Modelling the Evolution of Information Systems*

J.L.H. Oei,
H.A. Proper,
E.D. Falkenberg,
E.Proper@acm.org

June 23, 2004

PUBLISHED AS:

J.L.H. Oei, H.A. Proper, and E.D. Falkenberg. Modelling the Evolution of Information Systems. Technical Report 92-36, Department of Information Systems, University of Nijmegen, Nijmegen, The Netherlands, EU, 1992.

Abstract

In this article, we discuss the need for information systems capable of evolving to the same extent as organisation systems do. A set of requirements for evolving information systems is presented, implying the importance of the time concept in these systems. On the basis of these requirements an architecture and a conceptual framework for evolving information systems is proposed.

In our conceptual framework for update we distinguish recording, correction and forgetting. State transitions are considered on three different levels of abstraction, viz. the event level, the recording level and the correction level. A formal specification of the transformation process between user update requests, primitive update requests, and the three-level model for update is provided.

1 Introduction

As argued in [FOP92a], [FOP92c] and [Rod91], there is a substantial demand for information systems which are able to evolve to the same extent and at the same pace at which the supported organisation system evolves due to changes in the universe of discourse ([ISO87]). Most traditional information systems hardly support any aspect of evolution.

First of all, most traditional information systems only allow for update of the information base, i.e. the set of facts obeying a fixed (conceptual) schema with a fixed set of constraints. In other words, update of the conceptual schema, and consequently the internal data base schema, constraints, and specifications of dynamic aspects - i.e. activity and behaviour specifications - have not yet been supported by these traditional information systems. The evolving information systems at which we aim, however, do support update of all these specifications. A very important requirement for these evolving information systems is that no update of any sort should result in the interruption of activities of the evolving organisation ([FOP92a], [FOP92c], [FOP92b]). Related work on conceptual schema evolution can be found in e.g. [MS90], [Rod91].

Another limitation of many traditional systems is that they are *snapshot* systems, i.e. they reflect information valid at a certain point of time, but lack the ability to preserve the history of information. An evolving information system, on the contrary, must be *conservative* or *temporal* ([SA85] [SA86], [FOP92a], [FOP92c]) in the sense that it should not forget anything ever fed to the system, unless explicitly asked for, thus allowing for the formulation of queries about the history of the information base (populations).

*The investigations were partly supported by the Foundation for Computer Science in the Netherlands (SION) with financial support from the Netherlands Organization for Scientific Research (NWO). Department of Information Systems, University of Nijmegen, Toernooiveld 1, NL-6525 ED Nijmegen, The Netherlands

In such systems, the distinction between *event time* - the time at which an event occurs in the universe of discourse - and *recording time* - the time at which the event is recorded in the information system - is of great importance ([Bub80], [SA85], [SA86], [Sno90]). Traditional systems can thus be regarded as *degenerations* of these evolving information systems ([FOP92c]). For a more elaborate discussion on the difference between traditional and evolving information systems, see also [FOP92a].

An important concept in evolving information systems are updates. In traditional information systems, updates are a non-trivial aspect, in the field of evolving information systems, this is even more the case. In this article we focus on a suitable meta model for update in generalised evolving information systems. The presented model is equally applicable to evolving as well as historical information system, and may be characterised as state-transition-oriented and taking place on different levels of abstraction. On the event level, transitions of states of the organisation system are described - these states are modelled in what we call the application model states - which are caused by events in the universe of discourse. On the next level, the recording level, transitions of histories of these states are considered due to recording of these events. Finally, on a third level, which we will call the correction level, corrections of incorrect recordings are viewed as transitions of recording histories on the recording level. Such a correction is performed by a roll-back to the latest correctly recorded state. This roll-back should be followed by the recording of all events which happened after the event resulting in the state to which has been rolled back.

The organisation of the paper is as follows. In section 2, the need for evolving information systems, and their requirements are identified. Section 3 discusses a global architecture for an evolving information system. In section 4, distinct types of update requests in evolving information systems are identified, and captured in a formal framework. These types of updates are refined to a smaller set of, more primitive, types of update, in section 5. The processing of update requests by an evolving information system is discussed and formalised in section 6, where the three levels of state transitions as mentioned above, play a crucial role. The article is finallised in section 7 with a short summary, and outlook on future research on the field of evolving information systems.

2 Evolution of Information Systems

Nowadays, in order for an organisation to be competitive on the global market place, it must be flexible. The organisation must be able to adapt itself quickly to the production of new or different products - changes in the primary process of an organisation - and to the ever changing and more and more demanding consumer needs. The present day requirements of organisations imply a new, more demanding, set of requirements on their information systems. Due to the dynamic behaviour of organisations they have to deal with rapidly changing information needs. Given the fact that information is gradually becoming a production factor of more and more importance, it becomes crucial to have information systems which can, easily, be adapted to the same extent as these information needs change. In this context, it is interesting to note that in the current situation already 42% of all maintenance of information systems is needed for extensions/changes of the information system due to changes of the information needs ([Bem87]). In case of a highly dynamic organisation, this latter percentage is likely to be even higher.

Realizing this situation, it can be concluded that information systems are needed which can be adapted to the changing environment in smaller, easier, and more frequent steps than generally is the case. In [Bem87], the adaptability of (information) systems is termed flexibility in a broader sense, and the ability of a (information) system to continue to function in a satisfactory way without having to change the information system, although the organisation has changed, is termed flexibility in the narrower sense. The requirements for an evolving information systems are beyond those of traditional, including temporal, information systems. The main requirement for an evolving information system is that, as stated before, it is able to evolve to the same extent and at the same pace as the underlying organisations does, without the need to interrupt the processes of that organisation. The notion of 'to the same extent' and 'at the same pace' is now refined in more detail:

1. The information system allows update of all information depending on the specific universe of discourse of the information system. The notion of update, including recording, correction and forgetting, is discussed in section 4. The specification of information depending on a specific universe of discourse is part of the architecture for evolving information systems as discussed in section 3.

2. As information recorded in the information system may appear to be (empirically) invalid, the information system allows correction of all information (previously) recorded in the system. The notion of correction is discussed in section 4. Note that the need for correction results from validation and not from verification. Consistency of the recorded information is checked by the information system itself.

3. The system does not forget any information recorded in the information system unless explicitly asked for. In other words, the complete history of information inside the information system is kept, including that of correction, unless a user request or a law demands that information has to be forgotten (e.g. because of privacy reasons).

4. Updates of the information system may not interrupt activities of the organisation. As the intention of evolving information systems is to minimize the discrepancy between the information needs of the organisation and the information supply by the information system, the information system is required to remain on line when any part of the stored information is updated.

A major consequence of these requirements, is that the notion of time has to be introduced in the information system as a distinct concept, in order to meet these requirements. Even more, at least two distinct notions of time have to be distinguished. It will be obvious that for meeting requirement 3 events in the organisation have to be recorded together with their time of occurence. The point of time at which an event occurs in the organisation is called the event time of that event. To perform corrections a roll-back operator is needed (see section 4). This roll-back operator enables us to restore a former state of the information system. To accomplish this, the point of time at which recordings of events take place in the information system are needed. These point of times are called the recording time of events.

Our notion of event time and recording time is identical to the notions of valid time, and transaction time, respectively, in [SA86]. (The reason for this renaming is that the new names correspond better to the three level architecture we will introduce in section 6.) The classification made in [SA86] is based on the support of this valid and/or transaction time. Conform this classification, distinguishing snapshot-, historical-, rollback-, and temporal systems, evolving information systems are temporal systems, as both valid and transaction time are supported. It should be noted, however, that not all temporal systems are evolving information systems. As we have seen in this subsection, evolving information systems have to meet additional requirements.

3 The Architecture of Evolving Information Systems

In our systems view on organisations, conform the approach taken in [FHL$^+$98], an organisation system is considered to be a set of interrelated actors, activities, states and points of time. The information systems considered here, are restricted to information systems where the only actor performing information processing activities is computerized. This computerized actor is called the information processor. The information processor may be composed out of several sub-processors, which may be (physically) distributed. In this paper, however, the specific aspects of distributed information systems are not taken in consideration.

The restriction to a computerized actor performing information system processing activities, corresponds to what has been defined in [Ver89] as an information system in the narrower sense *IS(N)*. In this paper, whenever the term information systems is used, information systems in the narrower sense are meant. Conform this systems view on organisations and information systems, a general architecture for information systems is presented. On the basis of this architecture the distinction between traditional and evolving information systems is explained.

The architecture of an evolving information system is depicted in figure 1. The information processor in an evolving information system accepts input messages (requests) reflecting, among other things, changes of state (events) in the universe of discourse, and triggering the information processor to perform activities. As a result of these activities, the information processor may produce output messages (responses). These output messages are received in turn by the universe of discourse, which is embedded in the environment of the information system. In an information system, the description of that part which is consulted by

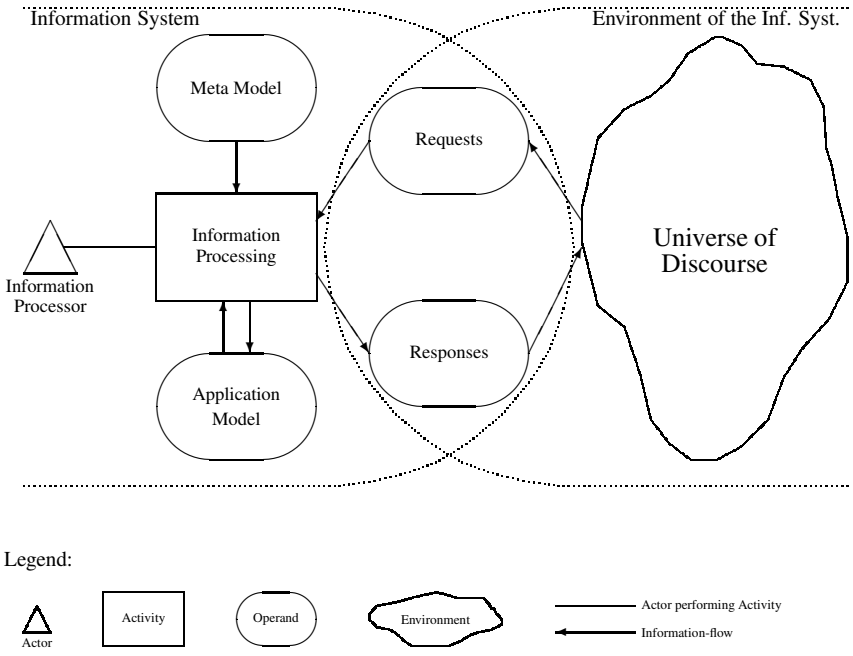

Figure 1: An (Evolving) Information System and its Environment

the information processor to process user requests, is called the processing model. (The description of the user requests themselves is not considered to be part of the processing model). The processing model can be divided into a part describing a particular universe of discourse, the application model, and a part describing the language (technique) in which this application model is specified and can be manipulated. The latter part is called the meta model, and contains the description of a classification of domain elements, general rules about these elements, their behaviour, and how they can be treated ([BF91]).

In this paper the meta model and the corresponding specification language(s) are assumed to be stable. Changes are restricted to the application model only. Conform the terminology introduced in [OHFB92] this means that in this paper we restrict ourselves to information systems supporting first-order evolution. Second-order evolution involves changes of the meta model as well. As a result, in the scope of this paper the meta model is provided in a particular information system once and for all, while the application model must be built up and maintained for each new application. The building-up and maintenance of an application model is done by the information processor, which acts on, or reacts to events in the universe of discourse (after receiving input messages) by consulting both the meta model and application model. Thus, unlike the meta model, the application model is not only input, but also output of the activities of the information processor. Besides update of the application model, information can be retrieved from the application model as well. Messages are correspondingly classified into update and retrieval messages. The language for formulating such messages in an information system are based on the meta model of that particular information system.

An application model can be subdivided further. On the one hand, a model of that part of the perceived world (universe of discourse) the interaction between the information system and the environment is about can be identified. This model is called the world model, and can be described in a modelling technique like ER ([Che76]), NIAM ([Win90]) or, for complex application domains, PSM ([HPW92b]).

On the other hand, rules are needed which determine the actions of the information processor. These rules are specified in what is called the action model. The action model can be subdivided into a part that specifies activities - the activity model - and a part that describes the (trigger-) relations between the activity model and the world model. This latter part is referred to as the behaviour model. In the behaviour model, for example, the relationship between events in the universe of discourse and the activities performed by the information processor in the information system is described. In other words, the behaviour model contains the description of *when* activities, under which conditions, and *what* activities should be performed by

the information processor, whereas the activity model specifies *how* these activities should be performed. Examples of modelling techniques for the action model are Data Flow Diagrams ([GS86]), the A-schemas in ISAC ([LGN81]), or Task Structures ([HN93]). The complete architecture of the processing model is depicted in figure 2.

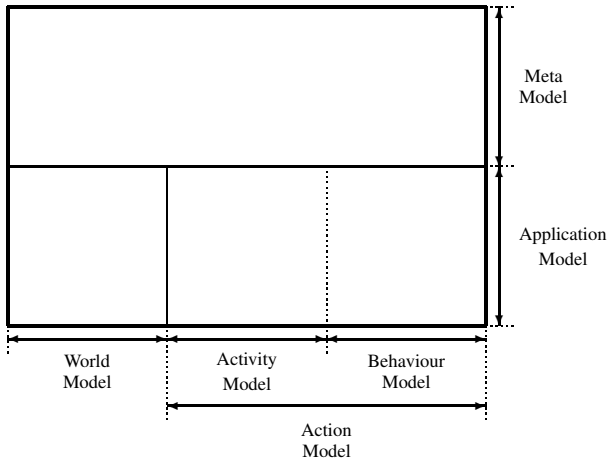

Figure 2: The Structure of the Processing Model

On the basis of this architecture, the distinction between a traditional information system and an evolving information system can be explained more specifically. In a traditional information system, where the traditional schema (type) vs. instance dichotomy (e.g. [BF91]) is applied to the application model, only the instances can be updated. That is, schema specifications, as well as activity and behaviour specifications (which are usually hidden in program procedures), cannot be updated in traditional information systems. The intention of an evolving information system, however, is that the complete application model becomes updatable.

In accordance with the above proposed architecture for evolving information systems, the metamodel contains all the application-independent knowledge needed by the information processor to perform its activities. Among other things, this knowledge includes the knowledge needed by the information processor to transform requests from users of the information system into update and retrieval actions on the application model, as well as output messages (responses), if any. The design of an Evolving Information System-shell has to be based on this transformation process. The input of this transformation process consists of the requests formulated by the users, and the state of the processing model of a particular evolving information system. The updated application model is, together with produced responses, if any, the output of the transformation process.

In this article, we focus on the update requests entered by the user. These requests will be considered on three levels, resulting in three models for user updates:

User update requests model This model reflects the updates as entered by the user, containing several kinds of updates, allowing for the specification of *user update requests* in a user friendly way.

Primitive user update requests model The updates as entered by the user can be transformed to *primitive update requests*, conform a set of more primitive (atomic) kinds of update requests.

User update processing model In this model, the effect of the primitive user update requests on the application model is reflected. The *semantics* of the user update requests is defined in terms of the semantics of the underlying modelling techniques used for modelling the application model.

The transformation process betweeen these three models will be defined subsequently in the following sections.

4 Update in Evolving Information Systems

Information systems are meant to fulfill the information needs of organisations. As both the information needs and the information itself change in time, information systems have to be updated from time to time.

In the remainder of this paper, a framework for update in evolving information systems is introduced, which is based on the possible causes for update requests. As argued in [FOP92c], update in traditional systems can be regarded as a degeneration of update in evolving systems. This is solely possible due to the fact that the requirements of traditional information systems, with respect to update are less demanding than those of evolving information systems.

In this section, three main classes of user update requests are identified: recording, correction and forgetting.

4.1 A Taxonomy of Update in Evolving Information Systems

Updates in (evolving) information systems, result in a change of state of the application model. In traditional information systems an elementary update is usually considered to be an addition, deletion or modification of (pieces of) information contained in the application model. Furthermore, traditional information systems do not support any notion of time. As a consequence, if there is a change of state in the information system due to an update (addition, deletion or modification), the former state cannot be *remembered* by the information system. Such systems are called snapshot systems ([SA86]), due to the fact that these information systems can only model and remember (single) *snapshots* of an organisation's evolution.

For evolving information systems, as well as for temporal or historical information systems ([SA86]), the traditional notion of update has to be revised. The framework for the processing of updates in evolving information systems, as it will be presented in this paper, is based on the possible causes for update requests. On this basis, three main classes of update are distinguished: recording, correction and forgetting ([FOP92c], [FOP92b]).

Since the application model should be a reflection of the universe of discourse, every *event* (change of state at a point of time) in the universe of discourse (for instance the hiring/firing of personell) should be reported to the information system by means of a *recording* of this event.

In an ideal (evolving) information system, the application model reflects the state of the modelled universe of discourse in a (empirically) correct way at any point of time. It can, indeed, be checked whether the application model is consistent with the meta model, but there is no automatable way to guarantee that the contents of the application model reflect the real state of the universe of discourse. The latter case has to be checked empirically. This implies that in case of a found flaw in the application model, an update should be performed which *corrects* this flaw. This class of update is called a *correction*, and corrects recordings which have already been performed, or which have not been performed at all.

A third class of update can be distinguished if information systems are required to be able to remove information if requested. For instance, a law may exist stating that information about former personell working for a company, may be stored for at most two years. This kind of update is called *forgetting*. Due to its complexity - the consistency of the stored information must be maintained even when parts of it are removed - this class of update is not considered in depth in this paper.

4.2 Recording of Events

The state of the universe of discourse, at a point of time, is reflected by a set of modelling constructs (e.g. entities, relationships, data base schemata, instances, rules, etc.) in the application model. These modelling constructs are referred to as *application modelling elements* ([FOP92a], [FOP92b]). A recording of an event can thus be regarded as the modification of the set of application model elements constituting the state of the application model. A (to be) recorded event is denoted as a sentence conform the language as determined by the chosen meta model.

As an example domain for an event specification, consider a rental store for audio records (lp's). This example will be used as a running example in the sequel. In this rental store, a record is kept of, among other things, the songs that are recorded on the lp's present in the library. In order to keep track of the wear

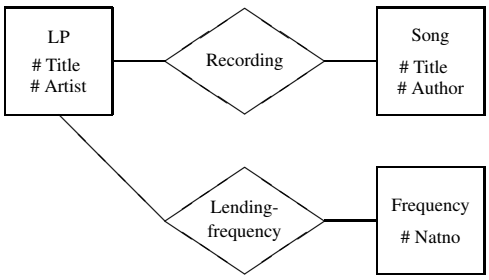

Figure 3: The Data Model of a lp library

and tear of the lp's, the number of times the lp has been lent is recorded as well. This part of the universe of discourse of the rental store has been modelled in figure 3 in the style of ER. Note that we abbreviated the graphical notation of attributes (Title) to a mark symbol (#) followed by the attribute (# Title), for reasons of readability. An example event specification conform the language LISA-D ([HPW93],[HPW94]) would be:

ADD Song: 'Walk of Life' on Record: 'Brothers in Arms'

stating that the song 'Walk of Life' is recorded on record 'Brothers in Arms'

In the next section, the modification of an application model state is performed by means of a set of elementary transitions. These transitions are the reflection of the events in the universe of discourse in the information system. Three kinds of elementary transitions are determined: a birth-transition, creating an application model element, a death-transition, terminating an element, and a change transition which transforms an existing application model element ([FOP92c]).

As has been argued in e.g. [FOP92c] and [SA85], the time of recording of an event is different from its occurence time. The time at which an event occurs in a universe of discourse is the event time, and the time at which the event is recorded is the recording time. This, and the above discussion leads to the following definitions:

Definition 4.1

1. *$\mathcal{ES}$ is the set of event specifications.*
2. *$\mathcal{T}_r$ is the time axis for event recordings, with a total order $<$.*
3. *$\mathcal{T}_e$ is time axis for event occurences, also with a total order $<$. Both the $\mathcal{T}_r$ and $\mathcal{T}_e$ time axis can be defined in a multitude of ways, see e.g. [CR87], [WJL91] or [All84].*
4. *$\mathcal{EO} = \mathcal{ES} \times \mathcal{T}_e$ is the set of all event occurences. An event occurence is identified by an event specification and the point of time at which the event occurs.*
5. *$\mathcal{ER} = \mathcal{EO} \times \mathcal{T}_r$ is the set of all event recordings. A recording of an event occurence is identified by the recorded event occurence, and the point of time at which the recording takes place*

On these concepts, the following functions (access routines) are defined:

Definition 4.2

1. *The point of time at which a recording has taken place:*
   $\mathsf{RcAt} : \mathcal{ER} \rightarrow \mathcal{T}_r$
   $\mathsf{RcAt}(x, t) = t$
2. *The event occurence which is recorded by a recording:*
   $\mathsf{RcOf} : \mathcal{ER} \rightarrow \mathcal{EO}$
   $\mathsf{RcOf}(x, t) = x$

3. *The point of time at which an event occurence, or a recorded event occurence, has taken place:*

   $\mathsf{OccAt} : \mathcal{ER} \cup \mathcal{EO} \rightarrow \mathcal{T}_e$

   $\mathsf{OccAt}(x,t) =$ **if** $x \in \mathcal{EO}$ **then** $\mathsf{OccAt}(x)$ **else** t **fi**

4. *The event specification of an occured, or recorded, event occurence:*

   $\mathsf{OccOf} : \mathcal{ER} \cup \mathcal{EO} \rightarrow \mathcal{ES}$

   $\mathsf{OccOf}(x,t) =$ **if** $x \in \mathcal{EO}$ **then** $\mathsf{OccOf}(x)$ **else** x **fi**

The order on points of time, from both time axes, can be generalised to an order on recordings and event occurences as:

Definition 4.3
Let $x, y \in \mathcal{ER}$ then:

$$\begin{aligned} x <_{\mathcal{T}_r} y &\equiv \mathsf{RcAt}(x) < \mathsf{RcAt}(y) \\ x =_{\mathcal{T}_r} y &\equiv \mathsf{RcAt}(x) = \mathsf{RcAt}(y) \\ x \leq_{\mathcal{T}_r} y &\equiv x <_{\mathcal{T}_r} y \vee x =_{\mathcal{T}_r} y \end{aligned}$$

Let $x, y \in \mathcal{ER} \cup \mathcal{EO}$ then:

$$\begin{aligned} x <_{\mathcal{T}_e} y &\equiv \mathsf{OccAt}(x) < \mathsf{OccAt}(y) \\ x =_{\mathcal{T}_e} y &\equiv \mathsf{OccAt}(x) = \mathsf{OccAt}(y) \\ x \leq_{\mathcal{T}_e} y &\equiv x <_{\mathcal{T}_e} y \vee x =_{\mathcal{T}_e} y \end{aligned}$$

In a traditional snapshot information system, a birth transition of an application model element is realised by means of an addition, and a death-transition by means of a deletion. In both cases, the *old* application model states are lost. In an evolving information system, on the other hand, one of the major requirements is that no information may be lost, implying that no application model element may be deleted. An exception to this rule is formed by the forget operator. The history of application model elements, in an evolving information system, is kept by storing the birth, death and change transitions of application model elements together with the points of time at which these transitions take place.

4.3 Correction of Recordings

The actual state of an information system depends completely on the processing of update requests formulated by users of the system. These update requests should result in an information system reflecting the modelled universe of discourse in a correct way.

An information system reflects an organisation correctly if and only if there exists a isomorphism between the states and transitions of the application model and the states and transitions in the universe of discourse ([HW93], [HPW92a]). From this requirement follows that the order in which the events occured in the universe of discourse (should) be the same as the order of the recorded events. This is needed because recordings of several events may interfere with each other, such that a different order may result in a different state of the information system. As an illustration of this phenomenon, consider the following two events for the rental store:

e_1: ADD Song: 'Walk of Life' on Record: 'Brothers in Arms'

e_2: DELETE Song on Record: 'Brothers in Arms'

The event specifications in the above example are, again, denoted in the semi-natural language LISA-D as defined in [HPW93]. The first event specification (e_1) represents the adding of the fact that song 'Walk of Life' is recorded on record 'Brothers in Arms', whereas the second event specification (e_2) represents the deletion of all facts about songs recorded on record 'Brothers in Arms'. When recording e_1, e_2 in the obvious order, a state of the system will result in which record 'Brothers in Arms' has no song recorded on it. When e_2 is recorded before e_1, i.e. in the wrong order, the result will be that 'Walk of Life' is the only song recorded on the record 'Brothers in Arms'.

From the above example can be concluded that events should be recorded correctly, in order of their occurence. In practice, however, recordings may be performed too late, implying a violation of the proper order in which the events occured. Or, alternatively, a recording of an event which actually did not happen at all, or occured differently (i.e. the event specification was wrong), may have been performed. This means that three kinds of corrections can be identified: the insertion of a (late) recording of an event in the sequence of already performed recordings, a removal of an already performed recording, or a replacement of a recording by a new recording of another event.

To accomplish these kinds of corrections, it must be possible to *travel backwards* in the sequence of recordings which is ordered on time of recording. The operation accomplishing this task is called a roll-back, and is the most primitive form of correction. In section 5, a formal definition is given of how the above discussed three kinds of corrections can be mapped onto roll-backs and (re)-recordings. The introduction of a correction mechanism leads to the following definitions:

Definition 4.4

1. $\mathcal{RM} = \mathcal{T}_r \times \mathcal{T}_r$ *the set of all removals. A removal of a recording is concerned with the removal of a recording, which is identified by a recording time, and occurs at a point of time. Therefore, a removal is identified by two points of time, the recording time of the recording which is to be removed, and the recording time of the removal.*

2. $\mathcal{RP} = \mathcal{T}_r \times \mathcal{ES} \times \mathcal{T}_r$ *the set of all replacements. All replacements of recordings, replace a recording of an event specification which happened at a certain point of time by a new event specification. Thus, a replacement can be identified by means of the point of time of the recording to be replaced, the new event specification, and the recording time of the replacement.*

3. $\mathcal{RB} = \mathcal{T}_r \times \mathcal{T}_r$ *the set of all roll-backs. Every roll-back rolls back to a recording which is identified by a recording time. Therefore, a roll-back is identified by the recording time of the recording to which the roll-back takes place, and the recording time at which the roll-back takes place.*

4. *For reasons of convenience the set of all possible corrections ($\mathcal{CR}$), as well as the set of all possible update requests ($\mathcal{UR}$) are defined as well:*

$$\begin{aligned} \mathcal{CR} &= \mathcal{RB} \cup \mathcal{RM} \cup \mathcal{RP} \\ \mathcal{UR} &= \mathcal{ER} \cup \mathcal{CR} \end{aligned}$$

Now that roll-backs have been defined formally, it is interesting to note that a recording can be regarded as an operation on event occurences, whereas a rollback can be regarded as an operation on recordings. On the above defined concepts, the following operations exist:

Definition 4.5

1. *The* RcAt *operation on recordings of events is generalised to all updates. The point of time at which an update took place is given by:*

   $\mathsf{RcAt} : \mathcal{UR} \rightarrow \mathcal{T}_r$

   $\mathsf{RcAt}(x, t) = t$

   $\mathsf{RcAt}(x, y, t) = t$ *for replacements*

2. *The point of time (of the recording) to which (and including) the given roll-back takes place is determined by:*

   $\mathsf{RbTo} : \mathcal{CR} \rightarrow \mathcal{T}_r$

   $\mathsf{RbTo}(t, x) = t$

   $\mathsf{RbTo}(x, y, t) = t$ *for replacements*

3. *For replacements, the new event specification which is to replace the old one at* $\mathsf{RbTo}(t_1, o, t_2)$*, is given by:*

   $\mathsf{RpBy} : \mathcal{RP} \rightarrow \mathcal{ES}$

   $\mathsf{RpBy}(t_1, e, t_2) = e$

Finally, based on the generalised RcAt function the order on points of time on recordings ($<_{\mathcal{T}_r}$) can be generalised to an order on updates ($\mathcal{UR}$) as:

Definition 4.6
Let $x, y \in \mathcal{UR}$ then:

$$\begin{aligned} x <_{\mathcal{T}_r} y &\equiv \mathsf{RcAt}(x) < \mathsf{RcAt}(y) \\ x =_{\mathcal{T}_r} y &\equiv \mathsf{RcAt}(x) = \mathsf{RcAt}(y) \\ x \leq_{\mathcal{T}_r} y &\equiv x <_{\mathcal{T}_r} y \vee x =_{\mathcal{T}_r} y \end{aligned}$$

4.4 User Update Requests

By means of the concepts defined in the previous subsections, the user update requests can be defined formally. As stated before, the update requests for an evolving information system are assumed to consist of four kinds of update requests: recordings, rollbacks, removals of recordings and replacements of recordings. This leads to the following definition:

Definition 4.7 *The set of update requests entered by the user are modelled as:*

$$\mathcal{UI} = \langle \mathcal{UI}_{rc}, \mathcal{UI}_{rb}, \mathcal{UI}_{rm}, \mathcal{UI}_{rp} \rangle \text{ such that } \mathcal{UI} \in \wp(\mathcal{ER}) \times \wp(\mathcal{RB}) \times \wp(\mathcal{RM}) \times \wp(\mathcal{RP})$$

where $\mathcal{UI}_{rc}$ (recordings), $\mathcal{UI}_{rb}$ (roll-backs), $\mathcal{UI}_{rm}$ (removals) and $\mathcal{UI}_{rp}$ (replacements) are presumed to be disjoint. As a shorthand, the set of corrections ($\mathcal{UI}_{cr}$) and the set of updates ($\mathcal{UI}_{ud}$) are also defined:

$$\begin{aligned} \mathcal{UI}_{cr} &= \mathcal{UI}_{rb} \cup \mathcal{UI}_{rm} \cup \mathcal{UI}_{rp} \\ \mathcal{UI}_{ud} &= \mathcal{UI}_{rc} \cup \mathcal{UI}_{cr} \end{aligned}$$

As a simplifying assumption it is presumed that only one update occurs at one point of time. Note that this does not mean that an evolving information system has to be a single user system, it is simply demanded that there exists a global order on the user's updates. The assumption is formulated as:

Axiom 4.1 *Different updates are not recorded at the same point of time:*

$$u_1, u_2 \in \mathcal{UI}_{ud} \wedge u_1 =_{\mathcal{T}_r} u_2 \Rightarrow u_1 = u_2$$

The set of event occurences recorded by the recordings in $\mathcal{UI}_{rc}$ is denoted as $\mathcal{UI}_{occ}$, and the set of event specifications associated with the recordings in $\mathcal{UI}_{rc}$ as $\mathcal{UI}_{spc}$. They are identified by:

Definition 4.8

$$\begin{aligned} \mathcal{UI}_{occ} &= \{\mathsf{RcOf}(r) \,|\, r \in \mathcal{UI}_{rc}\} \\ \mathcal{UI}_{spc} &= \{\mathsf{OccOf}(r) \,|\, r \in \mathcal{UI}_{rc}\} \end{aligned}$$

As an overview of the hitherto defined concepts, a (meta) conceptual schema of the update requests, relating all concepts used for user update requests, is given in figure 4. The model depicted there is in the style of the modelling technique from NIAM ([NH89], [Win90]).

5 Derivation of Primitive Update Requests

Update requests for the manipulation of an application model at hand, are formulated by the user in a particular language. This language is based on a user update request model, describing the set of possible update requests. Conform the architecture discussed in the section 3, user update requests are mapped onto primitive update requests.

The update requests as entered by the user may consist of recordings of events or corrections of recordings, where corrections may be roll-backs to old recordings, removals of old recordings, or replacements of old recordings by new (correct) ones. The semantics of these four kinds of user update requests can thus

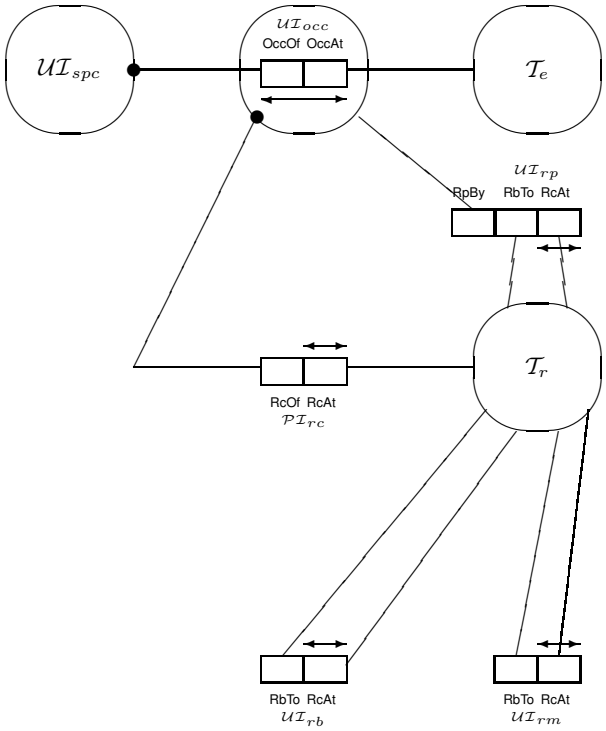

Figure 4: A Meta Model for Update Requests

be expressed in terms of recordings and roll-backs, representing the *primitive user update requests*. This section solely deals with this mapping.

From axiom 4.1, it follows that $<_{\mathcal{T}_r}$ defines a strict total order on the set of user update requests in $\mathcal{UI}_{ud}$. This allows for the following definition:

Definition 5.1 *Let $U_{\mathcal{UI}}$ be a list of updates such that $U_{\mathcal{UI}}(i)$ is the i-th update in $\mathcal{UI}_{ud}$ based on the order implied by $<_{\mathcal{T}_r}$. This definition implies that $dom(U_{\mathcal{UI}}) = \{1, \ldots, |\mathcal{UI}_{ud}|\}$.*

Before defining the actual translation, an extra function returning a first recording in a set of recordings has to be defined:

Definition 5.2 *The first recording in a set of recordings:*

$\mathsf{First} : \wp(\mathcal{ER}) \rightarrow \mathcal{ER}$

$\mathsf{First}(R) = r$ *such that:* $r \in R \wedge \mathsf{RcAt}(r) = min(\{\mathsf{RcAt}(s) \,|s \in R\})$

The set of rollbacks and recordings, at a primitive level, can now defined by translating removals and replacements of recordings, and inserted recordings, to rollbacks and rerecordings. This translation is formulated as:

Definition 5.3 *$R_{\mathcal{UI}}$ is defined as a list of sets of recordings such that $dom(R_{\mathcal{UI}}) = \{0, \ldots, |\mathcal{UI}_{ud}|\}$ and:*

$$
\begin{array}{l}
R_{\mathcal{UI}}(i) = \textbf{if } i = 0 \textbf{ then } \varnothing \\
\qquad \textbf{else} \\
\qquad\quad \textbf{if } U_{\mathcal{UI}}(i) \in \mathcal{UI}_{rc} \textbf{ then } \mathit{Insert}(U_{\mathcal{UI}}(i), R_{\mathcal{UI}}(i-1)) \\
\qquad\quad \textbf{elsif } U_{\mathcal{UI}}(i) \in \mathcal{UI}_{rb} \textbf{ then } \mathit{RollBack}(U_{\mathcal{UI}}(i), R_{\mathcal{UI}}(i-1)) \\
\qquad\quad \textbf{elsif } U_{\mathcal{UI}}(i) \in \mathcal{UI}_{rm} \textbf{ then } \mathit{Remove}(U_{\mathcal{UI}}(i), R_{\mathcal{UI}}(i-1)) \\
\qquad\quad \textbf{else } \mathit{Replace}(U_{\mathcal{UI}}(i), R_{\mathcal{UI}}(i-1)) \\
\qquad\quad \textbf{fi} \\
\qquad \textbf{fi}
\end{array}
$$

Note that by $dom(E)$ the domain of the sequence is meant, i.e. all the indices such that the sequence has a value at that position. In the above definition, the following extra functions (refinements) have been used:

1. The function RollBack applied to an update request u on a set of recordings R results in the set of recordings in R which are still correct after performing a proper rollback. $\mathit{RollBack}(u, R) \equiv$ **if** $u \in \mathcal{UI}_{rc}$ **then** $\{r \in R \,|\, r <_{\mathcal{T}_e} u\}$ **else** $\{r \in R \,|\, r <_{\mathcal{T}_r} \mathsf{RbTo}(u)\}$ **fi**

2. The set of recordings which are undone by the rollback is the result of the function Undone. $\mathit{Undone}(u, R) \equiv R - \mathit{RollBack}(u, R)$

3. Rerecording of all recordings in a set of recordings R from t onwards, implies the selection of the recordings in R one by one, ordered by time, and rerecording them at $\Box t$.

$$
\mathit{ReRec}(t, R) \equiv \begin{cases} \textbf{if } R = \varnothing \textbf{ then } \varnothing \\ \textbf{else } \{\ \langle \mathsf{RcOf}(\mathsf{First}(R)), \Box t \rangle\ \} \ \cup\ \mathit{ReRec}(\Box t, R - \{\mathsf{First}(R)\}) \\ \textbf{fi} \end{cases}
$$

   Note that the recordings in R are selected one by one, as due to axiom 4.1 there is just one candidate for $\mathsf{First}(R)$.

4. In order to remove the recording which happened at $\mathsf{RbTo}(b)$ from a set of recordings, first a rollback to $\mathsf{RbTo}(b)$ has to be performed on R. After this, the undone recordings (except for the one which happened at $\mathsf{RbTo}(b)$) have to be rerecorded.

   $\mathit{Remove}(u, R) \equiv \mathit{RollBack}(u, R) \cup \mathit{ReRec}(\mathsf{RcAt}(u),\ \mathit{UnDone}(u, R) - \{u\})$

5. Inserting a recording s in a set of recordings R means that s has to be added, and that all recordings which happened too early (may be none!) have to be removed from R, and rerecorded from $\mathsf{RcAt}(s)$ onwards.

   $\mathit{Insert}(u, R) \equiv \mathit{RollBack}(u, R) \cup \mathit{ReRec}(\mathsf{RcAt}(u),\ \mathit{UnDone}(u, R))$

6. A replacement is performed by an appropriate rollback, followed by the recording of the new event specification, followed by the rerecording of all undone recordings, except the one to be replaced.

   $$
   \begin{array}{l}
   \mathit{Replace}(u, R) \equiv \mathit{RollBack}(u, R) \cup\ \{\langle \langle \mathsf{RpBy}(u), \mathsf{OccAt}(r) \rangle, \mathsf{RcAt}(u) \rangle\} \\
   \qquad\qquad\qquad\qquad \cup\ \mathit{ReRec}(\mathsf{RcAt}(u),\ \mathit{Undone}(u, R) - \{r\}) \\
   \qquad \text{where } \mathsf{RcAt}(r) = \mathsf{RbTo}(u)
   \end{array}
   $$

By means of the definition of $U_{\mathcal{UI}}$ and $R_{\mathcal{UI}}$ some extra axioms on the user update requests, which could not be formulated before, can be formulated. As a well-formedness rule, every rollback implied by a correction has to rollback to an existing (on the primitive level) recording. Formally:

Axiom 5.1 $U_{\mathcal{UI}}(i) \in \mathcal{UI}_{cr} \Rightarrow \exists_{r \in R_{\mathcal{UI}}(i-1)} [\mathsf{RcAt}(r) = \mathsf{RbTo}(U_{\mathcal{UI}}(i))]$

Furthermore, the recording times of the user's update requests have to be such that there exists enough temporal space (a large enough time-interval) between the updates to allow for rerecordings. This is formulated by:

Axiom 5.2 $0 < i \leq |U_{\mathcal{UI}}| \wedge r \in R_{\mathcal{UI}}(i) \Rightarrow r <_{\mathcal{T}_r} U_{\mathcal{UI}}(i+1)$

The set of primitive user update requests ($\mathcal{PI}$) for the user update requests ($\mathcal{UI}$) can now be defined as two sets containing recordings and roll-backs:

Definition 5.4 *The primitive user update requests for the set of user upate requests $\mathcal{UI}$ is defined as: $\mathcal{PI} = \langle \mathcal{PI}_{rc}, \mathcal{PI}_{rb} \rangle$ where:*

$$\mathcal{PI}_{rc} = \bigcup_{i=1}^{|\mathcal{UI}_{ud}|} (R_{\mathcal{UI}}(i))$$

The set of recordings, at a primitive level, is the union of all recording sets in $R_{\mathcal{UI}}$.

$$\mathcal{PI}_{rb} = \{\ \langle \mathsf{RcAt}(r), \mathsf{RcAt}(U_{\mathcal{UI}}(i)) \rangle\ \mid R_{\mathcal{UI}}(i-1) \not\subset R_{\mathcal{UI}}(i) \wedge r = \mathsf{First}(R_{\mathcal{UI}}(i-1) - R_{\mathcal{UI}}(i))\ \}$$

A roll-back has taken place iff $R_{\mathcal{UI}}(i-1)$ is not a subset of the recordings in $R_{\mathcal{UI}}(i)$. The set of unrolled recordings due to a roll-back is $R_{\mathcal{UI}}(i-1) - R_{\mathcal{UI}}(i)$. Therefore the roll-back must have been to $\mathsf{First}(R_{\mathcal{UI}}(i-1) - R_{\mathcal{UI}}(i))$. *Note that $\mathcal{PI}_{rc} \subseteq \mathcal{ER}$ and $\mathcal{PI}_{rb} \subseteq \mathcal{RB}$.*

On the primitive user update requests two usefull properties hold. In [FOP92b], where only the primitive user update request were taken into account, these properties were axioms on the primitive user update requests. In this article, these properties are to be regarded as requirements on the translation algorithm in definition 5.3.

The first property reflects the uniqueness of primitive update requests with respect to their recording time, i.e. the uniqueness of recording time is inherited from the external level. This property is formulated as:

Lemma 5.1 $u_1, u_2 \in \mathcal{PI}_{rc} \cup \mathcal{PI}_{rb} \wedge u_1 =_{\mathcal{T}_r} u_2 \Rightarrow u_1 = u_2$

Proof: We will not give an exact proof here, instead the correctness of the lemma (and thus of definition 5.3) is made plausible, as this property is a requirement on definition 5.3. Three cases can be distinguished:

1. If $u_1, u_2 \in \mathcal{PI}_{rc}$, then from definitions 5.3, 5.4 and axioms 5.2, 4.1 follows that $u_1 = u_2$.
2. If $u_1, u_2 \in \mathcal{PI}_{rb}$ then from definition 5.1 and axiom 4.1 follows that

   $$i \neq j \Rightarrow \mathsf{RcAt}(U_{\mathcal{UI}}(i)) \neq \mathsf{RcAt}(U_{\mathcal{UI}}(j))$$

   Then because of $\mathcal{PI}_{rb}$'s definition, it follows that that $u_1 = u_2$.
3. $u_1 \in \mathcal{PI}_{rc}$ and $u_2 \in \mathcal{PI}_{rb}$ or vice versa.
   From definitions 5.1, 5.3 follows that $u_1 \neq u_2$.

□

The second property, being a direct consequence from the previous lemma (requirement), reflects the existance of a global order on the set of primitive user updates, i.e. definition 5.3 maintains the order implied by axiom 4.1 on the user updates:

Lemma 5.2 *The $<_{\mathcal{T}_r}$ relation defines a strict total order on $\mathcal{PI}_{rc} \cup \mathcal{PI}_{rb}$.*

Finally, the set of occurences associated with the recordings in $\mathcal{PI}_{rc}$ is denoted as $\mathcal{PI}_{occ}$, and the set of event specifications associated with the recordings in $\mathcal{PI}_{rc}$ as $\mathcal{PI}_{spc}$. They are identified by:

Definition 5.5

$$\begin{aligned} \mathcal{PI}_{occ} &= \{\mathsf{RcOf}(r) \mid r \in \mathcal{PI}_{rc}\} \\ \mathcal{PI}_{spc} &= \{\mathsf{OccOf}(r) \mid r \in \mathcal{PI}_{rc}\} \end{aligned}$$

An overview of the concepts defined thus far, is provided by the (meta) conceptual schema of the primitive user input depicted in figure 5. The model depicted there is, again, in the style of the modelling technique from NIAM.

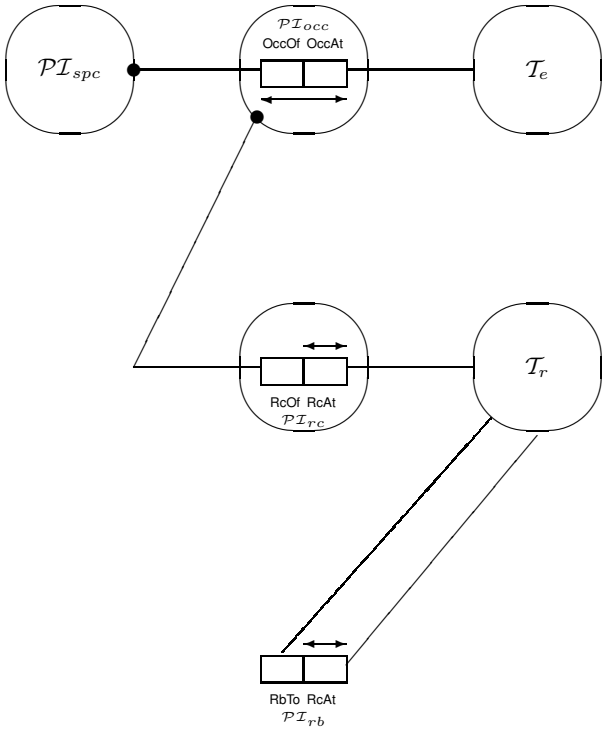

Figure 5: A Meta Model for the Primitive User's Input

6 Processing Primitive Update Requests

Based on the notion of primitive update request as defined in the previous section, a framework for the processing of these updates is presented ([FOP92b]). This framework distinguishes and relates different types of state transitions. Each type of state transition corresponds to a different level of abstraction in the context of update in evolving information systems. These levels are the event level, the recording level, and the correction level. State transitions on the event level take place due to events occuring in the organisation, state transitions on the recording level are caused by recordings of these events, whereas corrections of previous recordings cause state transitions on the correction level.

6.1 The Event Level

It is generally assumed that in the universe of discourse, described in the information system, a set of stable states can be recognised, and that there exist a number of actions that result in event occurences see e.g. [HW92], [HW93]. As stated before, the elements of the application model reflect the application-dependent or time-variant elements in the universe of discourse, implying that the state of a universe of discourse at a particular point of time can be modelled by means of a set of application model elements. This set of application model elements is called the application model state.

An event, and the underlying state transition in the universe of discourse, is communicated to the system by means of an event specification (and the occurence time of the event). This event occurence implies a state transition of the application model state. In the section 3, it was discussed that there exist three kinds of elementary transitions: birth transitions, death transitions, and change transitions. These concepts are captured formally by means of the following definition:

Definition 6.1

$$\begin{aligned} \mathcal{AME} &= \textit{“Set of Application Model elements”} \\ \mathcal{B} &= \textit{“Set of Birth transitions”} \\ \mathcal{D} &= \textit{“Set of Death transitions”} \\ \mathcal{C} &= \textit{“Set of Change transitions”} \\ \mathcal{TR} &= \mathcal{B} \cup \mathcal{D} \cup \mathcal{C} \;\; \textit{The set of all transitions} \end{aligned}$$

The transitions operate on application model elements. Which elements they operate upon is presumed to be determined by the following functions:

Definition 6.2

$$\begin{aligned} \mathsf{Brth}: &\quad \mathcal{B} \rightarrow \mathcal{AME} \\ \mathsf{Dth}: &\quad \mathcal{D} \rightarrow \mathcal{AME} \\ \mathsf{ChOf}: &\quad \mathcal{C} \rightarrow \mathcal{AME} \\ \mathsf{ChTo}: &\quad \mathcal{C} \rightarrow \mathcal{AME} \end{aligned}$$

The actual definitions of these functions depend on the chosen meta model, in particular on definition of the semantics of the event specifications. These event specifications, as they were introduced in the previous section, are application model elements themselves. This is expressed as:

Axiom 6.1 $\mathcal{ES} \subseteq \mathcal{AME}$

The set of elementary transitions, implied by an event occurence can be determined (can be given a Concr*ete* value) by means of the Concr function. The actual set of elementary transitions as implied by a given event occurence, depends on the current state and past states of the application model. These states can, as will be shown below, be derived from the sequence of sets of already performed elementary transitions. The actual definition of the Concr function will not be given here, since its definition depends on the chosen meta model (and language) for the application model. For instance, in the example discussed in the previous section, the set of elementary transitions implied by the statement DELETE Song on Record: 'Brothers in Arms' depends on the semantics of LISA-D ([HPW93]). The signature of the Concr function, nonetheless, can indeed be given:

Definition 6.3 $\mathsf{Concr} : \mathcal{EO} \times (\mathbb{N} \rightarrow \wp(\mathcal{TR})) \rightarrow \wp(\mathcal{TR})$

$\mathsf{Concr}(e, T)$ *must be interpreted as the set of elementary transitions needed to perform the event* e*, if the transitions in* T *have already occured.*

The actual definition of the Concr is beyond the scope of this paper.

An event taking place in the universe of discourse, usually a part of an organisation, is considered to occur on the organisational level ([FOP92a]). The corresponding events in the information system, implying transitions on the application model states, are considered to occur on the so called event level. A sequence of such application model state transitions is called an application model history. Such an application model history, models a sequence of events occurring in the underlying universe of discourse of the information system.

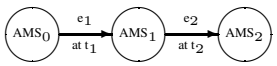

Figure 6: Application Model State (AMS) transitions on the event level

In figure 6 a graphical representation of a sample application model history is given. The circles represent the application model states, whereas transitions between these application model states are represented by arrows. Furthermore, the arrows are labeled with the denotation of the event causing the transition, and the event time of that event. This example illustrates that, to every sequence of event occurences, an application model history can be associated. This means that a function from a sequence of event occurences to an application model history AMH can be defined. In the definition of AMH, we make use of a slicing operation on lists. The slicing of a list is defined as:

Definition 6.4 *If E is a list, then E^n is the list containing the first n elements of E. So E is the list such that $dom(E^n) = \{i \in dom(E) \,|i \leq n\}$ and $i \in dom(E^n) \Rightarrow E^n(i) = E(i)$. Note that $E^{-1} = E^0$.*

For a given sequence of event recordings, the associated application model history is identified by:

Definition 6.5 *Let E be a list of event occurences. Then $\mathsf{AMH}(E) = \langle E, \sigma^{ams}, H^{ams}, \beta, \delta, \gamma \rangle$, where T is a list of sets of transitions such that $T(i) = \mathsf{Concr}(E(i), T^{i-1})$ and $dom(E) = dom(T)$, and furthermore:*

$$\begin{aligned}
\sigma^{ams} &= \mathsf{AMS}(T, 0) \\
H^{ams} &= \{\ \langle \mathsf{AMS}(T,i), \mathsf{AMS}(T,i+1), E(i+1) \rangle\ \mid 0 \leq i < |E|\ \} \\
\beta &= \{\ \langle E(i), \mathsf{Brth}(T(i)) \rangle\ \mid 1 \leq i \leq |E|\ \} \\
\delta &= \{\ \langle E(i), \mathsf{Dth}(T(i)) \rangle\ \mid 1 \leq i \leq |E|\ \} \\
\gamma &= \{\ \langle E(i), \mathsf{ChOf}(T(i)), \mathsf{ChTo}(T(i)) \rangle\ \mid 1 \leq i \leq |E|\ \}
\end{aligned}$$

The Brth*,* Dth*,* ChOf*, and* ChTo *functions are presumed to be generalised to sets of transitions as:*

$$\begin{aligned}
\mathsf{Brth}(T) &= \{\mathsf{Brth}(t) \,|t \in T \cap \mathcal{B}\} \\
\mathsf{Dth}(T) &= \{\mathsf{Dth}(t) \,|t \in T \cap \mathcal{D}\} \\
\mathsf{ChOf}(T) &= \{\mathsf{ChOf}(t) \,|t \in T \cap \mathcal{C}\} \\
\mathsf{ChTo}(T) &= \{\mathsf{ChTo}(t) \,|t \in T \cap \mathcal{C}\}
\end{aligned}$$

In the above definition, there is one *lose end* in the form of the $\mathsf{AMS}(T, i)$ function. This, recursive, function returns the i-th state of the application model based on a list of sets of elementary transitions. This function is defined as:

Definition 6.6 *Let T be a list of sets of transitions. Then $\mathsf{AMS}(T, i)$ is defined as:*

$$\begin{aligned}
\mathsf{AMS}(T,i) = \ & \textbf{if}\ \ i = 0\ \textbf{then}\ \varnothing \\
& \textbf{else}\ \mathsf{AMS}(T, i-1) - \mathsf{Dth}(T(i)) - \mathsf{ChOf}(T(i)) \cup \mathsf{Brth}(T(i)) \cup \mathsf{ChTo}(T(i)) \\
& \textbf{fi}
\end{aligned}$$

where, again, the Brth*,* Dth*,* ChOf*, and* ChTo *functions are presumed to be generalised to sets of transitions.*

6.2 The Recording Level

In this section, a second level is introduced on which state transitions take place: the recording level. Whenever an event occurs in the organisation, it should be communicated to the information system by means of an update request. The processing of this update request, i.e. the recording of the event, should result in an appropriate state transition in the information system. The point of time at which the recording of an event takes place in the information system, is called the recording time of that event.

The resulting state transition is more than a single transition of an application model state, it can be seen as a transition of the complete application model history modelling the history of the organisation up to the occurence of the newly recorded event. So this transition corresponds to a transition of a complete state transition graph. A sequence of these application model history transitions due to successive recordings is

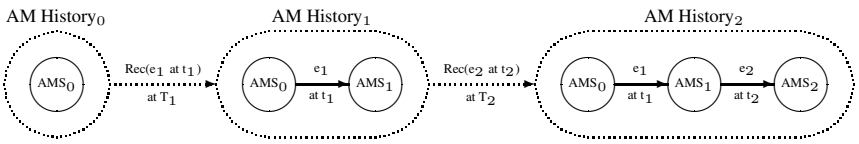

Figure 7: Application Model History (AMH)transitions on the recording level

called an application model recording history. Such an application model recording history reflects both the events occurring in the organisation, and the recordings of these events in the information system.

In figure 7, the graphical representation of application model state transitions (due to events in the organisation) on the event level is extended with the above discussed second level, the recording level, on which the application model history transitions (due to recording of these events) take place. The arrows representing transitions between application model histories are labeled with the denotation of the recording of the event causing that transition, and the recording time of the event in question.

As can be seen in the example, an application model recording history is determined by a set of recordings. In general, the application model recording history, for a given sequence of recordings, is identified by:

Definition 6.7 *Let R be a list of recordings, then $\mathsf{AMRH}(R) = \langle R, \sigma^{amh}, H^{amh} \rangle$, where E is a list of event occurences such that $E(i) = \mathsf{RcOf}(R(i))$ and $dom(R) = dom(E)$, and furthermore that:*

$$\begin{aligned} \sigma^{amh} &= \mathsf{AMH}(E^0) \\ H^{amh} &= \{\ \langle \mathsf{AMH}(E^i), \mathsf{AMH}(E^{i+1}), R(i+1) \rangle\ \mid 0 \leq i < |E|\ \} \end{aligned}$$

6.3 The Correction Level

As a user may make mistakes when entering recordings of event occurences into the information system, three kinds of correction have been identified in section 3. A recorded event may have to be replaced by another one, a recording may have to be inserted in the sequence of already recorded events, and a recording may have to be removed. As shown in section 5, these three kinds of recordings can be implemented by means of a roll-back and a series of re-recordings of already recorded (correct) events. In all cases where a correction is needed, a roll-back should take place to the latest correct application model history.

In order to process the primitive user's input requests ($\mathcal{PI}$) the roll-backs, contained in the input, have to be ordered in time. By means of this order, the recordings can be split up in a list of sets of recordings, such that each of these sets implies an application model recording history. The roll-backs in the primitive user input ($\mathcal{PI}_{rb}$), can be ordered according to the following definition:

Definition 6.8 *Due to lemma 5.2 there exists a strict total order on the elements of $\mathcal{PI}_{rb}$. Let $B_{\mathcal{PI}}(i)$ denote the i-th rollback in $\mathcal{PI}_{rb}$ based on this order. So $dom(B_{\mathcal{PI}}) = \{1, \ldots, |\mathcal{PI}_{rb}|\}$*

In figure 8, the performance of a correction by means of a roll-back is graphically represented in the case of a *replacement* of the recording of an event e_1 having event time t_1 by a recording of event e_1' with the same event time. The replacement is presumed to take place after the recording of event e_2.

A sequence of successive recordings, an application model recording history, can be seen as the believed world (organisation) of the information system. A a correction of this belief of the world is performed by means of a roll-back, causing a transition of the current application model recording history in the information system. A sequence of these application model recording history transitions due to roll-backs is called the application model evolution, which is said to take place on the correction level.

In figure 9 the situation of figure 8 is represented in an alternative way, identifying the three levels of state transitions more clearly. Note that the roll-back performed by the correction is implicitly present in this figure. In the same way corrections requiring the removal or insertion of a recording of an event can be represented. In [FOP92a] more examples are given and elaborated.

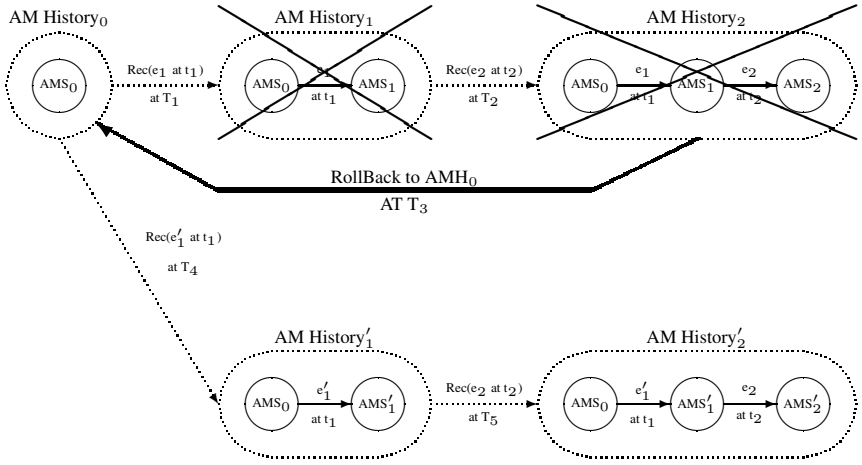

Figure 8: Correction by means of a roll-back

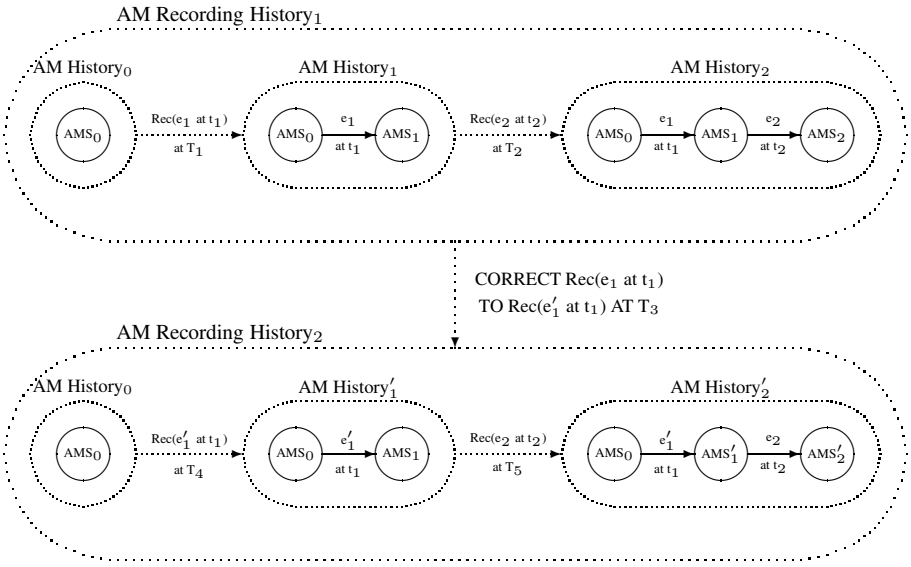

Figure 9: Application Model recording History (AMRH) transition on the correction level

As stated before, the recordings of the user's update requests have to be split up, based on the roll-backs, in a list of sets of recordings. Each of these sets will correspond to an application model's recording history. This distribution of recordings is given by:

Definition 6.9 *$U_{\mathcal{PI}}$ is a list of sets of recordings such that $dom(U_{\mathcal{PI}}) = \{0, \ldots, |\mathcal{PI}_{rb}|\}$, and:*

$$
\begin{aligned}
U_{\mathcal{PI}}(i) = \ & \textbf{if } i = 0 \textbf{ then} \\
& \quad \{r \in \mathcal{PI}_{rc} \mid \ |dom(B_{\mathcal{PI}})| \geq 1 \ \Rightarrow \ r <_{\mathcal{T}_r} B_{\mathcal{PI}}(1) \ \} \\
& \textbf{else} \\
& \quad \{r \in U_{\mathcal{PI}}(i-1) \mid \mathsf{RcAt}(r) < \mathsf{RbTo}(B_{\mathcal{PI}}(i))\} \cup \\
& \quad \{r \in \mathcal{PI}_{rc} \mid \ |dom(B_{\mathcal{PI}})| \geq i+1 \ \Rightarrow \ r <_{\mathcal{T}_r} B_{\mathcal{PI}}(i+1) \ \} \\
& \textbf{fi}
\end{aligned}
$$

The initial set of recordings $U_{\mathcal{PI}}(0)$ contains all recordings in $\mathcal{PI}_{rc}$ which have not been undone by the first roll-back (if present). The successive sets of recordings contain all recordings in the previous set of recordings $U_{\mathcal{PI}}(i-1)$ which have not been rolled-back by $B_{\mathcal{PI}}(i)$, and the set of new recordings done before the next roll-back at $B_{\mathcal{PI}}(i+1)$ (if present).

The sets of recordings in the above defined list have to be ordered themselves as well. Each of the resulting ordered list of recordings of events implies an application's model recording history. Due to axiom 4.1, and definition 6.9, $<_{\mathcal{T}_r}$ defines a strict total order on the recordings in $U_{\mathcal{PI}}(i)$. This allows for the following sequence of the recordings in $U_{\mathcal{PI}}$:

Definition 6.10 *$R_{\mathcal{PI}}(i)$ is a list of lists of recordings such that $R_{\mathcal{PI}}(i)(j)$ is the j-th recording based on $<_{\mathcal{T}_r}$ in $U_{\mathcal{PI}}(i)$, and such that $dom(R_{\mathcal{PI}}) = dom(U_{\mathcal{PI}})$. So for all $i \in dom(R_{\mathcal{PI}})$ the following holds: $dom(R_{\mathcal{PI}}(i)) = \{1, \ldots, |R_{\mathcal{PI}}(i)|\}$.*

The evolution of the application model (AMEV) for a given user input $\mathcal{PI}$ is now defined as:

Definition 6.11 $\mathsf{AMEV}(\mathcal{PI}) = \langle \mathcal{PI}_{rb}, \sigma^{amrh}, H^{amrh} \rangle$*, where:*

$$
\begin{aligned}
\sigma^{amrh} &= \mathsf{AMRH}(R_{\mathcal{PI}}(0)) \\
H^{amrh} &= \{ \ \langle \mathsf{AMRH}(R_{\mathcal{PI}}(i)), \mathsf{AMRH}(R_{\mathcal{PI}}(i+1)), B_{\mathcal{PI}}(i+1) \rangle \ \mid 0 \leq i < |\mathcal{PI}_{rb}| \ \}
\end{aligned}
$$

Finally, an overview of all the defined concepts is provided in the conceptual schema (meta schema), of the framework of the processing of the user update requests, in figure 10. The schema depicted there is in the style of PSM ([HW93], [HPW92b]), being an extension of the modelling technique from NIAM ([NH89], [Win90]). The extension used in figure 10 deals with schema objectifications, where a population of the objectified schema at hand is to be looked upon as an abstract object instance. The schemas contained in the graphical denotation of AMRH, AMH and AMS are objectified schemas, each reflecting a level of the discussed three level framework for updates. Note that the three objectified schemata are nested.

7 Conclusions & Further Research

In this paper the importance of evolving information systems has been shown. These systems are able to evolve at the same pace, and to the same extent as organisations do. This makes organisations more flexible, allowing them to react more quickly on changes in their dynamic environment.

An evolutionary approach to information systems development has been advocated which should result in evolving information systems. On the basis of requirements, and an general architecture for evolving information systems, the distinction with traditional information systems was explained. Traditional information systems appeared to be degenerations of evolving information systems.

In order to handle temporal and evolutionary aspects in an evolving information system (as well as in an historical information system), the traditional notion of update was revised, resulting in the triple: recording, correction and forgetting. It was stated that update should not forget any aspect ever fed to the system, unless explicitly asked for. The notion of updating an application model was described by

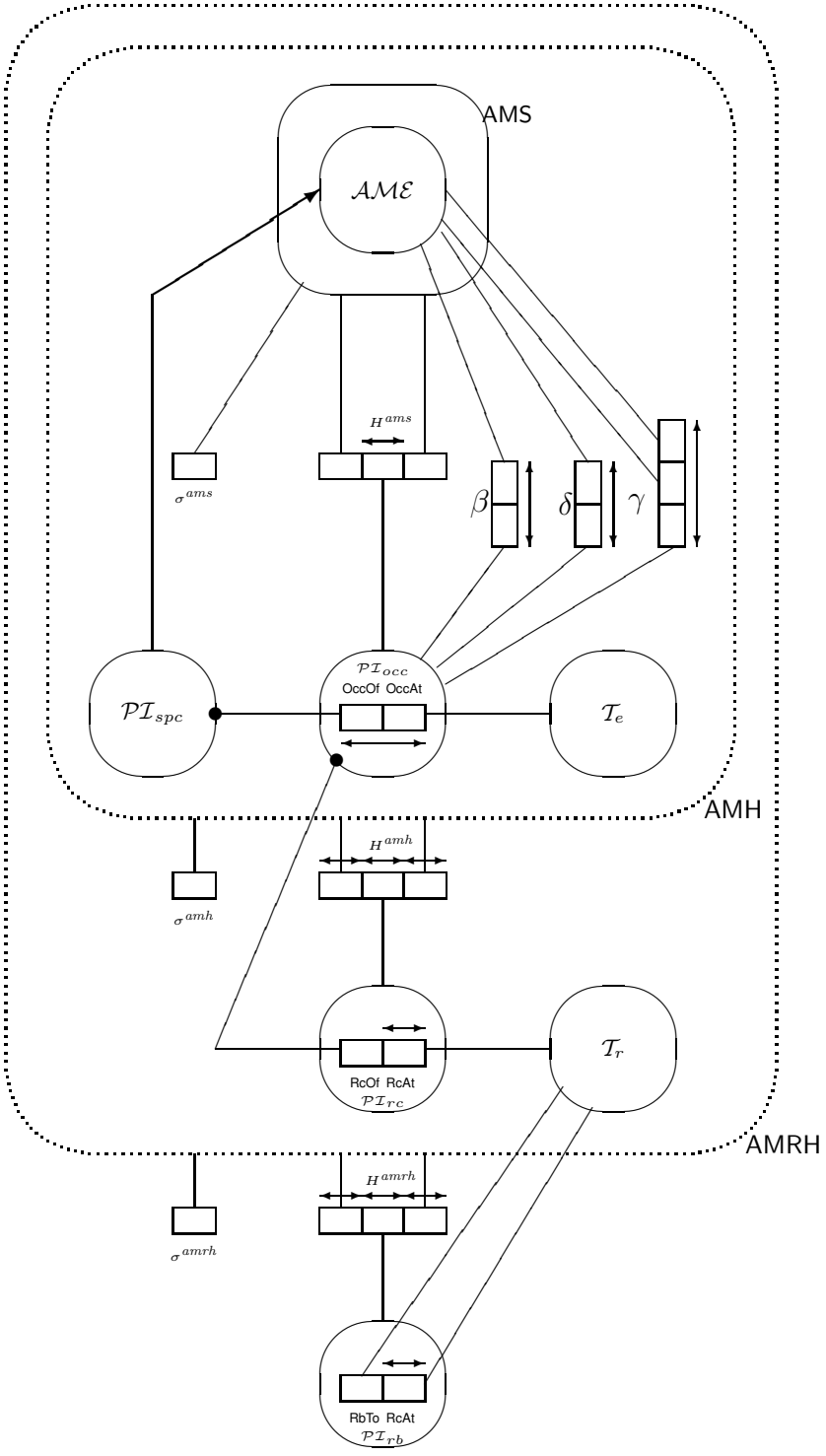

Figure 10: A Meta Model for the Processing of Update Requests

introducing a state-transition-oriented model on three levels of abstraction (organisation, recording and correction level). This model has been formalised.

An evolving information system shell is being developed on the basis of the meta model presented in this paper. This meta model should be extended with a general theory for the evolution of application models, and concrete modelling techniques for world models and action models. At the moment, the presented framework is being implemented.

References

[All84] J.F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 1984(23):123–154, 1984.

[Bem87] Th.M.A. Bemelmans. *Bestuurlijke informatiesystemen en automatisering*. Stenfert Kroese, Leiden, The Netherlands, 3rd edition, 1987. In Dutch.

[BF91] S. Brinkkemper and E.D. Falkenberg. Three Dichotomies in the Information System Methodology. In P.W.G. Bots, H.G. Sol, and I.G. Sprinkhuizen-Kuyper, editors, *Informatiesystemen in beweging*. Kluwer, Deventer, The Netherlands, 1991.

[Bub80] J.A. Bubenko. Information Modelling in the Context of System Development. In S.H. Lavington, editor, *Information Processing 80*, pages 395–411. North-Holland/IFIP, Amsterdam, The Netherlands, 1980.

[Che76] P.P. Chen. The entity-relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[CR87] J. Clifford and A. Rao. A simple, general structure for Temporal Domains. In C. Rolland, F. Bodart, and M. Leonard, editors, *Temporal Aspects in information Systems*, pages 17–28. North-Holland/IFIP, Amsterdam, The Netherlands, 1987.

[FHL+98] E.D. Falkenberg, W. Hesse, P. Lindgreen, B.E. Nilsson, J.L.H. Oei, C. Rolland, R.K. Stamper, F.J.M. Van Assche, A.A. Verrijn-Stuart, and K. Voss, editors. *A Framework of Information Systems Concepts*. IFIP WG 8.1 Task Group FRISCO, 1998. ISBN 3-901-88201-4

[FOP92a] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. A Conceptual Framework for Evolving Information Systems. In H.G. Sol and R.L. Crosslin, editors, *Dynamic Modelling of Information Systems II*, pages 353–375. North-Holland, Amsterdam, The Netherlands, EU, 1992. ISBN 0444894055

[FOP92b] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. A Metamodel for Update in Information Systems. Technical Report 92-05, Department of Information Systems, University of Nijmegen, Nijmegen, The Netherlands, EU, 1992.

[FOP92c] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. Evolving Information Systems: Beyond Temporal Information Systems. In A.M. Tjoa and I. Ramos, editors, *Proceedings of the Data Base and Expert System Applications Conference (DEXA'92)*, pages 282–287, Valencia, Spain, EU, September 1992. Springer Verlag, Berlin, Germany, EU. ISBN 3211824006

[GS86] C. Gane and T. Sarson. *Structured System Analysis: Tools and techniques*. IST Databooks. MacDonald Douglas Corporation, St. Louis, 1986.

[HN93] A.H.M. ter Hofstede and E.R. Nieuwland. Task structure semantics through process algebra. *Software Engineering Journal*, 8(1):14–20, January 1993.

[HPW92a] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. A Note on Schema Equivalence. Technical Report 92-30, Department of Information Systems, University of Nijmegen, Nijmegen, The Netherlands, EU, 1992.

[HPW92b] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Data Modelling in Complex Application Domains. In P. Loucopoulos, editor, *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, volume 593 of *Lecture Notes in Computer Science*, pages 364–377, Manchester, United Kingdom, EU, May 1992. Springer Verlag, Berlin, Germany, EU. ISBN 3540554815

[HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[HPW94] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. A Conceptual Language for the Description and Manipulation of Complex Information Models. In G. Gupta, editor, *Seventeenth Annual Computer Science Conference*, volume 16 of *Australian Computer Science Communications*, pages 157–167, Christchurch, New Zealand, January 1994. University of Canterbury. ISBN 047302313

[HW92] A.H.M. ter Hofstede and Th.P. van der Weide. Formalisation of techniques: chopping down the methodology jungle. *Information and Software Technology*, 34(1):57–65, January 1992.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[ISO87] *Information processing systems – Concepts and Terminology for the Conceptual Schema and the Information Base*, 1987. ISO/TR 9007:1987.
`http://www.iso.org`

[LGN81] M. Lundeberg, G. Goldkuhl, and A. Nilsson. *Information Systems Development - A Systematic Approach*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[MS90] E. McKenzie and R. Snodgrass. Schema evolution and the relational algebra. *Information Systems*, 15(2):207–232, 1990.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989. ASIN 0131672630

[OHFB92] J.L.H. Oei, L.J.G.T. van Hemmen, E.D. Falkenberg, and S. Brinkkemper. The Meta Model Hierarchy: A Framework for Information System Concepts and Techniques. Technical Report 92-17, Department of Information Systems, University of Nijmegen, Nijmegen, The Netherlands, 1992.

[Rod91] J.F. Roddick. Dynamically changing schemas within database models. *The Australian Computer Journal*, 23(3):105–109, August 1991.

[SA85] R. Snodgrass and I. Ahn. A Taxonomy of Time in Databases. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 236–246, Austin, Texas, 1985.

[SA86] R. Snodgrass and I. Ahn. Temporal Databases. *IEEE Computer*, 19(9):35–42, 1986.

[Sno90] R. Snodgrass. Temporal Databases Status and Research Directions. *SIGMOD Record*, 19(4):83–89, December 1990.

[Ver89] A.A. Verrijn-Stuart. Some Reflections on the Namur Conference on Information Systems Concepts. In E.D. Falkenberg and P. Lindgreen, editors, *Information System Concepts: An In-depth Analysis*. North-Holland/IFIP, Amsterdam, The Netherlands, 1989.

[Win90] J.J.V.R. Wintraecken. *The NIAM Information Analysis Method: Theory and Practice*. Kluwer, Deventer, The Netherlands, EU, 1990.

[WJL91] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with the Granularity of Time in Temporal Databases. In R. Andersen, J.A. Bubenko, and A. Sølvberg, editors, *Proceedings of the Third International Conference CAiSE'91 on Advanced Information Systems Engineering*, volume 498 of *Lecture Notes in Computer Science*, pages 124–140, Trondheim, Norway, May 1991. Springer-Verlag.