Evolving Information Systems: Beyond Temporal Information Systems*

E.D. Falkenberg, J.L.H. Oei, H.A. Proper
University of Nijmegen,
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
E.Proper@acm.org

PUBLISHED AS:

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

Abstract

Nowadays, in order for an organisation to be competitive, it must be able to adapt quickly to its dynamic environment. In this paper, we discuss the need for information systems which are capable to evolve to the same extent as organisations do. Requirements of evolving organisations on their information systems are identified, followed by alternative approaches to adequate information systems development life cycles. We adopt an evolutionary approach resulting in so-called evolving information systems.

On the basis of requirements and an architecture for these evolving information systems, the distinction from traditional information systems is explained. Traditional information systems, including temporal information systems, appear to be degenerations of our evolving information systems. A conceptual framework for update in evolving information systems is derived from the requirements. An event level, a recording level and a correction level are distinguished in this framework for update.

1 Requirements of Evolving Organisations on their Information Systems

Nowadays, in order for an organisation to be competitive on the global market place, it must be flexible. The organisation must be able to adapt itself quickly to the production of new or different products - changes in the primary process of an organisation - and to the ever changing and more and more demanding consumer needs.

Due to the dynamic behaviour of organisations, these 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. In this context, it is interesting to note that in the current situation already 42% of all maintenance of information systems is needed for extensions/changes of the information system due to changes of the information needs ([Bem87]). In case of a highly dynamic organisation, this latter percentage is likely to be even higher. (Note that our notion of information systems in this paper is that of information systems in the narrower sense as explained in section 4 and in [Ver89]).

Realizing this situation, it can be concluded that information systems are needed which can be adapted to the changing environment in smaller, easier, and more frequent steps than generally it is the case. In

*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).

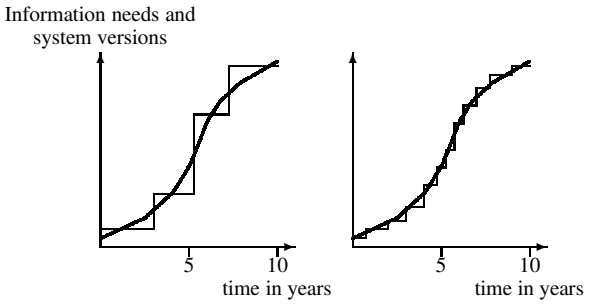

Figure 1: The discrepancy between information needs and systems versions

[Bem87], the adaptability of (information) systems is termed flexibility in a broader sense, and the ability of a (information) system to continue to function in a satisfactory way without having to change the information system, although the organisation has changed, is termed flexibility in a narrower sense.

With respect to flexibility in a broader sense consider figure 1, which is taken from [Bem87]. In this figure the dotted lines represent the evolution of the information system, whereas the solid lines represent the evolution of the information needs. From the figure it can be concluded that the bigger the time periods between adjustments of the information system are, the bigger the discrepancy is between the information needs and the information available from the information system. Such a discrepancy can result in an undesirable situation in which users will start to dislike using the information system to fulfill their information needs. Consequently, a rather expensive production factor - the information contained in the information systems - may become idle.

In order to cope with the problems identified, it is concluded that information systems are needed which are more flexible, both in the broader and in the narrower sense, than the current generation of information systems are. These information systems which are able to evolve to the same extent and at the same pace as their underlying organisations do, are called evolving information systems.

2 Information Systems Development in Evolving Organisations

If organisations change fast and frequently, the need emerges for an appropriate strategy which allows information systems to evolve to the same extent and at the same pace as these organisation do. An attempt to accomplish this goal would be to try to shorten or even abolish the traditional life cycle process of information system development as illustrated in figure 2, which is taken from [Som89].

In [Lan87], six approaches to shorten or abolish this expensive life cycle process of information system development have been identified. Following these approaches, the concept of reusability and an evolutionary approach towards information systems development and maintenance is discussed here in order to understand the philosophy behind evolving information systems.

The information system development life cycle can be made less expensive by designing information systems in such a way that parts of the system, and the system specifications are suitable for reuse. The

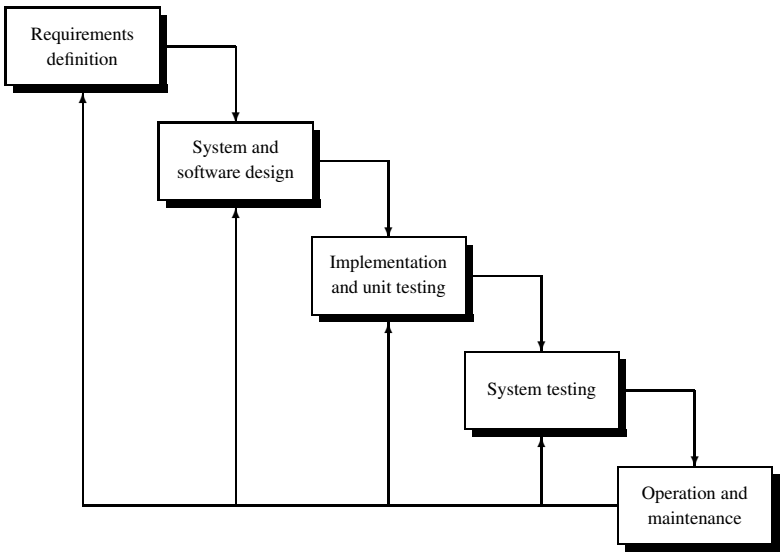

Figure 2: Traditional information systems development life cycle

development of CASE-tools is an example of an attempt to follow this strategy. Products developed in a CASE-tool during the analysis and design phase of an information system (eg schemas, repositories, etc.) can serve as a starting point for the analysis and design phase of a new information system to be developed ([McC89], [OHM$^+$88]). In the context of software engineering, the (re-)use of software modules ([Som89]) illustrates the strategy of reusability. The reusability of (software) objects is also the major claim of the object-oriented approach (eg [Mey88]).

Most approaches to information systems development, including the ones which support the reuse of products, demand for an actual replacement of the information system by another information system in case of a structural change. Processes in the organisation depending on information from the information system to be replaced, may have to be interrupted. The evolutionary approach that is proposed here, however, can be distinguished from traditional ones in the sense that it is required to maintain the information system without the need to interrupt processes in the organisation. This evolutionary approach can be regarded as an advanced kind of prototyping. Prototyping resulted from the observation that in a significant percentage of cases, as is argued in [Dav87], requirements for the systems can not be established correctly and completely in advance. Even information systems which are based on correctly elicitated requirements may be rejected by users, or require substantial rework to make them fit the actual needs of the user. These needs may have changed since the start of the (traditional) life cycle of the information system in question.

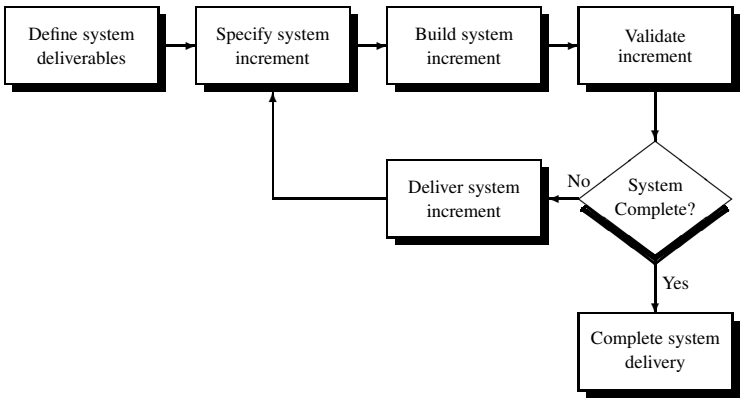

Figure 3: Prototyping

Prototyping has been introduced as an alternative approach to the strict incremental development process, used in the traditional life cycle. After the acquisition of an initial set of requirements, a first prototype is built on the basis of this set of requirements. After verification and validation by user and designer a new prototype is built, after which this process is repeated until the point is reached that the user of the information system is satisfied. The final prototype resulting from this iterative process will be the information system which becomes part of the organisation. The prototyping process is illustrated in figure 3 which is taken from [Som89].

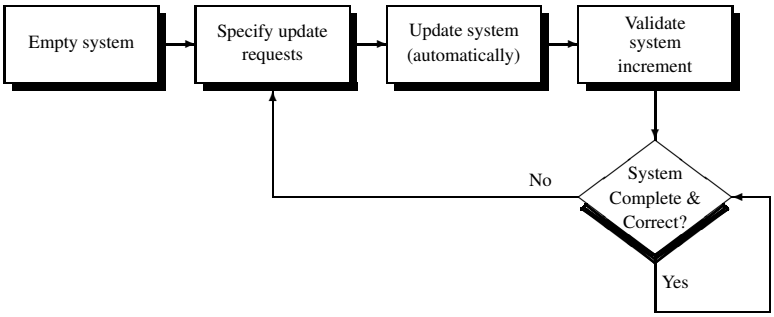

Figure 4: Evolutionary approach

In our approach to evolving information systems, the prototyping strategy is not restricted to the development phase of an information system, but is also followed in the operation and maintenance phase. Our approach is an evolutionary approach characterized by an iterative life cycle having the length of the organisation's existence. In this evolutionary approach there is no essential difference between the development and maintenance phase of information systems. The evolutionary approach which we advocate here, distinguishes itself from others by the demand that adjustments of the information system have to be made without the need to interrupt any processes in the organisation.

3 Requirements for Evolving Information Systems

As the title of this paper suggests, the requirements for these evolving information systems are beyond those of traditional, including temporal, information systems. The main requirement for an evolving information system is that, as stated before, it is able to evolve to the same extent and at the same pace as the underlying organisations does, without the need to interrupt the processes of that organisation. The notion of 'to the same extent' and 'at the same pace' is now refined in more detail.

1. The information system allows update of all information which is dependent on the specific organisation domain (universe of discourse) of the information system. Our notion of update, including recording, correction and forgetting, is discussed in section 5. The specification of information which is dependent on a specific organisation domain is part of the architecture for evolving information systems as is discussed in section 4.

2. The information system allows correction of all information (previously) recorded in the system. Information having 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 5. 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 organisation. The intention of evolving information systems is to minimize the discrepancy between the information needs of the organisation and the information supply by the information system. 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. Even more, at least two distinct notions of time have to be distinguished. It will be obvious that for meeting requirement 3 events in the organisation have to be recorded together with their time of occurence. The point of time at which an event occurs in the organisation is called the event time of that event. To perform corrections a roll-back operator is needed (see section 5). 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, as discussed in [SA86]. (The reason for this renaming is that the new names correspond better to the level architecture we will introduce in section 6.) The classification 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.

4 The Architecture for Evolving Information Systems

In our systems view on organisations, which is conform the approach taken in [FHL+98], an organisation system is considered to be a set of interrelated actors, activities, states and points of time. Actors perform certain activities resulting in a change of state at a certain point of time. The information systems considered here, 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 decomposed out of several sub-processors, which may also be (physically) distributed. In this paper, however, the specific aspects of distributed information systems are not taken in consideration.

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

The information processor in an information system accepts input messages (requests), which, among other things, 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 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 ([BF91]).

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

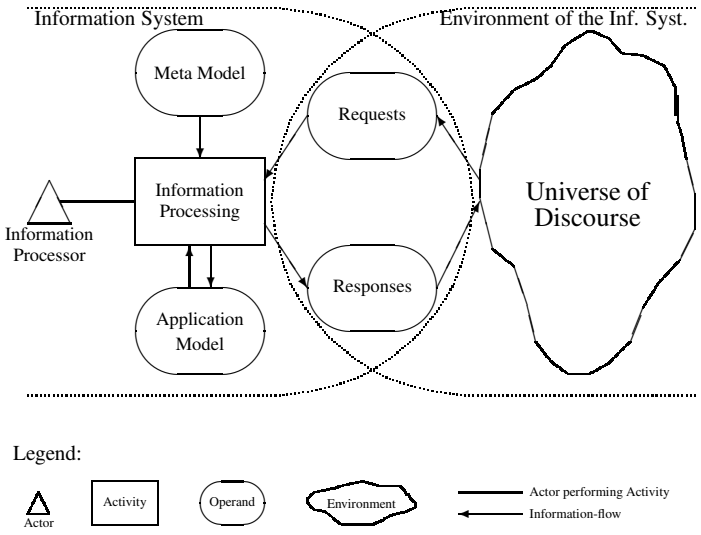

Figure 5: An (Evolving) Information System and its Environment

processor, which acts on, or reacts to events in the universe of discourse 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 5.

An application model can be subdivided (see figure 6). On the one hand, we need a model of that part of the perceived world (universe of discourse) the interaction between the information system and the environment is about. This model is called the world model. A world model can be described in a modelling technique like eg ER ([Che76]), NIAM ([Win90]) or the Predicator Set Model ([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 by the name behaviour model. In the behaviour model, for example, the relationship between events in the universe of discourse and the activities performed by the information processor in the information system is described. In other words, the behaviour model contains the description of *when* activities, under which conditions, and *what* activities should be performed by the information processor, whereas the activity model specifies *how* these activities should be performed. Examples of modelling techniques for the activity model are Data Flow Diagrams ([GS86]) or the A-schemas in ISAC ([LGN81]). Petri-Nets ([GL81]) and the WHEN-IF-THEN rules in REMORA ([RR82]) are examples of techniques which are used for modelling behaviour models.

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

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 maintained conform the language defined in the (meta model of the) EIS-shell. The fact that the EIS-shell is independent of any universe of discourse, and that application models de-

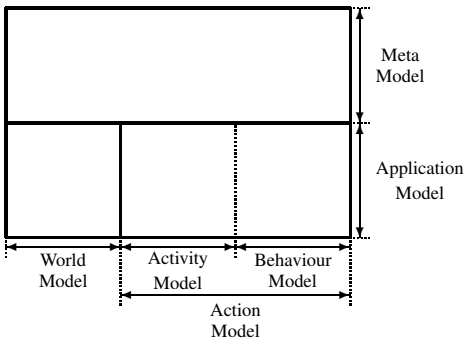

Figure 6: The Structure of the Processing Model

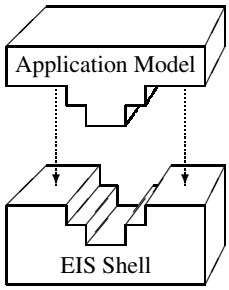

Figure 7: The EIS-shell: independent of any application model

scribing different domains can be 'plugged' into the EIS-shell is illustrated in figure 7.

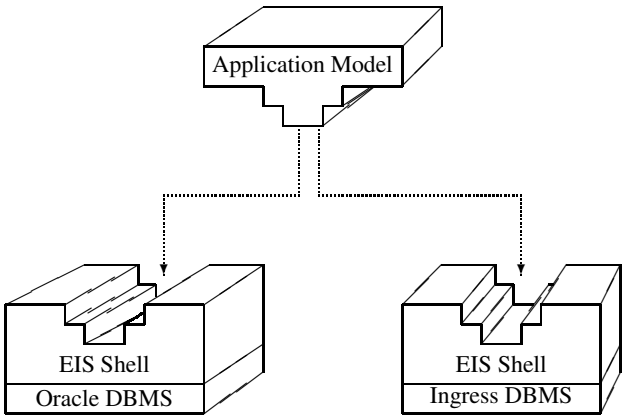

Figure 8: The EIS-shell: independent of any software environment

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. This is illustrated in figure 8.

5 Update in Evolving Information Systems

Information systems are meant to fulfill the information needs of organisations. As both the information needs and the information itself change in time, information systems have to be updated from time to time. In this section the notion of update is discussed. Update in traditional information systems is opposed to update in evolving information systems. It is concluded that update in traditional systems is a degeneration of update in evolving systems.

5.1 Update in traditional information systems

Update should result in a change of state of the information system such that the new state hopefully reflects the new state of the organisation in a satisfactory way. By validation and verification it has to be checked whether the information in the system is both (empiricaly) valid and (logically) consistent. In traditional information systems an elementary update is usually considered to be an addition, deletion or modification of (pieces of) information.

Furthermore, traditional information systems do not support any notion of time. As a consequence, if there is a change of state in the information system due to an update, the former state cannot be 'remembered' by the information system. Such systems are refered to as snapshot systems ([SA86]).

Starting from a particular state, update in snapshot systems is performed by an addition, deletion or modification of (pieces of) information in that state, resulting in a new state without keeping track of the updates made, and consequently, forgetting the former state. In a traditional relational database system, for example, tuples - representing information - can be added, deleted or modified in the database. After the deletion or a modification of a tuple, the former tuple can no longer be retrieved.

5.2 Update in evolving information systems

In this section we revise the traditional notion of update (viz addition, deletion and modification) which is actually based on elementary operations on the database. Our framework for update, however, is rather based on the possible causes for update requests, and which follow from the requirements for (evolving) information systems. On this basis three kinds of update are distinguished: recording, correction and forgetting ([FOP92a]).

An ideal information system reflects the state of an organisation system correctly at any time. Consequently, whenever an event, i.e. a change of state at a particular point of time, occurs in the organisation system, an update is needed which should result in an appropriate change of state in the information system, such that a correct mapping between information system and organisation system is preserved. This kind of update is performed by, what is called, a recording of an event occurence. Another situation requiring an update is a situation in which it appears, by empirical validation, that information derivable from the recordings of events is incorrect. In such situation, an update should be performed which corrects the mapping between information system and organisation system. This kind of update is called a correction, which corrects recordings which already have been performed.

A third kind of update can be distinguished if information systems are required to be able to remove information if requested. A law stating that information about former personell of a company may not be stored for more than two years, is an example of a situation which requires a third kind of update. This kind of update is called forgetting, which will not be considered in this paper.

5.2.1 Recording Whenever an event is said to occur, it has to be communicated to the information system by means of an update request. This event is reflected in the information system by the processing of this update request. The processing of an update request due to an event is called the recording of that event.

The state of an organisation is reflected by a set of modelling constructs (e.g. entities, relationships, rules, etc.) in the information system. The modelling constructs which are application dependent and, time-variant, are part of the application model, and therefore termed application model elements. Independently of any modelling technique it is concluded that an update results in the modification of the set of application model elements constituting the state of the application model of an information system. The modification is performed by a set of (elementary) transitions.

A birth-transition creates an application model element, an application model element is terminated by a death-transition, whereas an application model element is transformed into another application model element by a change-transition. (The question whether a change transition is elementary or a composition of a death transition and a birth transition at the same point of time is a philosophical one, which will be omitted here.) The lifetime of an application model element should correspond to the validity of the information represented by that application model element.

In the metamodel of a traditional snapshot system, a birth transition of an application model element is realized by a (physical) addition, a death-transition by means of a (physical) deletion of an application model element. Consequently, former states which contain deleted application model elements cannot be

retrieved anymore. In the case of evolving information systems, however, no information may be lost, so no application model elements may be deleted. The history of application model elements is kept by storing transitions of application model elements together with the points of time at which these transitions take place.

5.2.2 Correction The actual state of an information system depends completely on the processing of update requests formulated by users of the system. These update requests should result in an information system reflecting the organisation it is modelling in a correct way. 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 representing the organisation domain (universe of discourse), see e.g. [FOP92b] and [HW93].

In [FOP92b] it is also argued that the order in which the events occured in the organisation should be preserved by the mapping. This requirement is needed because recordings of several events may interfere with eachother, such that a different order may result in a different state of the information system. In operational terms this requirement states that events should be recorded correctly in order of their occurence.

In practice, however, by empirical validation it may appear that by mistake or by incomplete knowledge users have recorded an event which should have been recorded before, or users may have recorded events which did not happen at all. A combination of these two situations may also occur, i.e. the recording of an event which did not happen at all has to be replaced by the recording of an event which indeed did take place.

In evolving information systems a correction of recordings is performed by: an insertion of a recording of an event in the sequence of already performed recordings, a removal of a recording already performed or a replacement of a recording of an event by a new recording of another event. To accomplish these corrections, it must be possible to go back in the sequence of performed recordings. The operation accomplishing this task, is called a roll-back. As shown in [OPF92], insertions, removals, and replacements of recordings can be mapped onto roll-backs and (re-)recordings.

In snapshot systems there is no support of time, and consequently, no order can be preserved. As a result, incorrectness because of recordings (additions, deletions and modifications in snapshot systems) can only be corrected by a new set of additions, deletions and modifications. The determination of an appropriate set can be difficult due to the interferences between the (past) recordings.

6 A Conceptual Framework for Update

Based on the notion of update as discussed in the previous section, a conceptual framework for update is presented. The framework distinguishes and relates different types of state transitions. Each type of state transition corresponds to a different level of abstraction in the context of update in evolving information systems. An event level, a recording level, and a correction level will be distinguished. 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, wheras corrections of previous recordings cause state transitions on the correction level.

6.1 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 an universe of discourse are modelled in an information system. This means that the state of an organisation at a particular point of time is modelled by a set of application model elements. This set of application model elements is called the application model state.

A state transition in the organisation is modelled in the information system by means of a transition of the application model state. Note that 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 ([FOP92b]). 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. In figure 9 a graphical representation of a sample application model history on the event level is given. The circles represent the application model states, wheras transitions between these application model states is represented by arrows. Furthermore, the arrows are labeled with the denotation of the event causing the transition, and the event time of that event.

Figure 9: Application Model State (AMS) transitions on the event level

6.2 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 10, an example application model recording history is provided. The arrows representing transitions between application model histories are labeled with the denotation of the recording of the event causing that transition, and the recording time of the event in question.

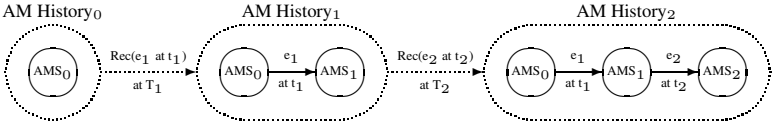

Figure 10: Application Model History (AMH) transitions on the recording level

6.3 The correction level

In the process of recording of events, mistakes can be made. Information about events in the organisation which have been recorded in the information system appears to be 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, 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 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 figure 11, the performance of a correction by means of a roll-back is represented in the case of a replacement of the recording of an event e_1 having event time t_1 by a recording of event e_1' with the same event time. An event e_2 which has an event time later than that of e_1 and e_1' has already been recorded.

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

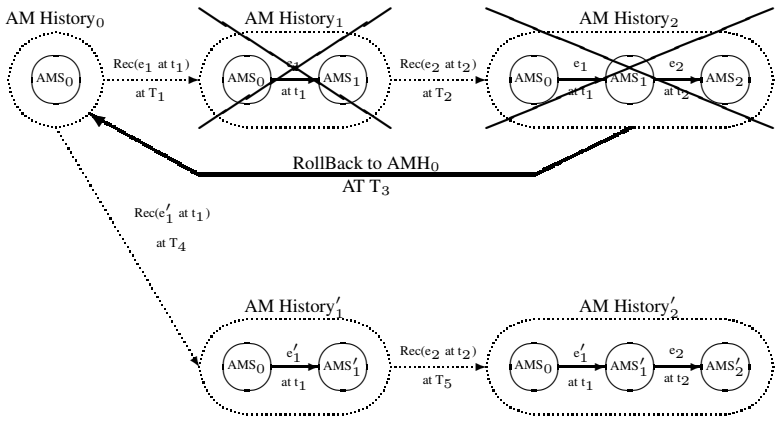

Figure 11: Correction by means of a roll-back

in the information system. A sequence of these application model recording history transitions due to roll-backs is called the application model evolution which is said to take place on the correction level. In figure 12 the situation of figure 11 is represented in an alternative way which identifies the three levels of state transitions more clearly. Note that the roll-back performed by the correction is implicitly present in this figure. In the same way corrections requiring the removal or insertion of a recording of an event can be represented. In [FOP92b] more examples are given and elaborated.

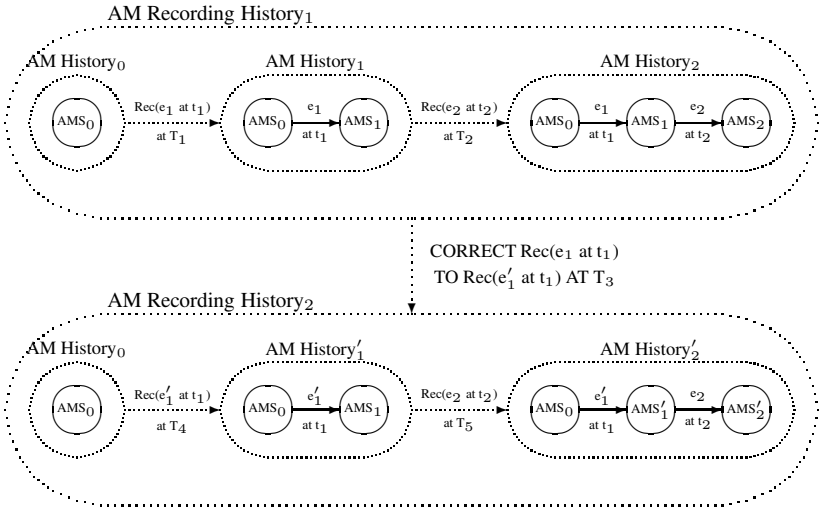

Figure 12: Application Model recording History (AMRH) transition on the correction level

7 Conclusions & Further Research

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

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

Further, the traditional notion of update (addition, deletion, and modification) has been replaced by an evolutionary one. With respect to update for evolving information systems, recording, correction, and for-

getting are distinguished. On the basis of the new notion of update, a conceptual framework was presented distinguishing state transitions on an event level, a recording level, and a correction level.

At the moment, a meta model is designed for evolving information systems ([FOP92c]). This meta model is based on the conceptual framework for update as discussed in this paper. Furthermore, an evolving information system shell is developed on the basis of that metamodel. Finally, a method for building up and maintaining application models in an EIS-shell will be designed. This method will be based on the evolutionary approach discussed.

References

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

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

[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.

[Dav87] G.B. Davis. Strategies for Information Requirements Determination. In Robert Galliers, editor, *Information Analysis: Selected readings*, chapter 13. Addison-Wesley, Reading, Massachusetts, 1987.

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

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

[FOP92b] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. A 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

[FOP92c] 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.

[GL81] H.J. Genrich and K. Lautenbach. System Modelling with High-Level Petri-Nets. *Theoretical Computer Science*, 13:109–136, 1981.

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

[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

[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.

[Lan87] F. Land. Adapting to Changing User Requirements. In Robert Galliers, editor, *Information Analysis: Selected readings*, chapter 12. Addison-Wesley, Reading, Massachusetts, 1987.

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

[McC89] C.L. McClure. *CASE is Software Automation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989. ISBN 0131193309

[Mey88] B. Meyer. *Object-oriented software construction*. Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[OHM+88] T.W. Olle, J. Hagelstein, I.G. Macdonald, C. Rolland, H.G. Sol, F.J.M. van Assche, and A.A. Verrijn-Stuart. *Information Systems Methodologies: A Framework for Understanding*. Addison-Wesley, Reading, Massachusetts, USA, 1988. ISBN 0-201-54443-1

[OPF92] J.L.H. Oei, H.A. Proper, and E.D. Falkenberg. Modelling the Evolution of Information Systems. Technical Report 92-36, Department of Information Systems, University of Nijmegen, Nijmegen, The Netherlands, EU, 1992.

[RR82] C. Rolland and C. Richard. The REMORA Methodology for Information System Design and Management. In T.W. Olle, H.G. Sol, and A.A. Verrijn-Stuart, editors, *Information Systems Design Methodologies: A Comparative Review*, pages 369–426. North-Holland/IFIP WG8.1, Amsterdam, The Netherlands, EU, 1982.

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

[Som89] I. Sommerville. *Software Engineering*. Addison-Wesley, Reading, Massachusetts, USA, 1989.

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

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