Evolving Information Systems: Meeting the ever-changing environment

J.L.H. Oei[1], H.A. Proper[2], E.D. Falkenberg[2]

June 23, 2004

[1]University of Twente
Dept. of Computer Science
P.O. Box 217, 7500 AE Enschede
The Netherlands

[2]University of Nijmegen
Faculty of Mathematics and Informatics
Toernooiveld, 6525 ED Nijmegen
The Netherlands
E.Proper@acm.org

PUBLISHED AS:

J.L.H. Oei, H.A. Proper, and E.D. Falkenberg. Evolving Information Systems: Meeting the Ever-Changing Environment. *Information Systems Journal*, 4(3):213–233, 1994.

Abstract

To meet the demands of organisations and their ever changing environment, information systems are required which are able to evolve to the same extent as organisations do. Such a system has to support changes of all time- and application-dependent aspects. In this paper, requirements and a conceptual framework for evolving information systems are presented. This framework includes an architecture for such systems, and a revision of the traditional notion of update. Based on this evolutionary notion of update (recording, correction, and forgetting) a state-transition-oriented model on three levels of abstraction (event level, recording level, correction level) is introduced. Some examples are provided to illustrate the conceptual framework for evolving information systems.

1 Introduction

Due to the dynamic behaviour of organisations, and their environment, organisations 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. However, organisations are increasingly faced with the problem of obsolete information systems. As a result, information needs are not met, not adequately met, or not met in time. The absence or overdue arrival of correct information makes adequate management impossible and rightly irritates the user. Besides the problem of the indadequate supply of information, the rise of automation costs (with regard to development, production and maintenance) is also increasingly causing concern ([VW91]).

Thus, in order to cope with rapidly evolving application domains, information systems are needed which are more flexible than the current generation of information systems are. These information systems, which are able to evolve on-line, are called evolving information systems ([FOP92a], [FOP92c]). A further requirement for evolving information systems is, that they do not forget any information ever fed to them (unless they are explicitly asked to do so).

This paper discusses the need, the requirements, and a conceptual framework for a generalized evolving information system. This framework for evolving information systems includes fundamental concepts, and an architecture for such systems, in which a distinction is made between a part which is both application-independent and time-invariant, and a part which is not. The description of the first part is contained in the *meta model*, while the latter part is described in the *application model*. The conceptual framework as presented in this paper, forms the basis for the meta model for such a generalized evolving information system. This meta model deals with all conceptual aspects of evolution. 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. This becomes particularly important for large organizations with various sorts of applications, in which furthermore new sorts of applications may become necessary from time to time.

The need for the support of evolution in information systems has already been recognized by others. However, most of them restrict themselves to evolution of only part of the application model, eg. schema evolution ([MS90], [Ari91], [Rod91]). In [MS90] a relational algebra is presented in which relational tables are allowed to evolve, e.g. change their arity. In this paper, we take a more conceptual approach to evolution of information systems. Furthermore, we do not restrict evolution to the data model (and its population) only. Others discuss support of evolution by version management (eg. [BCG+87], [Kat90], [JMSV92]). The existence of versions assumes a series of replacements of system versions by new ones, thus allowing interruption of the organization processes. In our evolving information systems, however, only one system version exists at any time, capturing the complete history of information recorded as well. Updates of any part of the application model in this system have to be performed on-line.

In our approach to evolving information systems, there is no essential difference between the development phase of an information system, and the operation and maintenance phase. Our evolving information systems approach is characterized by an iterative life cycle having the length of the organisation's existence. Starting from an empty system, the application model of the system is built up and maintained by processing update requests. These update requests are caused by changes in the organisation and/or changes in the (user) requirements. This evolving information systems approach distinguishes itself from other approaches by making no essential difference between development and operation and maintenance phase, and the additional requirement that adjustments of the information system have to be made without the need to interrupt any processes in the organisation.

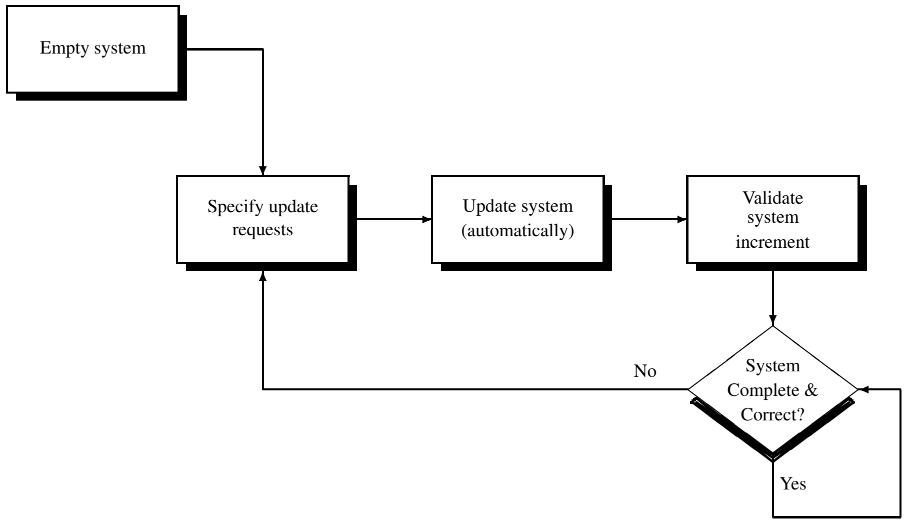

Empty system
Specify update requests
Update system (automatically)
Validate system increment
System Complete & Correct?
No
Yes

Figure 1: Evolving Information Systems approach

A major consequence of the requirements for evolving information systems, is that the notion of time has to be introduced in the meta model. Even more, at least two distinct notions of time have to be distinguished. Events occuring in the organisation will have to be recorded together with their time of occurence (the event time). In order to be able to perform corrections, a roll-back operator is needed (see section 3). 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 the renaming is that the new names correspond better to the three level architecture we will introduce in section 3. The classification of information systems which is made in [SA86] is based on the basis of support of valid and transaction time. Conform this classification (which distinguishes snapshot-, historical-, rollback-, and temporal systems), evolving information systems are temporal systems because of the fact that both valid and transaction time are supported. However, it should be noted that not all temporal systems are evolving information systems. As we have seen in this section, evolving information systems have to meet additional requirements.

2 The Architecture for Evolving Information Systems

In this paper, information systems are considered to be information systems in which the only actor performing information processing activities is computerized. This computerized actor is called the information processor. 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 section, a general architecture for information systems is presented (see also [FOP92a], [FOP92c]), by means of which the distinction between traditional and evolving information systems is explained.

The information processor in an information system accepts input messages (requests), which may reflect changes of a state (events) in the universe of discourse, 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 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 which describes a particular universe of discourse, the application model, and a part which describes the language (technique) in which this application model is specified and can be manipulated. The latter part is the meta model, and provided in a particular information system once and for all. Conversely, 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. The architecture discussed is depicted in figure 2.

An application model can be subdivided further. On the one hand, we need a model of that part of the perceived world (universe of discourse) where the interaction between the information system and the environment is about. This model is called the world model. Many techniques for describing world models distinguish in their language an information structure, a set of constraints defined upon the information structure, and a population of the information structure, conforming to these constraints (eg Entity Relationship Modelling ([Che76]), NIAM ([NH89], [Win90]) or PSM ([HW93], [HPW92]).

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 - we call it the activity model - and a part that describes the (trigger-) relations between the activity model and the world model. We will refer to this latter part as behaviour model. In the behaviour model 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 activity and behaviour model are Data Flow Diagrams ([GS86]) the A-schemas in ISAC ([LGN81]), or Task Structures ([HN93], [WHO92]). The subdivision of the proces model, is illustrated in figure 3.

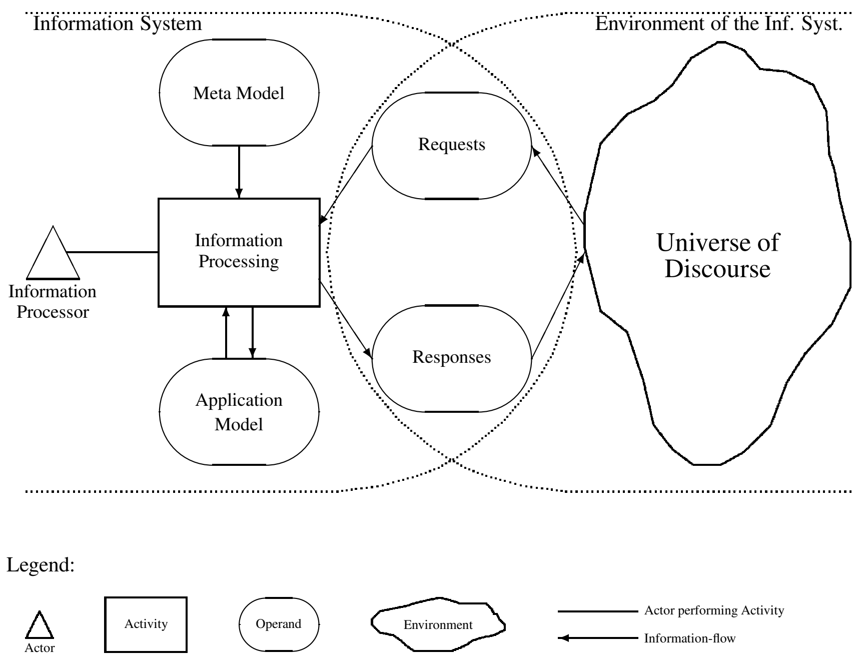


Figure 2: An (Evolving) Information System and its Environment

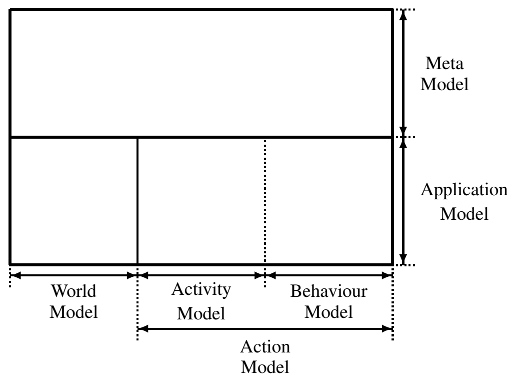


Figure 3: The Structure of the Processing Model

It is only obvious, that the world model should provide a unified conceptual model of the application. Recently, some unified conceptual modelling techniques have been proposed, such as Telos ([JMSV92]), and TMT ([Hof93]). TMT is an integration of the data modelling technique PSM, and the Task Structures activity modelling technique.

Example 2.1 *To illustrate the subdivision of processing models, a possible subdivision of the processing model of an information system supporting the calculation and registration of scholarships for Dutch students is presented. In the Netherlands, every student receives a scholarship from the government. The height of this scholarship depends on the fact whether students live on their own or with their parents, the income of the parents, and also the amount of their extra earnings.*

In figure 4 three different modelling techniques are used for describing the application model of this universe of discourse. The world model part is described in an Entity-Relationship modelling technique, the activity model part uses A-schema's of ISAC ([LGN81]), whereas Event Decomposition Diagrams of Yourdon ([You89]) are used for describing the behaviour model part. The language(s) used for specification of the application model are based on the meta model of the information system. Though we can use the same techniques for describing its meta model, in figure 4 the (partial) meta model is described in another modelling technique (NIAM [NH89]).

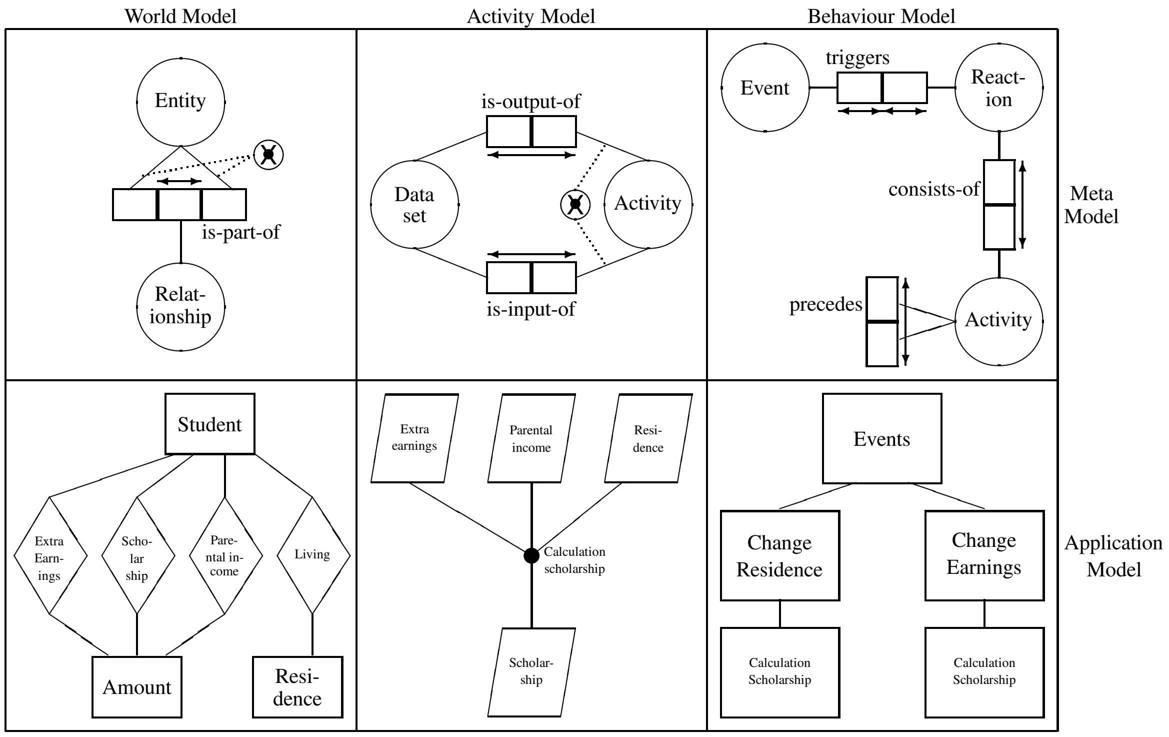


Figure 4: The processing model for the Dutch Scholarship Information System

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, in which the schema vs. instance dichotomy ([BF91]) is applied, only the instances can be updated. That is, schema specifications, as well as activity and behaviour specifications (which are usually hidden in programming procedures), cannot be updated in traditional information systems. The intention of an evolving information system, however, is that the complete application model is updatable.

In the scholarship system example, the need for update of all specifications in the application model is apparent, as the laws of the scholarship system in the Netherlands appear to change frequently. For instance, due to subsequent cuts in the total budget, maximum study time has been increasingly limited over the last few years. Another illustrative example of an evolving application domain, is provided in section 4. A more detailed, and formal, treatise of the notion of schema evolution can be found in [PW93].

Given a meta model for evolving information systems, a software environment for these evolving information systems can be developed which is time-invariant and independent of any universe of discourse. Such an environment is called an evolving information system shell (EIS-shell). Application models describing different domains can be 'plugged' into the EIS-shell. Furthermore, an EIS-shell has to be designed in such a way that it is independent of any software environment, i.e. independent of any database mananagement system and/or operating system.

3 Update in Evolving Information Systems

In [FOP92a] and [FOP92c] a conceptual framework for update in evolving information systems has been introduced. This framework has been formalised in [FOP92b]. In this section, the framework is explained and illustrated by means of our running example.

First of all, the notion of update in evolving information systems is summarized. The traditional notion of update, viz. addition, deletion, and modification is replaced by an evolutionary one which is based on the possible causes for update requests rather than internal database operations. Three kinds of updates are distinguished, viz. recording, correction and forgetting. Recording of an event is the processing of an update request caused by a change of state in the universe of discourse. Update requests are formulated in a language which is based on the meta model of the system. They are communicated to the system by the user.

During the operation phase of the system incorrect recordings may take place. These incorrect recordings are caused by accidental mistakes in the formulation of update requests, or due to incorrect or incomplete information available to the users. Incorrectnesses of these kind can only be detected by empirical validation. An information system reflects an organisation correctly, if and only if there exists an isomorphism between the states in the information system and the states in the organisation system being modelled. The order in which the events occured in the organisation has to be preserved by this mapping ([FOP92a]). The order of processing update requests is of importance because of possible interrelationships between events (as will also be shown in our example). For that reason, whenever it is detected that this constraint is violated, a correction should take place. To accomplish this correction, an operator has been introduced which retrieves a former state. This operator is called the roll-back operator. The use of this roll-back operator is explained in one of the following subsections.

Based on this notion of update, a conceptual framework is presented which distinguishes different types of state transitions on different levels of abstraction in the context of update in evolving information systems. The levels distinguished are called the event level, the recording level, and the correction level, respectively. 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.

3.1 The event level

It is generally assumed that the universe of discourse described in an information system, contains a set of stable states, and that there are a number of actions that result in a change of state (state transitions) (see eg [HW93]). The states and state transitions in a universe of discourse are modelled in an information system. The state of an organisation at a particular point of time is modelled by a set of modelling constructs which we call application model elements. This set of application model elements constitute the application model state. Note that the types of application model elements (eg. objects, entities, relationships, activities, events, triggers etc.) are dependent on the used modelling technique.

A state transition in the organisation is modelled in the information system by means of a transition of the application model state. A transition of an application model state can include more than one elementary transition of an application model element. The elementary transitions involved in a particular application model state transition depend on the trigger relationships between the elementary transitions invoked by the transition in the organisation.

A transition in the organisation taking place at a particular point of time is called an (organisational) event. The point of time at which such an event occurs in the organisation, is called the event time of that event. These events are considered to occur on the organisational level ([FOP92a]). The corresponding transitions in the information system are considered to occur on the so-called event level. A sequence of these application model state transitions is called an application model history (see figure 5).

$$\begin{tikzcd}
\boxed{AMS_0} \arrow[r, "e_1", "\text{at } t_1"'] & \boxed{AMS_1} \arrow[r, "e_2", "\text{at } t_2"'] & \boxed{AMS_2}
\end{tikzcd}$$

Figure 5: Application Model State (AMS) transitions on the event level

Example 3.1 *To illustrate our conceptual framework for update we now continue our example of the Dutch Scholarship Information System. First of all, the application model is specified in a more detailed way.*

World Model:
The information structure of our universe of discourse can be specified in LISA-D ([HPW93]) as follows:

- ENTITY-TYPE Student, Amount, Residence
- LABEL-TYPE Residence-status HAS-DOMAIN {'Parents','Independent'}
- BRIDGE-TYPE Residence HAS Residence-status
- FACT-TYPE Scholarship:(getting-scholarship: Student, being-scholarship-of: Amount),
- FACT-TYPE Earnings:(having-earnings: Student, being-earnings-of: Amount)
- FACT-TYPE Living:(living-at: Student, being-residence-of: Residence)
- FACT-TYPE Parental-income:(having-parents-with: Student, being-parental-income-of: Amount)

Behaviour Model:
There are some rules concerning the calculation of a student's scholarship. One of them is, that when you live with your parents you receive Dfl 300, wheras you receive Dfl 400 when you live on your own. Another rule states that the scholarship is cut whenever you have some extra earnings above a certain amount (200). This can be specified in the behaviour model as follows:

```
• WHEN BIRTH(Student $s$ living-at Residence $r$)
  IF $r$ ='Parents'
  THEN
     CREATE(Student $s$ getting-scholarship Amount 300)
  ELSE
     CREATE(Student $s$ getting-scholarship Amount 400)
  FI

• WHEN BIRTH(Student $s$ having-earnings Amount $y$)
  IF $y > 200$ AND Student $s$ getting-scholarship Amount $z$
  THEN ReduceScholarship($s$,$y$,$z$)
```

Activity Model:
The way how the scholarship is reduced in case of extra earnings above Dfl 200 is specified in the activity model as follows:

```
• ACTIVITY ReduceScholarship ($s$,$y$,$old$)
  BEGIN $new$ := $old - 0.75 * (y - 200)$
     CHANGE (Student $s$ getting-scholarship Amount $old$)
     INTO (Student $s$ getting-scholarship Amount $new$)
  END
```

From now on the history of a student called Jim is observed. From the information we obtained from the environment two events concerning Jim's scholarship has occured. The first event is that Jim started living at his own at January 1st 1990. The second event says that one year later Jim found a job providing him extra earnings of Dfl 300. The application model history caused by these events can be derived from figure 5 by using the following substitutions:

- e1=(BIRTH('Jim' living-at 'Independent') AT 01/01/90)
- e2=(BIRTH('Jim' having-earnings 300) AT 01/01/91)
- t1=01/01/90
- t2=01/01/91
- AMS$_0$={initial state}
- AMS$_1$=AMS$_0$ $\cup$ FACT('Jim' living-at 'Independent')
  $\cup$ FACT('Jim' getting-scholarship 400)
- AMS$_2$=AMS$_1$ $\cup$ FACT('Jim' having-earnings 300)
  $-$ FACT('Jim' getting-scholarship 400)
  $\cup$ FACT('Jim' getting-scholarship 325)

Note that Jim's scholarship had been reduced from 400 (living independently) to 325 (being: $400 - 0.75 * (300 - 200)$*) because of his extra earnings. The substitutions in figure 5 for our example result in figure 6.*

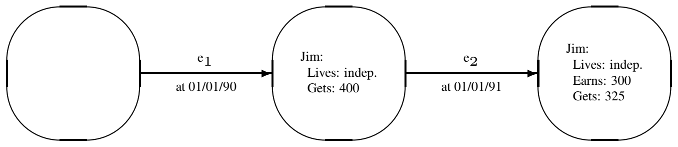


Figure 6: Application Model History for our example

3.2 The recording level

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, called the recording of an 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 which modelled the history of the organisation up to the occurence of the newly recorded event. A sequence of these application model history transitions due to successive recordings is 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 an application model recording history is given.

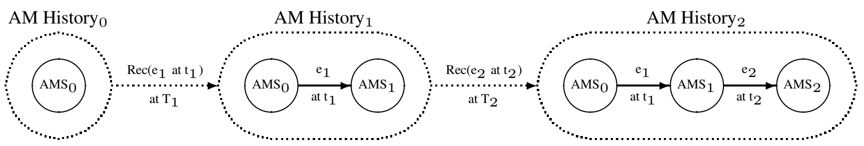


Figure 7: Application Model History (AMH) transitions on the recording level

Example 3.2 *In our Dutch Scholarship Information System update requests are processed at the end of every month. The recordings of the two events e1 and e2 are formulated in a language based on our framework for update.*

- RECORD e1(BIRTH('Jim' living-at 'Independent') AT 01/01/90) AT 31/01/90
- RECORD e2(BIRTH('Jim' having-earnings '300') AT 01/01/91) AT 31/01/91

The application model recording history resulting from these recordings is obtained by further substituting T1=31/01/90 and T2=31/01/91 in figure 7. This results in an application model recording history for our example, which is represented in figure 8.

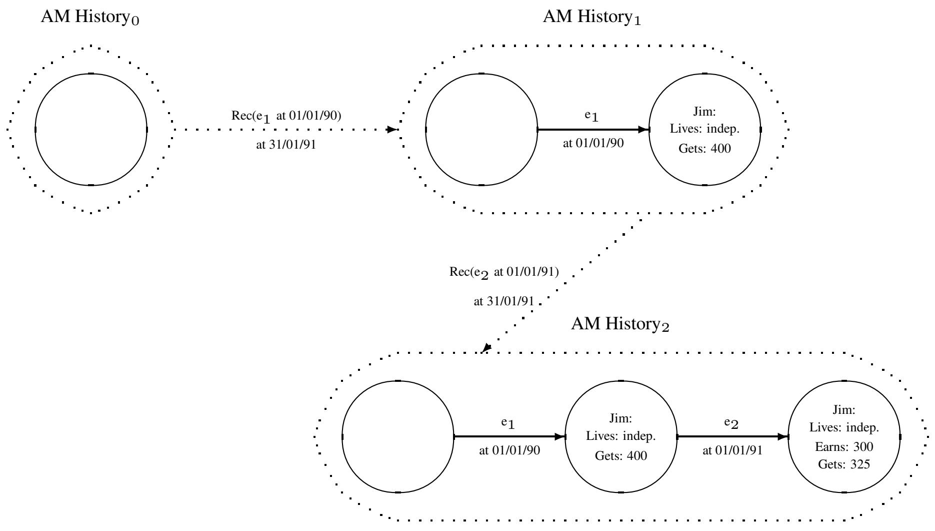


Figure 8: Application Model Recording History for our example

3.3 The correction level

In the process of recording events, mistakes can be made. By validation it may appear that information about events in the organisation which have been recorded in the information system are empirically wrong. To perform corrections, an operation has to be introduced which makes it possible to go back in a sequence of successive recordings. This operation is called the roll-back operation.

In all cases which need a correction, ie. the mapping between organisation system and information system is not isomorphic, a roll-back should take place to the latest application model history which is correct. A replacement, removal, and insertion of a recording of an event require a roll-back to the appropriate application model history in the application model recording history of the information system. After performing the appropriate roll-back, all correct (rolled-back) events have to be re-recorded. In the case of a replacement and an insertion, the first event recorded after the roll-back is the replacing event, and the event to be inserted, respectively. In figure 9, the performance of a correction by means of a roll-back is represented.

A sequence of successive recordings, i.e. an application model recording history, can be seen as the belief of the world (organisation) by the information system. 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 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.

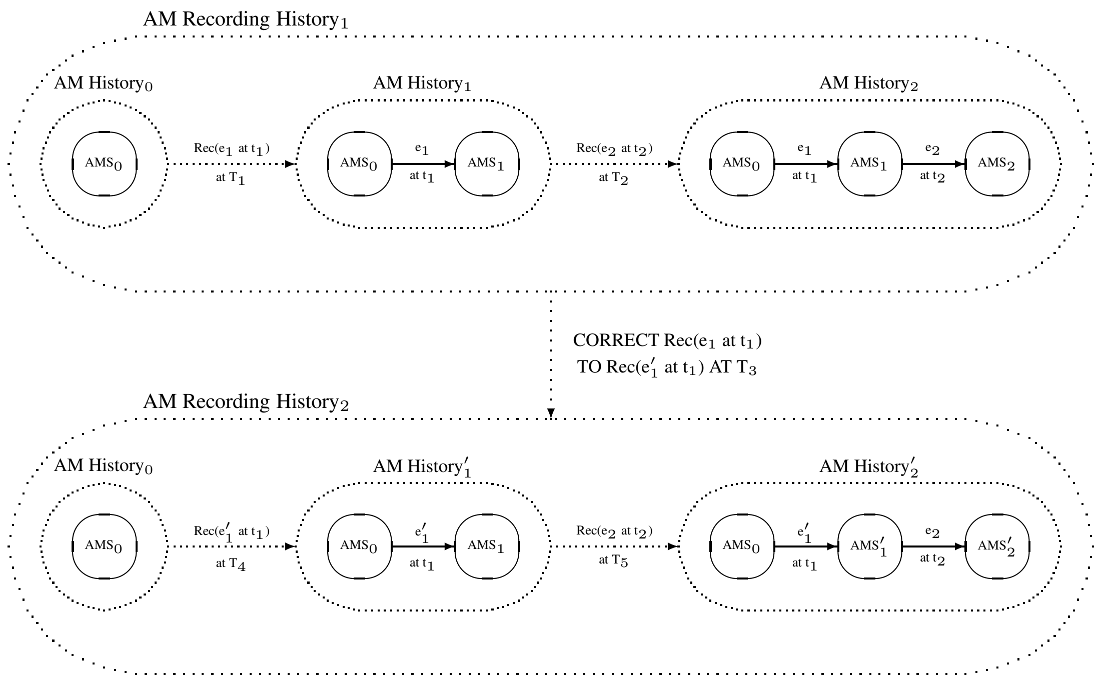


Figure 9: Application Model recording History (AMRH) transition on the correction level

Example 3.3 *Suppose in our running example that at 31/01/92, it is detected that Jim did not live independently when he became student at 01/01/90, but that he still lived with his parents. The update request for correcting this mistake is formulated as follows:*

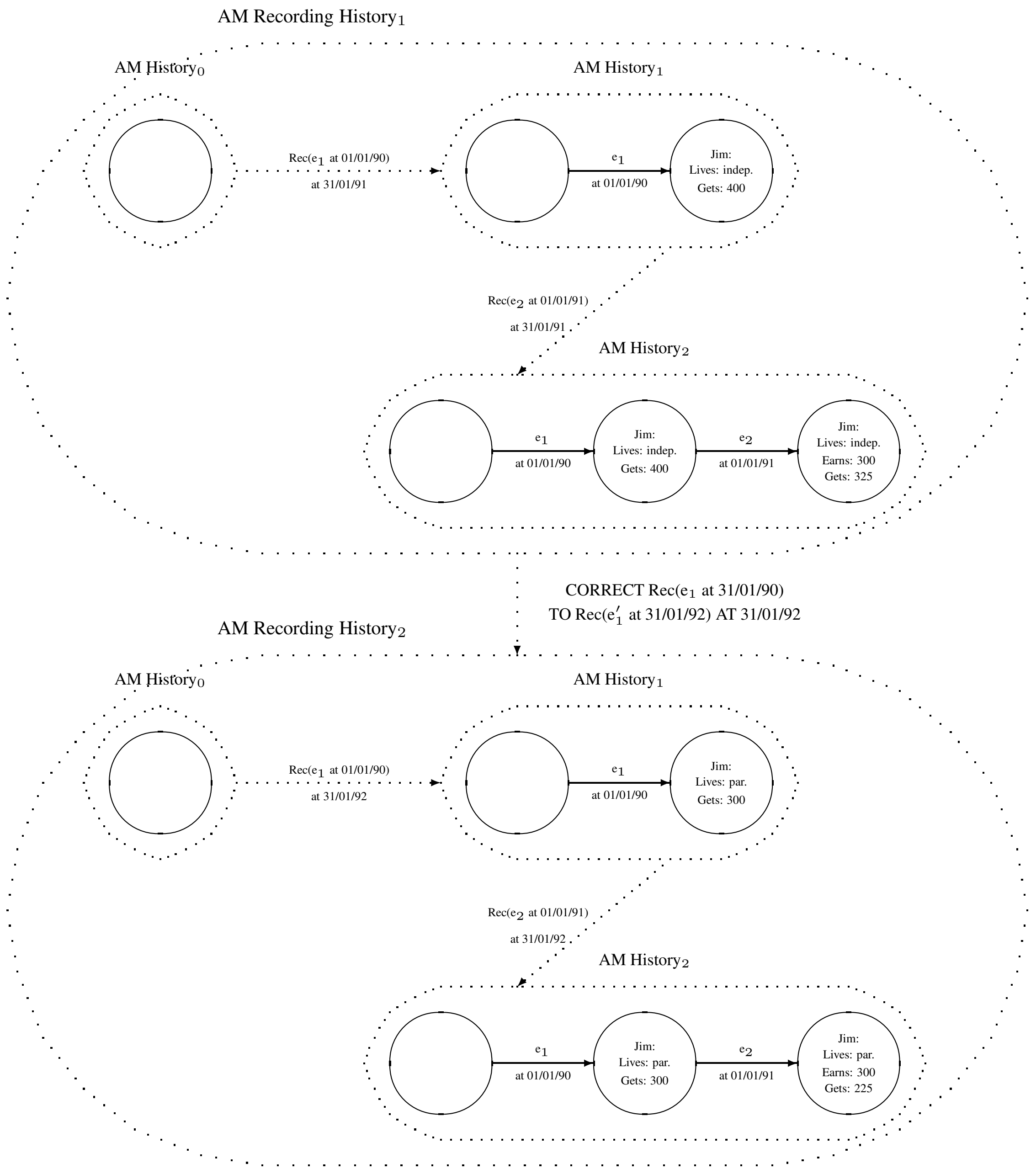
- CORRECT RECORD e1(BIRTH('Jim' living-at 'Independent') AT 01/01/90)
  BY RECORD e1′(BIRTH('Jim' living-at 'Parents') AT 01/01/90) AT 31/01/92

This correction is obtained from figure 9 by extending the substitutions of the previous figures with:

- T3=T4=T5=31/01/92 and
- AMS$'_1$= AMS$_0$ $\cup$ FACT('Jim' living-at 'Parents')
  $\cup$ FACT('Jim' getting-scholarship 300)
- AMS$'_2$= AMS$'_1$ $\cup$ FACT('Jim' having-earnings 300)
  $-$ FACT('Jim' getting-scholarship 300)
  $\cup$ FACT('Jim' getting-scholarship 225)

Note that this example shows that it is really important to roll-back the system to the latest correct state and to perform the re-recordings of events afterwards. It is insufficient just to correct the latest state. Suppose that we only replaced the FACT('Jim' living-at 'Independent') by FACT('Jim' living-at 'Parents') in the latest application model state (AMS$_2$), it would have cost the Dutch government a lot of money, because of the fact that Jim would still have received a scholarship of 325 instead of the correct 225. The resulting application model evolution for our running example is represented in figure 10.

This concludes the explanation of the conceptual framework for update in evolving information systems. It should be noted that the complete framework has been formalised ([FOP92b]). On the basis of this formalisation a prototype of a generalised EIS-shell is being implemented.



AM Recording History$_1$
AM History$_0$
AM History$_1$
Rec(e$_1$ at 01/01/90) at 31/01/91
e$_1$ at 01/01/90
Jim: Lives: indep. Gets: 400
Rec(e$_2$ at 01/01/91) at 31/01/91
AM History$_2$
e$_1$ at 01/01/90
Jim: Lives: indep. Gets: 400
e$_2$ at 01/01/91
Jim: Lives: indep. Earns: 300 Gets: 325
CORRECT Rec(e$_1$ at 31/01/90) TO Rec(e$'_1$ at 31/01/92) AT 31/01/92
AM Recording History$_2$
AM History$_0$
AM History$_1$
Rec(e$_1$ at 01/01/90) at 31/01/92
e$_1$ at 01/01/90
Jim: Lives: par. Gets: 300
Rec(e$_2$ at 01/01/91) at 31/01/92
AM History$_2$
e$_1$ at 01/01/90
Jim: Lives: par. Gets: 300
e$_2$ at 01/01/91
Jim: Lives: par. Earns: 300 Gets: 225

Figure 10: Application Model Evolution for our example

4 Schema Evolution in Evolving Information Systems

The framework for update in evolving information systems does not yet take the evolution of schemas into consideration. Nevertheless, in [PW93], [PW95], [Pro94], a formal discussion of this topic is provided. In this article we will only provide an example of schema evolution. It should be noted, however, that the framework for update is applicable for updates of any part of the application model. An event can be any change of state of the application model. That is, an event can also be a change of the schema, or the action rules in the application model. In the following an example is provided of an evolving application domain, which involves both changes of the schema, and the action rules.

Consider a rental store for audio records (LP's). In this store a registration is maintained of the songs that are recorded on the available LP's. In order to keep track of the wear and tear of LP's, the number of times an LP has been lent, is registered. The information structure and constraints of this universe of discourse are modelled in figure 11 in the style of ER, according to the conventions of [You89]. Note the special notation of attributes (Title) using a mark symbol (#) followed by the attribute (# Title).

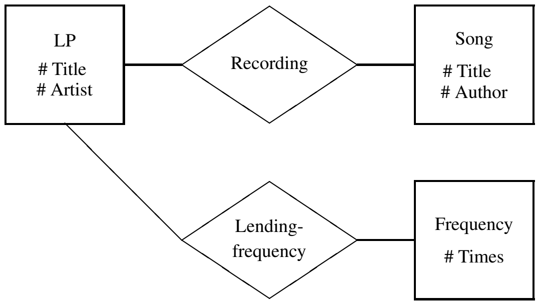


Figure 11: The information structure of an LP rental store

An action specification in this example is the rule Init-freq, stating that whenever a new LP is added to the assortment of the store, it's lending frequency must be set to 0:

```
ACTION Init-freq =
WHEN ADD Lp:$x$ DO
ADD Lp:$x$ has Lending-frequency of Frequency:$0$
```

After the introduction of the compact disc, and its conquest of a sizeable piece of the market, the rental store has transformed into an 'LP and CD rental store'. This leads to the introduction of object type Medium as a generic term for LP and CD. The relation type Medium-type effectuates the subtyping of Medium into LP and CD. In the new situation, the registration of songs on LP's is extended to cover CD's as well. The frequency of lending, however, is not kept for CD's, as CD's are hardly subject to any wear and tear. As a consequence, the application model has evolved to figure 12.

The action specification Init-freq evolves accordingly, now stating that whenever a medium is added to the assortment of the rental store, it's lending frequency is set to 0 provided the medium is an LP:

```
ACTION Init-freq =
WHEN ADD Medium:$x$ DO
IF Lp:$x$ THEN
ADD Lp:$x$ has Lending-frequency of Frequency:$0$
```

After some years, the CD's have become more popular than LP's. Consequently, the rental store has decided to stop renting LP's and to become a CD rental store. This change in the rental store, leads to the information structure as depicted in figure 13. As a result of this evolution step, the action specification Init-freq can be deleted, since the lending frequency of CD's is not recorded anymore.

The three ER schemata, and the associated action specifications, as discussed above, correspond to three distinct snapshots of an evolving universe of discourse. Several approaches can be taken to the modelling

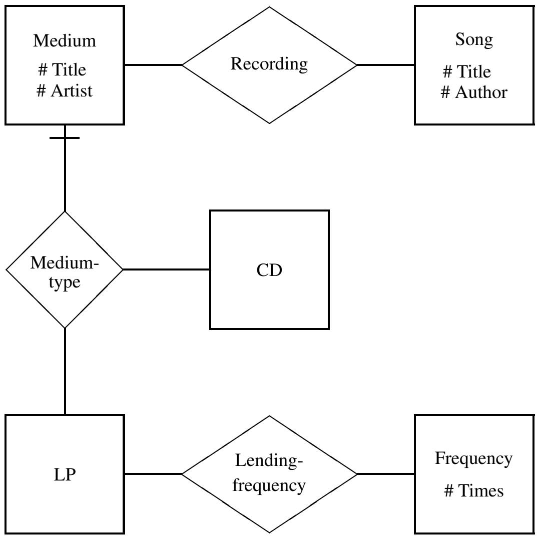


Figure 12: The information structure of a LP and CD rental store

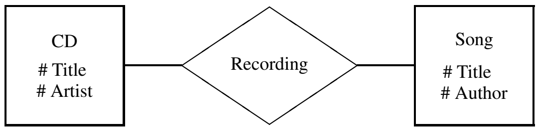


Figure 13: The information structure of a CD rental store

of this evolution (see for a more elaborate discussion [PW93]). We consider evolution, or rather the time axis, of an application model as a separate concept. The evolution of distinct application model elements is maintained, thus keeping track of the evolution of individual object types, instances, methods, etc. This approach enables one to state well formedness rules about, and query, the evolution of distinct application model elements. Furthermore, a snapshot view showing the distinct versions of the application models in the course of time, can be derived by constituting the application model version of any point of time from the current versions of its components.

5 Conclusion

To meet the demands of organisations and their ever changing environment, this paper presented the requirements and a conceptual framework for evolving information systems. An architecture was presented which divided the processing model in an application-independent and time-invariant part, the meta model, and a part which is application-dependent and/or time- variant, the application model. Another subdivision was made into a world model, activity model and behaviour model. Unlike traditional information systems, evolving information systems allow update of all application dependent aspects, ie. the complete application model, without the need to interrupt the processes in the organisation.

In order to handle temporal and evolutionary aspects in an evolving information system, we revised the traditional notion of update, resulting in the triple: recording, correction and forgetting. With this notion of update, we required the meta model to provide concepts and axioms supporting the update of all constituent parts of the application model. Furthermore, we required an evolving information system not to forget any aspect ever fed to the system, unless explicitly asked for. The notion of updating the application model was clarified by introducing a state-transition-oriented model distinguishing three levels of abstraction (event, recording and correction level). This framework was illustrated by a concrete example. The conceptual framework proposed in this paper is the basis of a meta model for update in a generalized evolving information system. In this meta-modelling process, further work is being done. This work involves the formalization of the meta model ([FOP92b], [PW93]), and the design of a language for manipulating and specifying application models. Furthermore, a (prototype) information system shell based on that meta model and that language is being implemented, and a design method is developed for the process of building up and maintaining an application model of an evolving information system based on the presented evolving information systems approach.

Acknowledgements

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).

References

[Ari91] G. Ariav. Temporally oriented data definitions: Managing schema evolution in temporally oriented databases. *Data & Knowledge Engineering*, 6(6):451–467, 1991.

[BCG+87] J. Banerjee, H.-T. Chou, J.F. Garza, W. Kim, D. Woels, and N. Ballou. Data Model Issues for Object-Oriented Applications. *ACM Transactions on Office Information Systems*, 5(1):3–26, 1987.

[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.

[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.

[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.

[Hof93] A.H.M. ter Hofstede. *Information Modelling in Data Intensive Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[HPW92] 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.

[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.

[JMSV92] M. Jarke, J. Mylopoulos, J.W. Schmidt, and Y. Vassiliou. DAIDA: An Environment for Evolving Information Systems. *ACM Transactions on Information Systems*, 20(1):1–50, January 1992.

[Kat90] R.H. Katz. Toward a Unified Framework for Version Modelling in Engineering Databases. *ACM Computing Surveys*, 22(4):375–408, 1990.

[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.

[Pro94] H.A. Proper. *A Theory for Conceptual Modelling of Evolving Application Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1994. ISBN 909006849X

[PW93] H.A. Proper and Th.P. van der Weide. Towards a General Theory for the Evolution of Application Models. In M.E. Orlowska and M.P. Papazoglou, editors, *Proceedings of the Fourth Australian Database Conference*, Advances in Database Research, pages 346–362, Brisbane, Australia, February 1993. World Scientific, Singapore. ISBN 981021331X

[PW95] H.A. Proper and Th.P. van der Weide. A General Theory for the Evolution of Application Models. *IEEE Transactions on Knowledge and Data Engineering*, 7(6):984–996, December 1995.

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

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

[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.

[VW91] Th.H. Visschedijk and R.N. van der Werff. (R)evolutionary system development in practice. *Journal of Software Research*, pages 46–57, December 1991. Special Issue.

[WHO92] G.M. Wijers, A.H.M. ter Hofstede, and N.E. van Oosterom. Representation of Information Modelling Knowledge. In V.-P. Tahvanainen and K. Lyytinen, editors, *Next Generation CASE Tools*, volume 3 of *Studies in Computer and Communication Systems*, pages 167–223. IOS Press, 1992.

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

[You89] E. Yourdon. *Modern Structured Analysis*. Printice-Hall, Englewood Cliffs, New Jersey, 1989.