# Evolving information systems: meeting the ever-changing environment

J. L. H. Oei, H. A. Proper* & E. D. Falkenberg†

*University of Twente, Department of Computer Science, PO Box 217, 7500 AE Enschede, The Netherlands*, Department of Computer Science, University of Queensland, Queensland 4072, Australia and †University of Nijmegen, Faculty of Mathematics and Informatics, Toernooiveld, 6525 ED Nijmegen, The Netherlands*

**Abstract.** *To meet the demands of organizations and their ever-changing environment, information systems are required which are able to evolve to the same extent as organizations do. Such a system has to support changes in 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. Examples are provided to illustrate the conceptual framework for evolving information systems.*

*Keywords:* evolving information systems, information system concepts, meta modelling, schema evolution, temporal information systems.

## 1 INTRODUCTION

Owing to the dynamic behaviour of organizations and their environment, organizations 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, organizations are increasingly faced with the problem of obsolete information systems. Information needs are not met, not adequately met or not met in time. The absence of overdue arrival of correct information makes adequate management impossible and rightly irritates the user. Besides the problem of the inadequate information supply of information, the increase in automation costs (with regard to development, production and maintenance) is also increasingly causing concern (Visschedijk & van der Werff, 1991).

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. Information systems which are able to evolve to the same extent and at the same pace as their underlying organizations are called evolving information systems (Falkenberg *et al.*, 1992a–c; Oei *et al.* 1992a). 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 the architecture, a distinction is made between a part that is application independent and time invariant and a part that is not. The description of the former part is contained in the meta model, while the latter part is described in the application model.

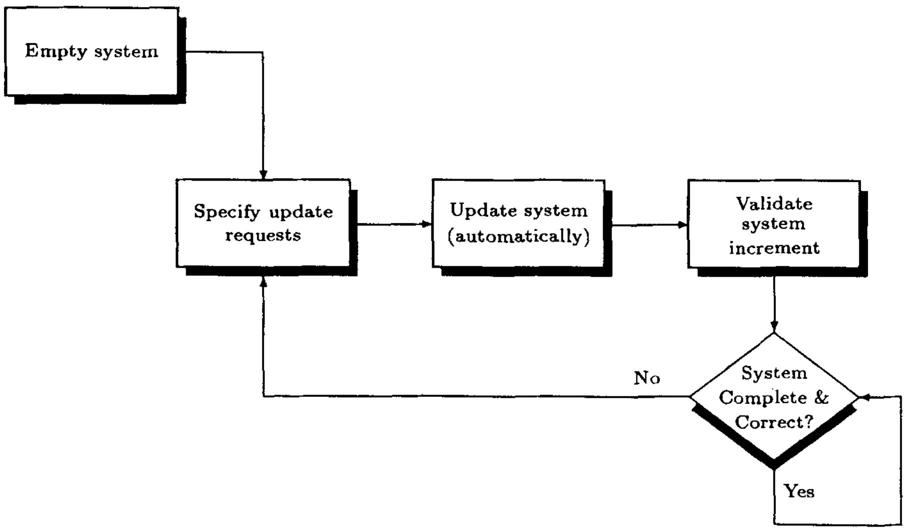
In the process of the development of a generalized evolving information system, we distinguish three subobjectives:

1 The design of a meta model and a language based on it. This language must be able to support all aspects of evolution.
2 The implementation of a generalized information system shell which is based on that meta model and language.
3 A suitable method for the process of designing, building up and maintaining an 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, i.e. the ability to update all the constituent parts of the application model, without forgetting any aspect ever fed to the system, unless explicitly asked for. Furthermore, update is not allowed to interrupt the activities of the organization system. It should be noted that we do not deal with the evolution of user interfaces, implementation and technical aspects.

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. In accordance with the terminology introduced by Oei *et al.* (1992b), this means that in this paper we restrict ourselves to information systems supporting first-order evolution. Second-order evolution involves changes in the meta model. This becomes particularly important for large organizations with various sorts of application, and in which new sorts of applications also become necessary from time to time. An approach towards second-order evolution is considered in Oei *et al.* (1992b) and Oei & Falkenberg (1994).

The need for 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, e.g. schema evolution (McKenzie & Snodgrass, 1991; Ariav, 1991; Roddich, 1991). In McKenzie & Snodgrass (1990) a relational algebra is presented in which relational tables are allowed to evolve, e.g. change of their arity. In this paper, we take a more conceptual approach to the evolution of information systems. Furthermore, we do not restrict evolution to the data model (and its population) only. We allow evolution of the application model as a whole. Others discuss support of evolution by version management (e.g. Banerjee *et al.*, 1987; Katz, 1990; Jarke *et al.*, 1992). 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 and captures the complete history of information recorded as well. Updates of any part of the application model in



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

**Figure 1.** Evolving information systems approach.

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 organization'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 organization and/or changes in the (user) requirements. This evolving information approach is represented in Fig. 1.

As stated before, the main requirement for an evolving information system is that it is able to revolve to the same extent and at the same pace as the underlying organizations. The notions of 'to the same extent' and 'at the same pace' are now refined in more detail.

1 The information system allows update of all information that is dependent on the specific organization domain (universe of discourse) of the information system. Our notion of update which includes recording, correction and forgetting, is discussed in section 3. The specification of information which is dependent on a specific organization domain is part of the architecture for evolving information systems, as is discussed in section 2.

2 The information system allows correction of all information (previously) recorded in the system. Information which has been recorded in the information system may appear to be (empirically) invalid. In evolving information systems correction of this invalid information is possible. The notion of correction is discussed in section 3. Note that the need for correction results from validation and not from verification. Consistency 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 organization. The intention of evolving information systems is to minimize the discrepancy between the information needs of the organization and the information supply by the information system. For that reason, the information system is required to remain available to users of the system in the case of updates.

A major consequence of these requirements is that the notion of time has to be introduced to meet these requirements. Further, at least two distinct notions of time have to be distinguished. It will be obvious that to meet requirement 3 events in the organization have to be recorded together with their time of occurrence. The point of time at which an event occurs in the organization is called the event time of that event. To perform corrections a roll-back operator is needed (see below).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 must be known. These points of time are called the recording times of events.

Our notions of event time and recording time are identical to the notions of valid time and transaction time, respectively, in Snodgrass & Ahn (1986). (The reason for this renaming is that the new names correspond better to the level architecture that will be introduced in section 3.) The classification which is made in Snodgrass & Ahn (1986) is based on the support of valid and transaction time. In accordance with this classification (which distinguishes snapshot, historical, roll-back and temporal systems), evolving information systems are temporal systems because 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

The information systems which we will consider are restricted to information systems in which the only actor performing information processing activities is computerized. This computerized actor is called the information processor. The information processor may be formed from several subprocessors, which may also be (physically) distributed. In this paper, however, the specific aspects of distributed information systems are not taken into consideration.

The restriction to a computerized actor performing information system processing activities corresponds to what has been defined (Verrijn–Stuart, 1989) as an information system in the narrower sense 'IS(N)'. In this paper, whenever we use the term information systems, we mean information systems in the narrower sense. Conforming with our systems view on organizations and information systems, a general architecture for information systems is presented (see also Falkenberg *et al.* 1992a, c). On the basis of this architecture the distinction between traditional and evolving information systems is explained.

The information processor in an information system accepts input messages (requests), which, among other things, may reflect changes in 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 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 (Brinkkemper & Falkenberg, 1991).

As stated before, the meta model is application independent and time invariant. The application model, however, is application dependent, and can be time variant. As a result, 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 the 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 is based on the meta model of that particular information system. The architecture discussed is depicted in Fig. 2.

This figure shows that the information processor (actor) performs the information processing (activity), having requests, the meta model and the application model as input (operand), and having responses and the (updated) application model as output (operand). Requests are generated and responses are consumed by the universe of discourse (in the environment).

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) that 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 [e.g. entity relationship modelling (Chen, 1976), Natural Language for Information Analysis Method — NIAM (Nijssen & Halpin, 1990; Wintraecken, 1990) or Predicator Set Model — PSM (ter Hofstede *et al.*, 1992a; ter Hofstede & van der Weide, 1993)]

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

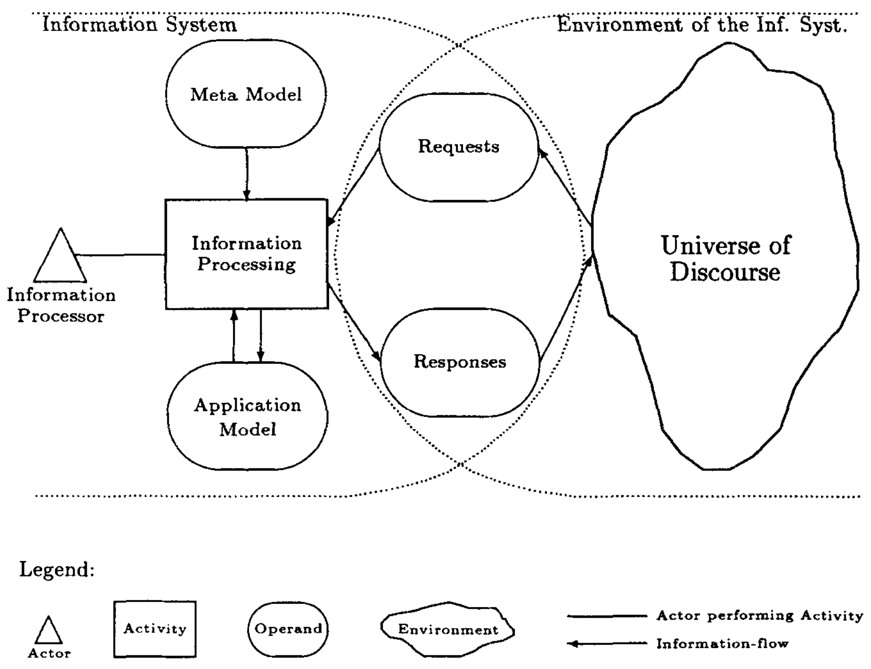


**Figure 2.** An (evolving) information system and its environment.

this latter part as 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 systems 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 model are data flow diagrams (Gane & Sarson, 1986) or the A-schemas in Information Systems work and Analysis of Changes (ISAC) (Lundeberg *et al.*, 1981). Petri-Nets (Genrich & Lautenbach, 1981), task structures (Wijers *et al.*, 1992; ter Hofstede & Nieuwland, 1993) and the WHEN-IF-THEN rules in REMORA (Rolland & Richard, 1982) are examples of techniques which are used for modelling behaviour models. The subdivision of the processing model is illustrated in Fig. 3. Note that there exist also (unified) modelling techniques, such as Telos (Jarke *et al.*, 1992) and Transaction Modelling Technique (TMT) (ter Hofstede, 1993), which intend to specify both world, activity and behaviour model. Furthermore, meta models and application models can be specified either in one and the same modelling technique (language) or in different modelling techniques (as will be shown in Fig. 4).

*Example 2.1*

*To illustrate the subdivision of processing models, a possible subdivision of the processing model*

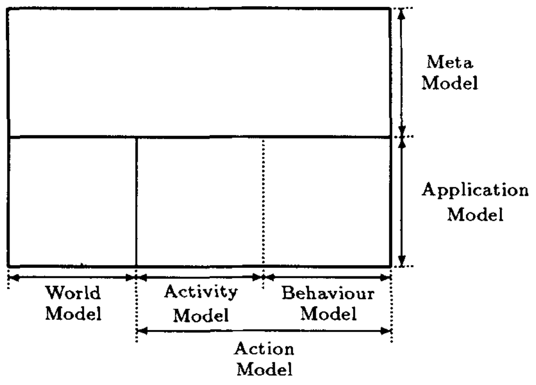


**Figure 3.** The structure of the processing model.

*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 size of this scholarship depends whether students live on their own or with their parents, the parents' income and the amount of their extra earnings. In Fig. 4 three different modelling techniques are used to describe 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-schemas of ISAC (Lundeberg* et al., *1981), whereas event decomposition diagrams of Yourdon (1989) are used for describing the behaviour model part. The language(s) used for specification of the application model is based on the meta model of the information system. Although we can use the same techniques for describing its meta model, in Fig. 4 the (partial) meta model is described in another modelling technique, namely NIAM (Nijssen & Halpin, 1989). Note, however, that the specifications of the constituent parts of the processing model are incomplete. For example, the instantiations of the conceptual schema at a particular point of time are omitted from the application model. This example is simply intended to be illustrative.*

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 (type) versus instance dichotomy (e.g. Brinkkemper & Falkenberg, 1991) 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 programming procedures), cannot be updated in traditional information systems. Note that the schema versus instance dichotomy is also applicable to the processing model as a whole. The relationship between the meta model and the application model is that a particular application model is an instantiation of the meta model of the technique in which the application model is specified.

The intention of an evolving information system, however, is that the complete application model will become updatable. As can be seen in Fig. 3, this application model is composed of the world model (including both schema and instance specification), activity model and behaviour model. In our scholarship system example, the need to update all specifications in the

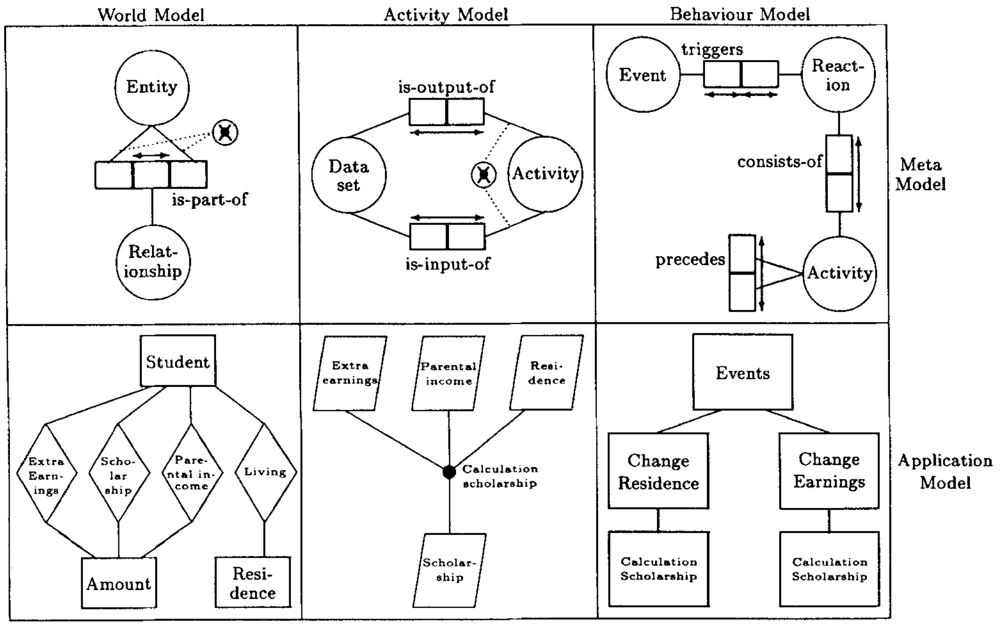


**Figure 4.** The processing model for the Dutch scholarship information system.

application model is apparent, because the laws of the scholarship system in the Netherlands appear to change frequently. For example, because of subsequent cuts in the total budget, maximum study time has been limited over years.

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). When an evolving information system has to be developed for a particular universe of discourse, an application model describing this domain is built up and conforms with the language defined in the (meta model of the) EIS shell.

The EIS shell is independent of any universe of discourse. 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 management system and/or operating system. This is illustrated in Fig. 5.

## 3 UPDATE IN EVOLVING INFORMATION SYSTEMS

In Falkenberg *et al.* (1992a, c) a conceptual framework for update in evolving information systems was introduced. This framework has been formalized by Falkenberg *et al.* (1992b) and

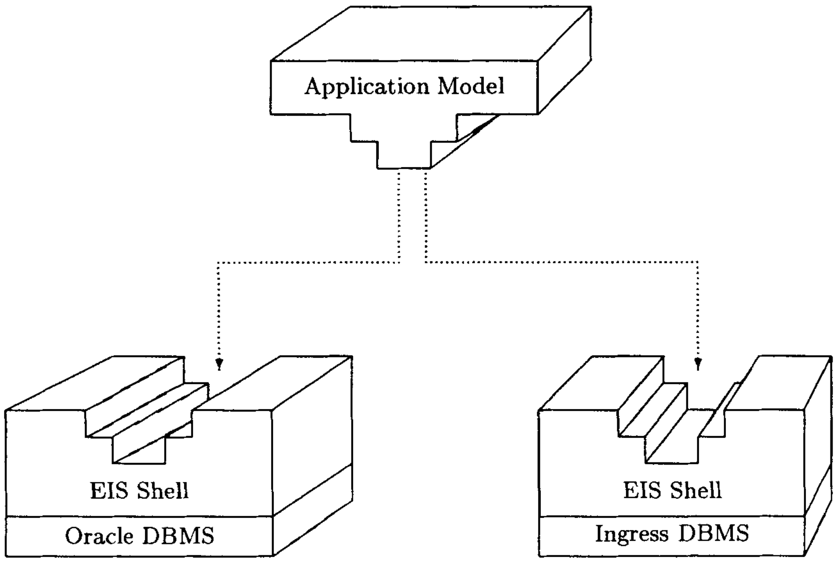


**Figure 5.** The EIS shell: independent of any application model and software environment.

Oei *et al.* (1992a). 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, namely addition, deletion and modification, is replaced by an evolutionary one that is based on the possible reasons for update requests rather than internal database operations. Three kinds of updates are distinguished, namely recording, correction and forgetting. Recording of an event is the processing of an update request caused by a change in the state of the universe of discourse. Update requests are formulated in a language that 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 because incorrect or incomplete information is available to the users. Incorrectness of these kinds can only be detected by empirical validation. An information system reflects an organization correctly if and only if there exists an isomorphism between the states in the information system and the states in the organization system being modelled. The order in which the events occurred in the organization has to be preserved by this mapping (Falkenberg *et al.*, 1992a). 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. State transitions on the event level take place as a result of events occurring in the organization, 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, for example, ter Hoftede & van der Weide, 1993). The states and state transitions in a universe of discourse are modelled in an information system. The state of an organization 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 constitutes the application model state. Note that the types of application model elements (e.g. objects, entities, relationships, activities, events, triggers, etc.) are dependent on the modelling technique used.

A state transition in the organization 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 organization.

A transition in the organization taking place at a particular point of time is called an (organizational) event. The point of time at which such an event occurs in the organization is called the event time of that event. These events are considered to occur on the organizational level (Falkenberg *et al.*, 1992a). 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 Fig. 6).

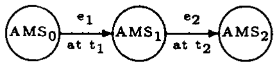


**Figure 6.** Application model state (AMS) transitions at the event level.

*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 (ter Hostede* et al., *(1992b) 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 Fl 300, whereas you receive Fl 400 when you live on your own. Another rule states that the scholarship is cut whenever you have some extra earnings exceeding a certain amount (200). This can be specified in the behaviour model as follows:*

```
● WHEN BIRTHS (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-earning Amount y)
  If y > 200 AND Student s getting-scholarship Amount z
  THEN Reduce-Scholarship(s,y,z)
```

*Activity model. The way in which the scholarship is reduced in case of extra earnings exceeding FL 200 is specified in the activity model as follows:*

```
● ACTIVITY Reduce Scholarship (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 we know that two events affecting Jim's scholarship have occurred. The first event is that Jim started living on his own on 1 January 1990. The second event says that 1 year later Jim found a job providing him with extra earnings of FL 300. The application model history caused by these events can be derived from Fig. 6 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
● AMS₀={initial state}
● AMS₁=AMS₀ U FACT ('Jim' living-at 'Independent')
            U FACT('Jim' getting-scholarship 400)
● AMS₂=AMS₁ U Fact ('Jim' having-earnings 300)
            - FACT('Jim getting-scholarship 400)
            U FACT ('Jim' getting-scholarship 325)
```

*Note that Jim's scholarship has been reduced from 400 (living independently) to 325* [400 – 0.75 × (300 – 200)] *because of his extra earnings. The substitutions in Fig. 6 for our example result in Fig. 7.*

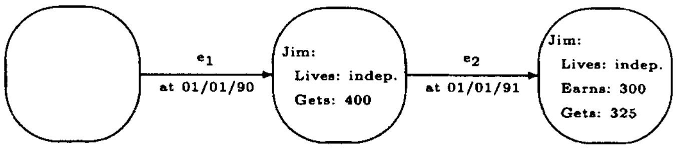


**Figure 7.** Application model history for our example.

## 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 organization, 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 organization up to the occurrence 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 organization and the recordings of these events in the information system. In Fig. 8, the graphical representation of an application model recording history is given.

*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.*

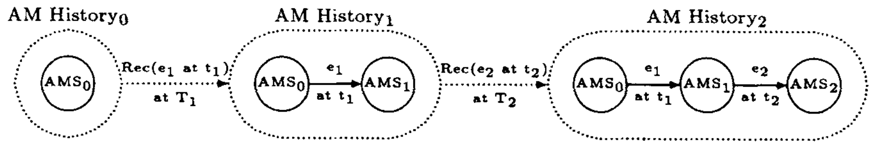


**Figure 8.** Application model history (AMH) transitions at the recording level.

- **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 Fig. 8. This results in an application model recording history for our example which is represented in Fig. 9.*

### 3.3 The correction level

In the process of recording events, mistakes can be made. Validation may reveal that information about events in the organization 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, i.e. the mapping between organization system and information system appears to be non-isomorphic, a roll-back should take place to the latest

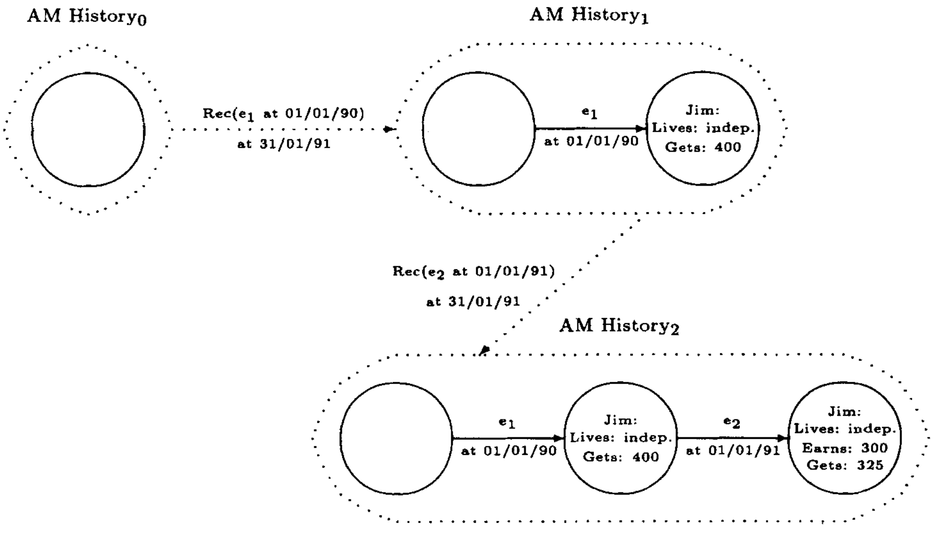


**Figure 9.** Application model recording history for our example.

application model history which is correct. 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 rerecorded. 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 Fig. 10, 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 view of the world (organization) 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 Falkenberg *et al.* (1992a) more examples are given and elaborated.

*Example 3.3*

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

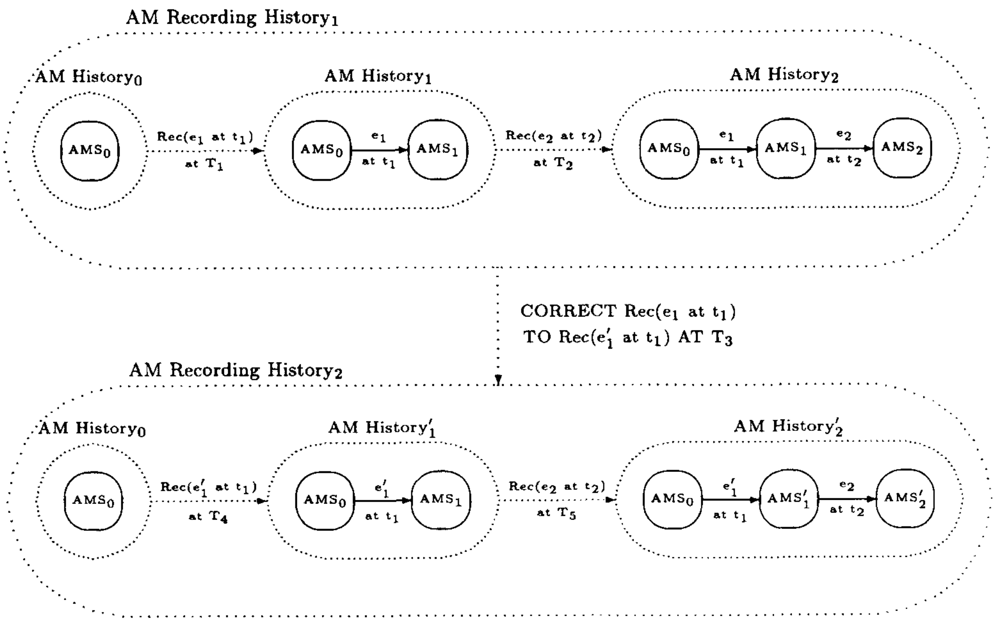


**Figure 10.** Application model recording history (AMRH) transition at the correction level.

- `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 Fig. 10 by extending the substitutions of the previous figures with:*

- `T3=T4=T5=31/01/92 and`
- $AMS_1'=AMS_0$ `U FACT ('Jim' living-at 'Parents')`
  `U FACT ('Jim' getting-scholarship 300)`
- $AMS_2'=AMS_1'$ `U FACT('Jim' having-earnings 300)`
  `- FACT('Jim' getting-scholarship 300)`
  `U 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 rerecordings of events afterwards. It is insufficient just to correct the latest state. If 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 Jim would still have received a scholarship of FL 325 instead of the correct FL 225. The resulting application model evolution for our running example is represented in Fig. 11.*

This concludes the explanation of the conceptual framework for update in evolving information systems. It should be noted that the complete framework has been formalized (Falkenberg *et al.*, 1992b; Oei *et al.*, 1992a). On the basis of this formalization a prototype of a generalized EIS shell is being implemented.

## 4. SCHEMA EVOLUTION IN EVOLVING INFORMATION SYSTEMS

The example provided in the previous section involved the recording and correction of recordings of events on the instance level of (the world model) of the application model. 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, i.e. 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 in both changes the schema and the action rules.

Consider a rental store for audio records (LPs). In this store a registration is maintained of the songs that are recorded on the available LPs. In order to keep track of the wear and tear of LPs, the number of times an LP has been lent is registered. The schema (or information structure) and constraints of this universe of discourse are modelled in Fig. 12 in the style of Entity Relationship (ER) modelling, according to the conventions of Yourdon (1989). Note the special notation of attributes (`Title`) using a mark symbol (`#`) followed by the attribute (`# Title`).

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, its lending frequency must be set to 0:

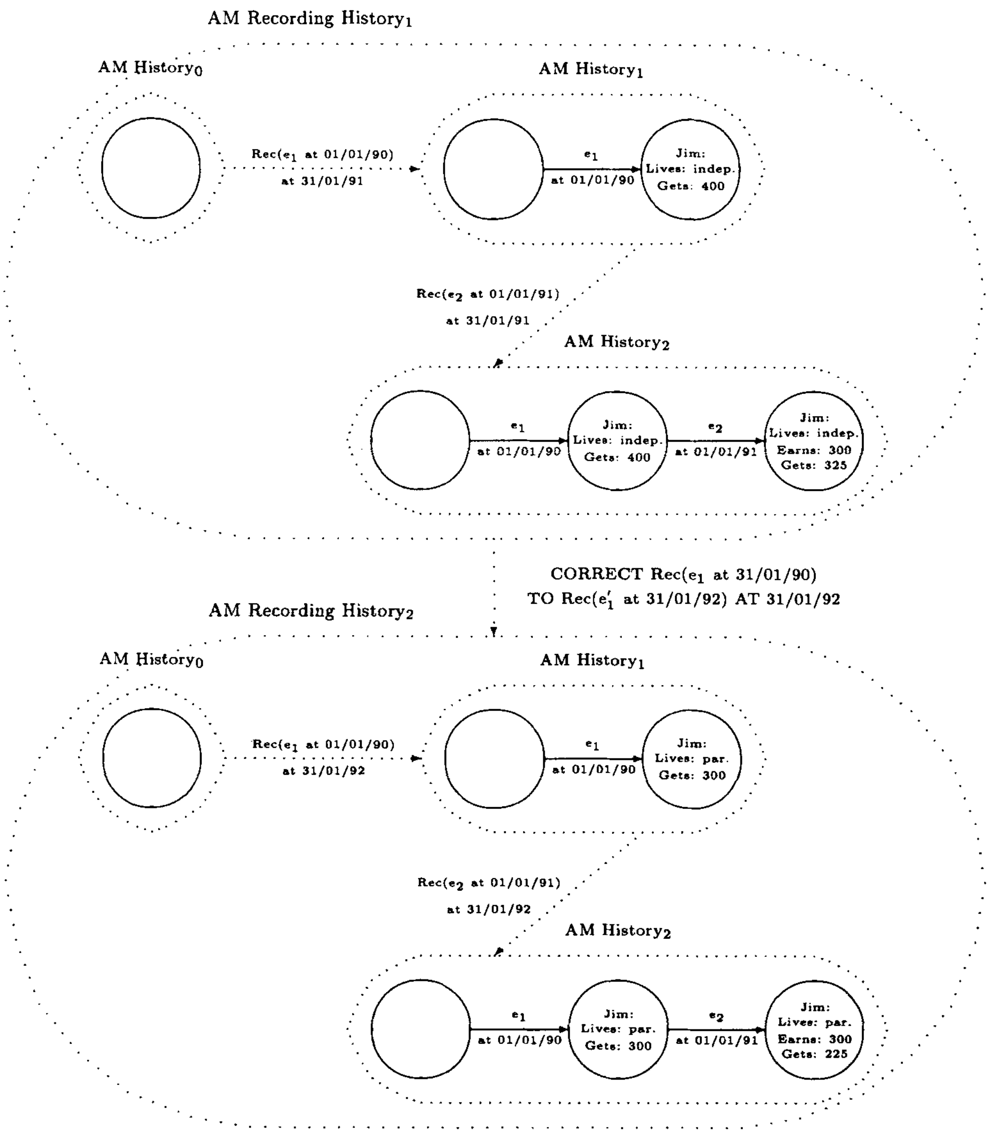

AM Recording History1
AM History0
AM History1
Rec(e1 at 01/01/90) at 31/01/91
e1
at 01/01/90
Jim: Lives: indep. Gets: 400
Rec(e2 at 01/01/91) at 31/01/91
AM History2
e1
at 01/01/90
Jim: Lives: indep. Gets: 400
e2
at 01/01/91
Jim: Lives: indep. Earns: 300 Gets: 325
CORRECT Rec(e1 at 31/01/90) TO Rec(e'1 at 31/01/92) AT 31/01/92
AM Recording History2
AM History0
AM History1
Rec(e1 at 01/01/90) at 31/01/92
e1
at 01/01/90
Jim: Lives: par. Gets: 300
Rec(e2 at 01/01/91) at 31/01/92
AM History2
e1
at 01/01/90
Jim: Lives: par. Gets: 300
e2
at 01/01/91
Jim: Lives: par. Earns: 300 Gets: 225

**Figure 11.** Application model evolution for our example.

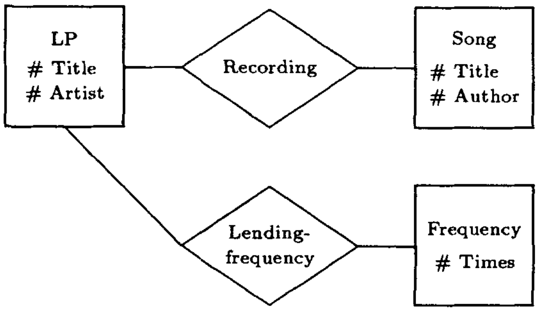


**Figure 12.** The information structure of an LP rental store.

- **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 been 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 LPs is extended to cover CD as well. The frequency of lending, however, is not kept for CDs, as CDs are hardly subject to any wear and tear. As a consequence, the application model has evolved to Fig. 13.

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

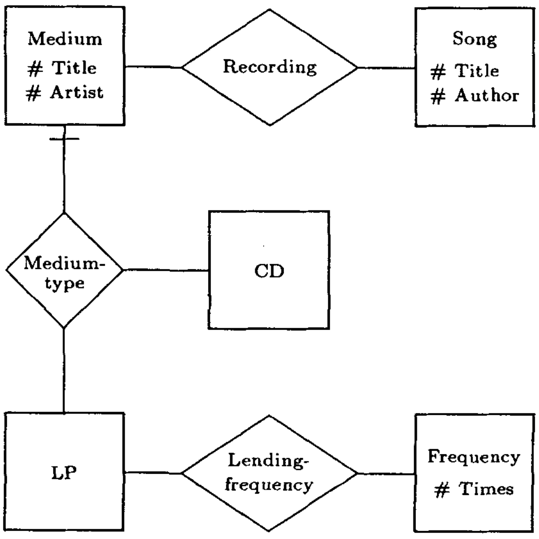


**Figure 13.** The information structure of a LP and CD rental store.

- `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 CDs have become more popular than LPs. Consequently, the rental store has decided to stop renting LPs and to become a CD rental store. This change in the rental store leads to the information structure as depicted in Fig. 14. As a result of this evolution step, the action specification `Init-freq` can be deleted, since the lending frequency of CDs is no longer recorded.

The three ER schemata and the associated action specifications, as discussed above, correspond to three distinct snapshots of an evolving universe of discourse.

In our framework for update, updates of the schema (e.g. the birth of the CD and the death of the LP) and updates of the action specification are managed in exactly the same way as updates of events on the instance level were managed in the previous section. Update requests are accepted or rejected by the information processor on the basis of the rules being specified in the meta model, i.e. the rules for preserving consistency between schema and instances are also part of the meta model. It should be noted that this issue of consistency is treated independently from the evolution management discussed in the framework for update.

Several approaches can be adopted to guarantee consistency between schema and instances. We consider an application model to be composed of distinct application model elements (such as, for example, object types, instances, constraints). The specific application model elements, and the relationships between them, depend on the chosen modelling technique. As a consequence, the implications of evolution of application model elements belonging to the schema also depend on the chosen modelling technique. In Veenstra *et al.* (1991), for example, the detection and correction of population conflicts with scheme evolution in object-role models, such as NIAM, is considered.

In Proper & van der Weide (1993), a more technique-independent approach is adopted. Instead of maintaining the evolution of the application model as a whole (as in most approaches for scheme versioning), it is proposed to maintain the evolution of each distinct application model element, thus keeping track of the evolution of individual object types, instances, constraints, etc. A (snapshot) version of the application model as a whole can be derived from the (current) versions of its component application model elements. The approach being proposed enables one to state well-formedness rules about the evolution of distinct application model elements which are more or less independent of the chosen modelling technique.

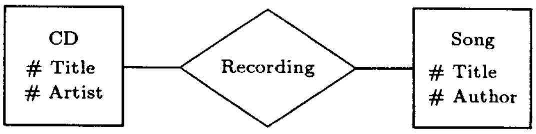


**Figure 14.** The information structure of a CD rental store.

## 5 CONCLUSION

To meet the demands of organizations 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 into an application-independent and time-invariant part, the meta model, and a part that 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, i.e. the complete application model, without the need to interrupt the processes in the organization.

In order to handle temporal and evolutionary aspects in an evolving information system, we revised the traditional notion of update, resulting in the three-element system: 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. It is claimed that the framework for update is applicable to a change in any part of the application model. Preservation of consistency in the case of schema evolution is considered as an additional, but separate, problem.

The conceptual framework proposed in this paper is the basis of a metal 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 (Falkenberg *et al.*, 1992b; Proper & van der Weide, 1993) 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 being 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

Ariav, G. (1991) Temporally oriented data definitions: managing scheme evolution in temporally oriented databases. *Data & Knowledge Engineering*, **6**, 451–67. (1987)

Banerjee, H.-T., Chou, J.F., Garza, W., Kim, D. & Ballou, N. Data model issues for object-oriented applications. *ACM Transactions on Office Information Systems*, **5**, 3–26.

Brinkkemper, S. & Falkenberg, E.D. (1991) Three dichotomies in the information system methodology. In: Bots, P.W.G., Sol, H.G. and Sprinkhuizen-Kuyper, I.G. (eds) *Informatiesystemen in beweging, pp.75–87.* Kluwer, Deventer.

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

Falkenberg, E.D., Oei, J.L.H. & Proper, H.A. (1992a) A conceptual framework for evolving information systems. In: Sol, H.G. and Crosslin, R.L. (eds) pp. 353–375. *Dynamic Modelling of Information Systems II.* North-Holland, Amsterdam. 1991.

Falkenberg, E.D., Oei, J.L.H. & Proper H.A. (1992b) *A Metamodel for Update in Information Systems.* Technical Report 92-05. Department of Information Systems, University of Nijmegen, Nijmegen.

Falkenberg, E.D., Oei, J.L.H. & Proper, H.A. (1992c) Evolving information systems: beyond temporal information systems. In: Tjoa, A.M. & Ramos, I. (eds) pp. 282–287. *Proceedings of the Data Base and Expert System Applications Conference (DEXA 92), Valencia, Spain, September 1992.* Springer-Verlag, Berlin.

Gane, C. & Sarson, T. (1986) *Structured System Analysis: Tools and Techniques.* IST Databooks. MacDonald Douglas Corporation, St Louis.

Genrich, H.J. & Lautenbach, K. (1981) System modelling with high-level Petri-Nets. *Theoretical Computer Science*, **13**, 109–136.

ter Hofstede, A.H.M. (1993) *Information Modelling in Data Intensive Domains.* PhD thesis. University of Nijmegen, Nijmegen.

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

ter Hofstede, A.H.M., Proper, H.A. & van der Weide, Th.P. (1992a) Data modelling in complex application domains. In: *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering, Manchester, United Kingdom, May 1992*, Loucopoulos, P. (ed.) *Lecture Notes in Computer Science*, **S93**, 364–377.

ter Hofstede, A.H.M., Proper, H.A. & van der Weide, Th.P. (1992b) *Formal Definition of a Conceptual Language for the Description and Manipulation of Information Models.* Technical Report 92/10. Software Engineering Research Centre, Utrecht.

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

Jarke, M., Mylopoulos, J., Schmit, J.W. & Vassiliou, Y. (1992) DAIDA: an environment for evolving information systems. *ACM Transactions on Information Systems*, **20(1)**; 1–50.

Katz, R.H. (1990) Toward a unified framework for version modelling in engineering databases. *ACM Computing Surveys*, **22**, 375–408.

Lundeberg, M., Goldkuhl, G. & Nilsson, A. (1981) *Information Systems Development — A Systematic Approach.* Prentice-Hall, Englewood Cliffs, NJ.

McKenzie, E. & Snodgrass, R. (1990) Scheme evolution and the relational algebra. *Information Systems*, **15**, 207–232.

Nijssen, G.M. & Halpin, T.A. (1989) *Conceptual Schema and Relational Database Design: a Fact Oriented Approach.* Prentice-Hall, Sydney.

Oei, J.L.H. & Falkenberg, E.D. (1994) *Harmonisation of Information System Modelling and Specification Techniques.* Memoranda Informatica 94–07. University of Twente, The Netherlands. In: Proceedings of the IFIP W68.1. CRIS-94 Conference on 'Methods and Associated Tools for the Information System Life Cycle', Maastricht, The Netherlands, September 1994.

Oei, J.L.H. & Falkenberg, E.D. (1992a) *Modelling the Evolution of Information Systems.* Technical Report 92–36. Department of Information Systems, University of Nijmegen, Nijmegen.

Oei, J.L.H., van Hemmen, L.J.G.T., Falkenberg, E.D. & Brinkkemper, S. (1992b) *The Metal Model Hierarchy: A Framework for Information System Concepts and Techniques.* Technical Report 92-17. Department of Information Systems, University of Nijmegen, Nijmegen.

Proper, H.A. & van der Weide, Th.P. (1993) Towards a general theory for the evolution of application models. In: *Proceedings of the Fourth Australian Database Conference, Brisbane, Australia, February 1993*, Orlavska, M.E. and Papazoglou, M. (eds). *Advances in Database Research*, 346–362.

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

Rolland, C. & Richard, C. (1982) The REMORA methodology for information system design and management. In: Olle, T.W., Sol, H.G. and Verrijn-Stuart, A.A. (eds) *Information Systems Design Methodologies: A Comparative Review*, pp. 369–426. North-Holland/IFIP, Amsterdam.

Snodgrass, R. & Ahn, I. (1986) Temporal databases. *IEEE Computer*, **19(9)**, 35–42.

Veenstra, B.M.J.M., Oei, J.L.H. & van Smaalen, W. (1991) Detectie en correctie van populatatieconflicten bij schemawijzigingen (Detection and correction of population conflicts with schema changes). *Journal of Software Research*, **3(4)**, 48–55.

Verrijn-Stuart, A.A. (1989) Some reflections on the Namur conference on information systems concepts. In: Falkenberg, E.D. and Lindgreen, P. (eds) *Information System Concepts: An In-depth Analysis*, pp. ix–x. North-Holland/IFIP, Amsterdam.

Visschedijk, Th.H. & van der Werff, R.N. (1991) (R)evolutionary system development in practice. *Journal of Software Research*, December (special issue), 46–57.

Wijers, G.M., ter Hofstede, A.H.M. & van Oosterom, N.E. (1992) Representation of information modelling knowledge. In: Tahvanainen, V.-P. and Lyytinen, K. (eds), pp. 167–223. *Next Generation CASE Tools* Vol. 3 of *Studies in Computer and Communication Systems*. IOS Press, Trondheim, Norway.

Wintraecken, J.J.V.R. (1990) *The NIAM Information Analysis Method: Theory and Practice*. Kluwer, Deventer.

Yourdon, E. (1989) *Modern Structured Analysis*. Prentice-Hall, Englewood Cliffs, NJ.

## Biographies

Han Oei is an assistant professor in the subdepartment of Information Systems at the University of Twente, The Netherlands. From 1990 to 1993, after receiving his master degree in Computer Science (major in Business Informatics) from the University of Groningen, he has been involved in the evolving information systems project as an assistant researcher at the University of Nijmegen. He is a member of the IFIP WG 8.1 Task Group FRISCO investigating the conceptual foundations of information systems. His current research interests include (evolving) information system methods, techniques and concepts, meta modelling and (situational) method engineering.

Erik Proper received his master degree in Computer Science from the University of Nijmegen, The Netherlands, in 1990. He is currently a PhD student at the University of Nijmegen, The Netherlands, and expects to receive his PhD before the summer of 1994. His main research interests include (evolving) information systems, information retrieval, hypertext and knowledge-based systems.

Eckhard D. Falkenberg is best known for his pioneering work on information modelling. His major research projects have been on conceptual schemata, three-level database systems and information system design methodology. He has been affiliated with the University of Stuttgart, the Siemens Research Laboratories in Munich and the University of Queensland. Since 1986, he has been a Full Professor of Informatics at the University of Nijmegen (Netherlands). Professor Dr E.D. Falkenberg is a member of IFIP working groups on databases and information systems. He is chairman of the IFIP WG 8.1 Task Group FRISCO investigating the conceptual foundations of information systems.