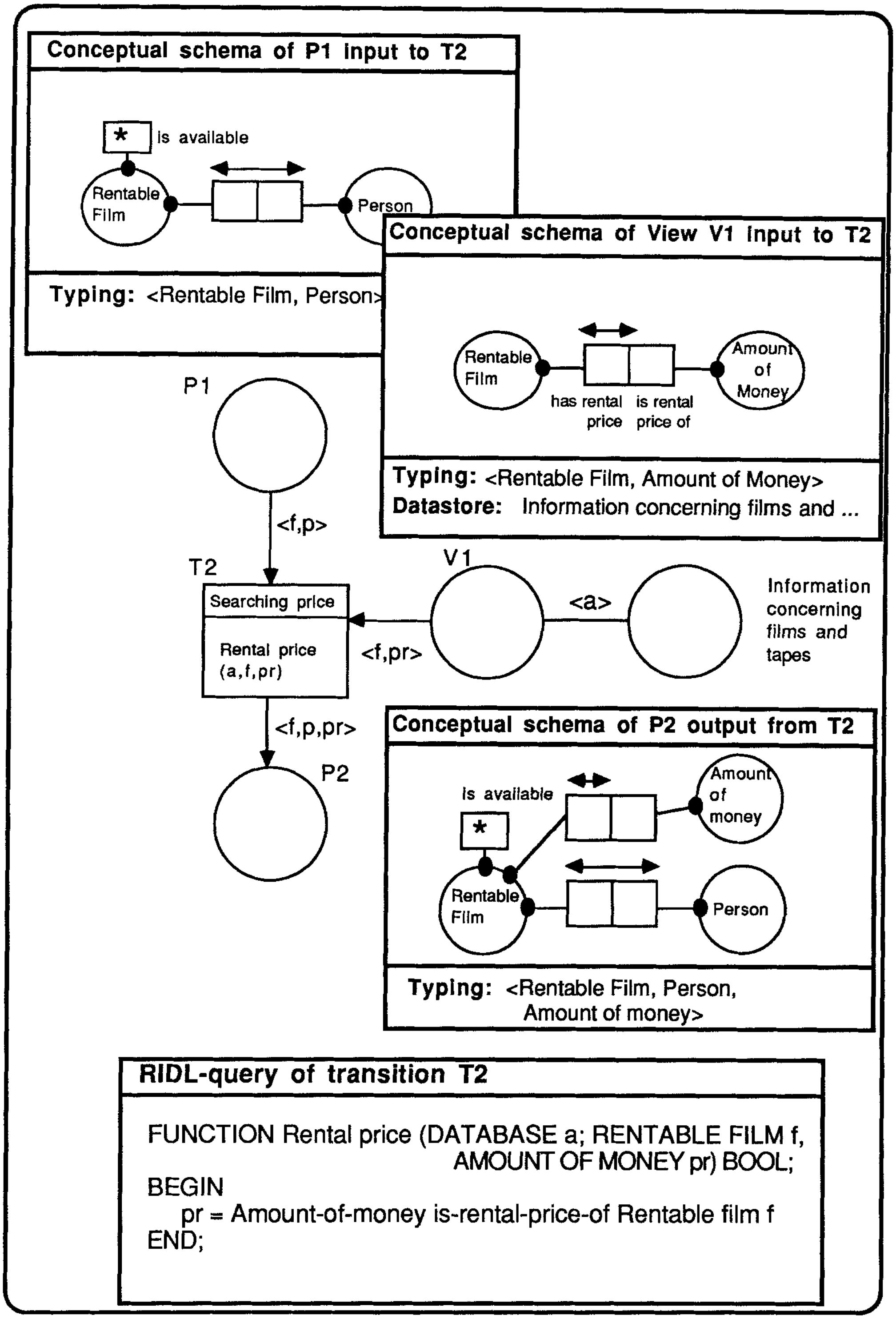
There are more properties a correct CTM-net must have. We will defer discussion of one of those properties to section 4. For a discussion of other properties of correct CTM-nets, we refer to [Ter Hofstede 89]. Worth stating here is that a CTM-net should contain the complete specification of a task. This requirement however, is not verifiable, since the completeness of a specification depends on the completeness of the informants' specification.

3. CASE TOOL IMPLEMENTATION

The CTM technique is hardly applicable in a manual way for the modelling of tasks of a realistic sized IS, due to the complexity of the resulting diagrams. Automated support of the technique in a tool, possibly combined with modelling techniques for activities, data and user interaction, is required. Properties are formulated, on which all sorts of analysis of application models can be based.

In fig. 3.1 we show a proposal for a screen layout of a tool supporting the modelling of tasks using the CTM. The data models of the places and the RIDL-queries of the transition are shown in separate pop-up windows. When these windows are left out, a plain PrT-net remains.

The tool may provide additional support for a modelling procedure in the sense as described in [Brinkkemper 88]. To be distinguished are the preliminary task modelling, identification of individual transitions, modelling of data at the places, formulation of the RIDL-queries and the checks on the components. When the transitions in a task are known, they can be put in a preliminary schema, with some intermediate task places connecting them. Those transitions can be modelled and analysed separately. After that they can be integrated for global analysis of consistency, connectivity or for other purposes.


Conceptual schema of P1 input to T2
*
is available
Rentable Film
Person
Typing: <Rentable Film, Person>
Conceptual schema of View V1 input to T2
Rentable Film
has rental price
is rental price of
Amount of Money
Typing: <Rentable Film, Amount of Money>
Datastore: Information concerning films and ...
P1
<f,p>
T2
Searching price
Rental price (a,f,pr)
<f,pr>
V1
<a>
Information concerning films and tapes
<f,p,pr>
P2
Conceptual schema of P2 output from T2
is available
*
Amount of money
Rentable Film
Person
Typing: <Rentable Film, Person, Amount of money>
RIDL-query of transition T2

```
FUNCTION Rental price (DATABASE a; RENTABLE FILM f,
                               AMOUNT OF MONEY pr) BOOL;
BEGIN
    pr = Amount-of-money is-rental-price-of Rentable film f
END;
```

Fig. 3.1 Screen layout of CTM tool

However, we propose in this section a practical diagramming convention, that deviates in two ways from the theoretical technique. This is necessary to improve the practical applicability. The discrepancies between the practical and the theoretical technique can be overcome by standard transformations, that can be derived from the descriptions below. The adaptations are the following.

First, we use data base views instead of database tuples. Since tasks need only a certain part of the data present in the data base, we define a view that models this part. This view is positioned on the arrow from the information place to the transition. An information place gets surrounded by such views. The conceptual schema of a view is a derivable part of the conceptual schema of the information place. Recall that the information places correspond with the data for which retrieval queries or update queries are formulated, whereas the data model at the task places stand for the parameters of the transition. Syntactical and semantical cross-checks of queries and parameters versus data models can be performed automatically.
All inputs and outputs of a transition are now specified by small conceptual schemas. An example of a view is shown in fig. 3.1 for the information place *Information concerning films and tapes*. The conceptual schema of view V1 is part of the conceptual schema C6 in fig. 2.3. In the PrT formalism such views are not prohibited, but the strong relation of the data in the data store with that in the data view must be described completely. This is not practical, since database management systems implement views very effectively.

Secondly, we propose for the tool implementation to support the decomposition of tasks. This decomposition obeys analogous rules as those for the decomposition of activities in data flow diagrams. The conceptual schemas at the places may also be decomposed, but the decomposition must always satisfy the requirement that tasks process data elements (see the definitions in [Brinkkemper 89a]).
In PrT-nets this is again not possible due to the unclear firing semantics of the decomposition, when data elements corresponding to more than one input or output place are optional.

Next to this and next to the discussions in the previous sections, we suggest three additional functionalities in a CTM-tool.

1. Support of modelling transparency. Because of the dependencies between the task models and the models of other types, like activity models and global conceptual schemas, developers working with the tool wish to be able to transfer directly from one type of model to the other via a dependency between the models. For example the transfer from a task model to the activity it belongs to. See [Brinkkemper 89b] for a discussion of the modelling transparency functionality of workbenches and the various degrees of it.
2. Syntactic and semantic analysis of data models and queries. As already suggested above, the presence and the type of the data that are processed or created in a transition can be analysed and compared with the queries specified. Furthermore, the violation of the constraints can be pointed out.
3. Support of re-use. A support tool can compare the patterns of the data models or of the transitions with existing models and suggest to make use of them.

For a discussion of system generation, reverse engineering and simulation in the CTM, we refer to [Ter Hofstede 89].

4. THEORETICAL ISSUES

In this section we will address some theoretical issues concerning the CTM. First the relation between the CTM and data flow diagrams is investigated. Then the computational power of the CTM is considered briefly. Finally some remarks about correctness of conceptual schemas at task places are made.

4.1 The relation between activities and tasks

Data flow diagrams generally consist of activators and flows. Flows represent information in motion, activators can be considered as functions on these flows. A well-known representative of data flow diagrams are the ISAC activity graphs or A-graphs [Lundeberg 80]. We will use an adapted version of these activity graphs here.

Definition 4.1 A data flow diagram is a 9-tuple (S, A, D, F, R, Q, U, G, H), where

S is a non-empty finite set of states,

A is a non-empty finite set of activities,

$D \subseteq S$ is a set of data stores; $E = S \setminus D$ (by definition) is a set of flows,

$F \subseteq S \times A \cup A \times S$ is a non-empty set of arrows,

$R \subseteq E \times E$ is the subflow relation,

$Q \subseteq A \times A$ is the subactivity relation,

$U \subseteq S \times A$ is the substate-activity relation,

G is a non-empty finite set of conceptual schemas,

$H \in G^S$ is a function from the set of states to the set of conceptual schemas.

In [Falkenberg 89] some of the rules are stated this 9-tuple must fulfil. An example of such a rule would be that every state has a source, which could formally be expressed as:

$\forall\, s \in S\, \exists\, a \in A\, [\, (a,s) \in F\,]$

Definition 4.2 The set of tasks Y_d of a data flow diagram $\mathcal{D} = (S_d, A_d, D_d, F_d, R_d, Q_d, U_d, G_d, H_d)$ is given by:

$Y_d = \{\, t \mid t \in A_d \mid \neg\exists\, v \in A_d [\, (v,t) \in Q_d]\, \}$

Informally, a task is an activity at the bottom level of the decomposition hierarchy, i.e. an activity that is not decomposed into other activities.

Definition 4.3 Let a be a task of a diagram $\mathcal{D}$, $a \in Y_d$, then

$W_d(a) = \{ s \mid s \in S_d \mid ((a,s) \in F_d \vee (s,a) \in F_d) \wedge (s \in E_d \Rightarrow \neg\exists\, t \in E_d [\, (t,s) \in R_d]\,)\, \}$

$W_d(a)$ is the set of states which are input for or output of the task a and do not have any subflows.

These definitions enable us to formulate the relations between data flow diagrams and CTM-nets formally.

Suppose a is a task of data flow diagram

$\mathcal{D} = (S_d, A_d, D_d, F_d, R_d, Q_d, U_d, G_d, H_d)$

So $a \in Y_d$. Let

$C_a = (\Pi_a, T_a, P_a, \Sigma_a, \Psi_a, Z_a, I_a, \Phi_a, \Lambda_a, \Theta_a, X_a, \Omega_a)$

represent the CTM-net for task a.

The relation between the data flow diagram $\mathcal{D}$ and the CTM-net C_a is then expressed via an injective function f_a from the set of input and output states of task a in the data flow diagram $W_{\mathcal{d}}(a)$ and the set of places of task a in the CTM-net Π_a:

$$f_a: W_{\mathcal{d}}(a) \rightarrow \Pi_a.$$

For this function f_a the following properties must hold:

Property 4.4 Bijective data store mapping

$$f_a \mid_{D_{\mathcal{d}}} \rightarrow I_a \text{ is bijective}$$

This property states that the restriction of f_a to $D_{\mathcal{d}}$ is a bijective mapping on I_a, i.e. every information place of the CTM-net C_a is the unique image of a data store in the data flow diagram $\mathcal{D}$ which is input or output of the task a.

Property 4.5 Consistent input property

$$\forall s \in W_{\mathcal{d}}(a)\ [(s,a) \in F_{\mathcal{d}} \Leftrightarrow \exists u \in T_a\ [\ (f_a(s),u) \in \Phi_a \wedge (f_a(s) \in I_a \Rightarrow \Lambda_a((f_a(s),u)) = \Lambda_a((u,f_a(s))))]\ ]$$

If a flow s, which is not decomposed, is input for task a in the data flow diagram $\mathcal{D}$, then there exists a transition in the CTM-net C_a which has the corresponding place $f_a(s)$ as input. If a data store s is input for task a in the data flow diagram $\mathcal{D}$, then there exists a transition in the CTM-net C_a which is connected to the corresponding place $f_a(s)$ by two arrows, one input arrow and one output arrow, with the same labeling. Conversely, if a place that is the image of a flow s, is input for a transition of the CTM-net C_a, then flow s must be input for task a in the data flow diagram $\mathcal{D}$. If a place, that is the image of a data store s, is input as well as output of a certain transition of the CTM-net C_a with both arrows having the same labeling, then data store s must be input for task a in the data flow diagram $\mathcal{D}$.

Note that when in a CTM-net a place is input as well as output of a certain transition and the arrow going from that place to the transition has the same labeling as the arrow going from the transition to that place, this means that the contents of that place is only used, not changed, by the transition.

Property 4.6 Consistent output property

$\forall s \in W_{\mathcal{d}}(a)\ [(a,s) \in F_{\mathcal{d}} \Leftrightarrow \exists u \in T_a\ [\ (u,f_a(s)) \in \Phi_a \wedge (f_a(s) \in I_a \Rightarrow \Lambda_a((f_a(s),u)) \neq \Lambda_a((u,f_a(s))))]\ ]$

The explanation of this property is analogous to the explanation of the previous property.

Property 4.7 Identical conceptual schemas property

$\forall s \in W_{\mathcal{d}}(a)\ [\ H_{\mathcal{d}}(s) = X_a(f_a(s))\ ]$

A state *s* in the data flow diagram of task *a* must be associated to the same conceptual schema as its corresponding place *$f_a(s)$* in the CTM-net.

Normally activities, flows, places and transitions can be named. In this case *s* and *$f_a(s)$* should also have the same name.

Based on the properties of the CTM and the ones specified above, some theorems can be formulated of which we present one.

Theorem 4.8 The domain of all queries of a task is specified in the conceptual schemas related to the data stores of the task.

Proof: Let *a* be an arbitrary task and let T_a be the set of all transitions of *a*.
Define $Q = \{q \in P_a \mid \exists t \in T_a\ [q = \Theta(t)]\}$. Q is then the set of all queries of the task *a*.
According to property 2.5 the domain of an arbitrary query $q \in Q$ is specified in conceptual schemas corresponding to information places p, that are input for the transition t: $(p,t) \in \Phi_a$.
From property 4.4 we deduce that this p is the image of a data store s: $p = f_a(s)$, and according to property 4.7 the conceptual schema $H_{\mathcal{d}}(s)$ of s is the same as the conceptual schema $X_a(p)$ of p. QED

In the same style it can be proven that all data stores of the data flow diagram are used by queries in the tasks and that the images of any two states related to the task *a* in the data flow diagram are connected via a path in the CTM-net.

4.2 Computational power of the CTM

There are various approaches to capture the idea of computation. The class of the Turing computable functions is an example of such an approach. The principle that Turing machines are formal versions of algorithms and that no computational procedure will be considered an algorithm unless it can be presented as a Turing machine is known as Church's Thesis or the Church-Turing Thesis [Lewis 81]. If we can prove that in the CTM one can simulate any arbitrary Turing machine, we prove in fact that the CTM can compute any computable function. In [Ter Hofstede 89] a CTM-net is presented that simulates an arbitrary Turing machine.

4.3 Correctness of conceptual schemas at task places

A conceptual schema of a task place describes that part of the Universe of Discourse of those individuals that can enter that particular task place. The conceptual schemas output of a certain transition must be derivable from the conceptual schemas at the places input for that transition and the RIDL expression belonging to that transition.

As an example consider fig. 4.1. In the simple CTM-net shown there, either the conceptual schema in task place *P2* or the conceptual schema in task place *P1* is incorrect. In the conceptual schema of *P1* we see that a manager never is a coworker and vice versa, while in the conceptual schema of *P2* we see that it is forbidden to be manager and coworker of the same project. Tuple *<e,m,p>* comes in and goes out of transition *T*, so every tuple *<e,m,p>* that enters *P2* was previously contained in *P1*. Hence the population of *P2* satisfies the constraints belonging to *P1*. The conclusion must be that one of the schemas is incorrect.

The example shown was extremely simple, in general the situation is much more complex. Places can be output of more transitions, transitions can have more input places and schemas can change due to the RIDL expressions in the transitions and the schemas in the information places. For a more detailed discussion on correctness aspects of conceptual schemas at task places we refer to [Ter Hofstede 89].

5. CONCLUSIONS

In this paper the Conceptual Task Model was introduced, which was intended to fill the gap between informal requirements engineering and program development. CTM-nets were defined as special kinds of PrT-nets and the CTM was defined formally accompanied with some of the properties of a correct

CTM-net. A CASE tool implementation of the CTM was discussed briefly and the relation between activities and tasks investigated. Finally the issues of computational power of the CTM and correctness of conceptual schemas at task places, were addressed.

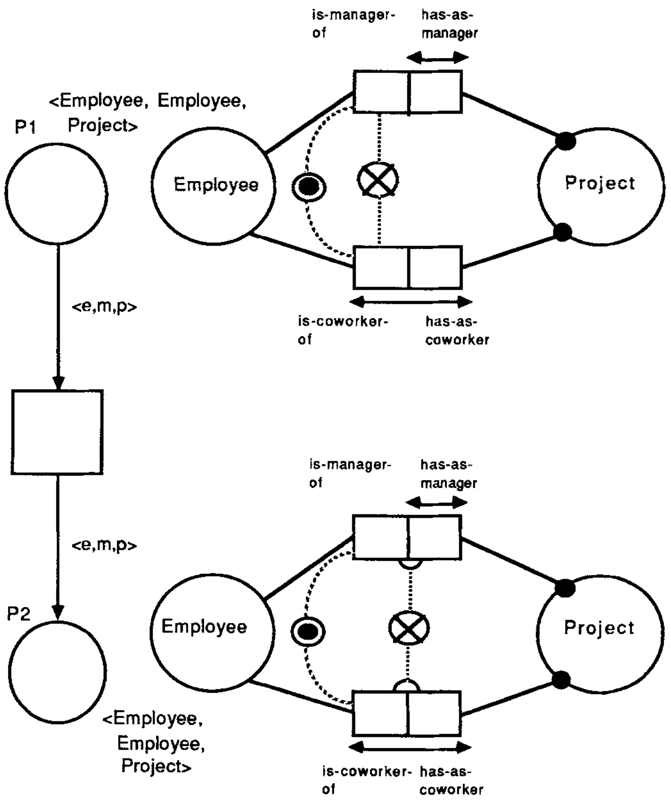


Figure 4.1 Incorrect CTM-net with respect to conceptual schemas

The CTM has several strong points. It was developed to fulfil the requirements on task modelling techniques as formulated in section 1. Fluent transfer between intermediate design results is supported due to the well-defined relation between activities and tasks and the incorporation of a data modelling technique. Tasks can be modelled on a conceptual level, thus enabling the analyst to abstract from particular machines and programming languages and their limitations. The CTM allows for code generation and the verification of all kinds of theoretical statements. Common constructs used in the processing of data elements as well as data manipulation and data retreival can be expressed easily. Finally, in principle every computable function can be specified in the CTM.

The CTM has however also some weak properties, inherited from its PrT basis. Update and retrieval form a database cannot be modelled in an elegant way. Hierarchical decomposition of tasks is not possible, due to the unclear firing semantics of the decomposition, when data elements corresponding to more than one input or output place are optional. CTM-nets tend to be diagrammatically quite complex, which makes the support by a tool an absolute requirement.

The above mentioned weak properties of the CTM suggest directions for future research. Other options for future research are the development of a detailed modelling procedure for the CTM and the implementation of a tool, that includes the various consistency and correctness analyses based on the formulated properties.

6. REFERENCES

[Bergstra 86] Bergstra, J.A. and J.W. Klop, "Process Algebra: specification and verification in bisimulation semantics". In: Mathematics and Computer Science II, CWI Monograph 4, Eds. M. Hazewinkel, J.K. Lenstra and L.G.L.T. Meertens, North-Holland, 1986, pp.61-94.

[Brinkkemper 88] Brinkkemper, S. , N. Brand and J. Moormann, "Deterministic Modelling Procedures for Automated Analysis and Design Tools". In: Proceedings of the CRIS 88 conference on Computerized Assistance during the Information Systems Life Cycle, Eds. T.W. Olle, A.A. Verrijn Stuart and L. Bhabuta, Egham, England, September 1988, North-Holland, Amsterdam, pp. 117 - 160.

[Brinkkemper 89a] Brinkkemper, S. and A.H.M. ter Hofstede, "The Modelling of Tasks at a Conceptual Level in Information Systems Development Methods". In: Workshop Proceedings for the CRIS review workshop, Eds. G.M. Nijssen and S. Twine, IFIP WG 8.1 meeting, Sesimbra, Portugal, June 1989.

[Brinkkemper 89b] Brinkkemper, S., "The Essence and Support of Modelling Transparency", Position paper. In: Advance Working Papers, Third International Conference on Computer Aided

Software Engineering, Ed. J. Jenkins, Imperial College, London, UK, July 1989.

[Brodie 82] Brodie, M.L. and E. Silva, "Active and Passive Component Modelling: ACM/PCM". In: [Olle 82], pp.41-92.

[Falkenberg 89] Falkenberg, E.D., R. van der Pols and Th.P. van der Weide, "Understanding Process Structure Diagrams". In: Workshop Proceedings for the CRIS review workshop, Eds. G.M. Nijssen and S. Twine, IFIP WG 8.1 meeting, Sesimbra, Portugal, June 1989.

[Genrich 79] Genrich, H. and K. Lautenbach: "The Analysis of Distributed Systems by means of Predicate/Transition Nets", Semantics of Concurrent Computation. Evian 1979, Ed. G. Kahn, Lecture Notes in Computer Sciences, vol.70, Springer Verlag 1979, pp.123-146.

[Genrich 87] Genrich, H.: "Predicate/Transition Nets". In Petri Nets: Central models and their properties, Eds. W. Brauer, W. Reisig and G. Rozenberg, L.N.C.S. nr 254, Springer Verlag 1987, pp 207-247.

[van Hee 88] van Hee, K.M., G.J. Houben, L.J. Somers and M. Voorhoeve, "Executable Specifications for Information Systems", Computing Science Notes, nr. 88/05, Department of Computing Science, Eindhoven University of Technology, March 1988.

[Jackson 83] Jackson, M.A., "System Development", Prentice Hall, 1983.

[Kung 86] Kung, C.H. and A. Sölvberg, "Activity Modeling and Behavior Modeling". In: Information System Design Methodologies - Improving the Practice, Eds. Olle, T.W., H.G. Sol and A.A. Verrijn Stuart, Proceedings of the CRIS-86 conference, North Holland Publ. Co., 1986, pp. 145 - 171.

[Lewis 81] Lewis, H.R. and C.H. Papadimitriou, "Elements of the theory of Computation", Prentice Hall, 1981.

[Lundeberg 80] Lundeberg, M., G. Goldkuhl and A. Nilsson, "Information Systems Development - A Systematic Approach". Prentice Hall, Englewood Cliffs, 1980.

[Martin 85] Martin, J. and C. McClure, "Action Diagrams", Prentice Hall, Englewood Cliffs, N.J., 1985.

[Meersman 82] Meersman, R., "The RIDL Conceptual Language", Research Report ICIAS, Brussels, 1982.

[Nijssen 89] Nijssen, G.M. and T.A. Halpin, "Conceptual Schema and Relational Database Design: a Fact-Based Approach", Prentice Hall, 1989.

[Olle 82] Olle, T.W., H.G. Sol and A.A. Verrijn Stuart (Eds.), "Information System Design Methodologies - A Comparative Review". North Holland Publ. Co., 1982.

[Reisig 85] Reisig, W., "Petri Nets", EATCS Monographs on Theoretical Computer Science Springer Verlag, 1985.

[Richter 82] Richter, G. and R. Durchholz, "IML-Inscribed High-Level Petri Nets". In: [Olle 82], pp.335-368.

[Rolland 82] Rolland, C. and C. Richard, "The REMORA Methodology for Information System Design and Management". In: [Olle 82], pp. 369-426.

[Ter Hofstede 89] Ter Hofstede, A.H.M. and S. Brinkkemper, "Conceptual Task Modelling", Technical report nr. 89-14, Department of Information Systems, University of Nijmegen, September 1989.

[Yourdon 79] Yourdon, E. and L. Constantine, "Structured Design", Yourdon Press, Englewood Cliffs, N.J., 1978.