Towards an Integration of Evolving Information Systems and CASE-Tools

H.A. Proper

Department of Information Systems, University of Nijmegen
Toernooiveld, NL-6525 ED Nijmegen, The Netherlands
E.Proper@acm.org

PUBLISHED AS:

H.A. Proper. Towards an Integration of Evolving Information Systems and CASE-Tools. In S. Brinkkemper and F. Harmsen, editors, *Proceedings of the Fourth Workshop on the Next Generation of CASE Tools*, pages 23–33, Paris, France, EU, June 1993. ISSN 09243755

Abstract

In this article, CASE-Tool technology is linked to the research concerning evolving information systems. First, an introduction to the notion of evolving information system is provided. Then, CASE-Tools and evolving information systems are related to each other from two different points of view, by looking how each one can server the other.

1 Introduction

Nowadays, the financial prosperety of an organisation depends more and more on its ability to change. By being flexible, an organisation can be more competitive on the global market place, thus improving its chances of survival. This means that organisations must be able to adapt themselves quickly to the production of new or different products - changes in the primary process of an organisation - resulting from the ever changing and more and more demanding consumer needs.

Flexible behaviour of an organisation implies rapidly changing information needs, and therefore call for more flexible information systems. Given the fact that information is gradually becoming a production factor of more and more importance, the need for flexible information systems increases significantly. Furthermore, the rise of (software!) automation costs is of increasing concern to many organisations ([VW91]).

The need for information systems, not only allowing for changes of their information base, but also for modifications in their underlying structure (conceptual schema and specification of dynamic aspects) has also been identified in [MS90], [Ari91], [Rod91], [JMSV92] and [FOP92b]. The intention of an evolving information system ([FOP92a]) is to be able to handle updates of all components of the so-called *application model*, containing the information structure, the constraints on this structure, the population conforming to this structure and the possible operations.

As an illustration of an evolving universe of discourse, consider a library, or rental store, for audio records (lp's). In this library, a record is kept of, among other things, the songs that are recorded on the lp's present in the library. In order to keep track of the wear and tear of the lp's, the number of times the lp has been lent is recorded as well. This part of the universe of discourse of the library has been modelled in figure 1 in the style of ER. Note that we abbreviated the graphical notation of attributes (Title) to a mark symbol (#) followed by the attribute (# Title), for reasons of readability.

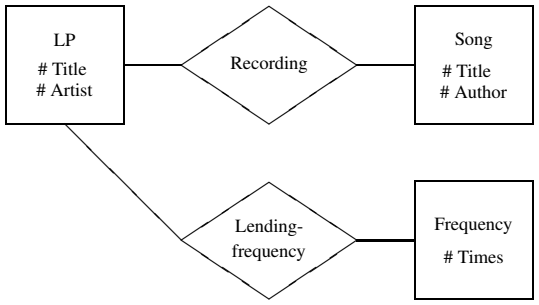

Figure 1: The Data Model of an lp library

A possible action specification for the action model of this example universe of discourse would be the following rule, stating that every time a new lp is added, it's lending frequency is set to 0:

WHEN ADD Lp: x
THEN ADD Lp: x has-a Lending-Frequency of Frequency: 0

After the introduction of the compact disc, and its conquest of a sizeable piece of the market, the library has become an lp and cd library. This resulted in the, evolved,

universe of discourse modelled in figure 2. In the new situation, the record dealing with recordings of songs on lp's is still kept, but is extended to cd's as well. The frequency of lending, however, is not kept for cd's, as the cd's are hardly subject to any wear and tear.

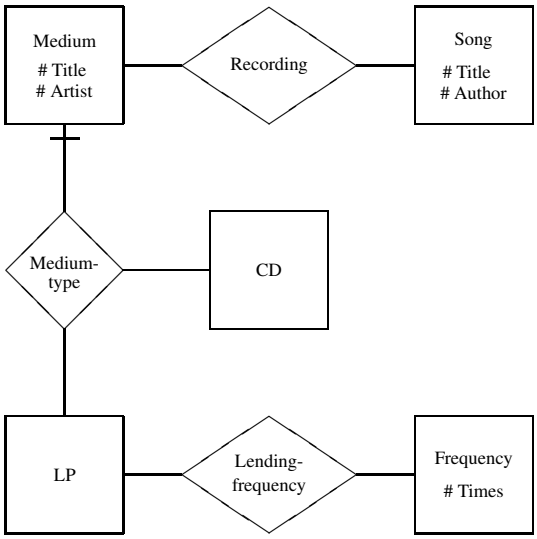

Figure 2: The Data Model of a lp and cd library

The action specification of the initial universe of discourse, evolves accordingly resulting in the following rule, stating that for each added medium, if it is an lp, it's lending frequency is set to 0:

WHEN ADD Medium: x
IF Lp: x THEN
  ADD Lp: x has-a Lending-Frequency of Frequency: 0

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

Related research regarding evolving information systems can be found in [MS90], [Ari91] and in the area of *version modelling* in engineering databases: [BCG$^+$87], [Kat90], [JMSV92]. A first database system supporting some aspects of evolution is the ORION system ([BKKK87], [BCG$^+$87], [KBC$^+$89]). In [MS90] a relational algebra is presented in which relational tables are allowed to evolve, e.g. change their arity.

Version modelling in engineering databases offers a fast body of knowledge concerning evolution of several types of engineering applications. The requirements for evolving information systems ([FOP92b]) are indeed related to the general requirements for version modelling as presented in [Kat90]. An important requirement for evolving information systems, however, is that changes to the structure can be made on-line. In traditional approaches to the evolution of information systems, and software evolution in general, a structural change still requires the replacement of the old system by a new system. This latter notion of evolution is the approach to evolving information systems as taken in [JMSV92], there the focus is on the support of evolution of the specification of the information system alone. A further distinction between the version modelling approach, and our approach to the evolution of information systems is, that we are able to provide well formedness rules regarding evolution of application models ([PW93]).

In this paper, we discuss a possible integration between evolving information systems and CASE-Tools ([McC89]). In section 2 we provide a short overview of the architecture of an evolving information system, further clarifying the notion of evolving information system. Note however, that it is not our intention to provide a detailed discussion on evolivng information systems. In section 3 we augment this architecture, with the architecture of an envisioned evolving information system shell (EIS-Shell). Before relating evolving information systems to CASE-Tools, the methodological aspects of such systems are related, in section 4, to the traditional notion of method. Two possible integrations between EIS-Shells and CASE-Tools are discussed in section 5 and 6. The support which can be provided by CASE-Tools to EIS-Shells is discussed in section 5. Conversely, the effect of considering an EIS-Shell as a CASE-Shell ([HVWB90], [VHW91]) is discussed in section 6.

2 An Architecture for Evolving Information Systems

In this section, we provide an overview of the architecture for evolving information systems, and discuss the way in which the evolution of application domains is modelled (for more details, refer to [PW93]). We start out with the identification of that part of an information system that can be subject to evolution, due to evolution of the universe of discourse. From this definition, the difference between a traditional information system, and its evolving counterpart, will become clear. This is followed by a discussion on how the evolution of a universe of discourse can be dealt with.

2.1 The Extent of Evolution

A complete specification of a universe of discourse typically ([ISO87]) contains the following components:

1. an intentional description of the set of states, also called the underlying *information structure*.
2. a further refinement of the set of states by means of *static constraints*.

3. an intentional description of the set of transitions that can be performed automatically (by the system), usually as a set of *action specifications* that bring about those transitions as a reaction to other transitions, caused automatically or manually.

4. a refinement of the set of possible transitions (between valid states), both automatically and manually, by means of *dynamic constraints*.

5. an extentional specification of the current state of the universe of discourse, i.e. the *population*, or information base, of the underlying *information structure*.

The notion of *application model* is defined ([FOP92a], [FOP92b], [PW93]) as the formal description of the universe of discourse.

In most traditional information systems, the part of the system that is allowed to change in the course of time is very restricted. First of all, most traditional information systems only allow for update of the information base, i.e. the set of facts which obeys a fixed (conceptual) schema with a fixed set of constraints. In other words, update of the conceptual schema, and consequently the internal data base schema, constraints, and specifications of dynamic aspects - i.e. the action model - have not yet been supported by these traditional information systems. Some systems do allow modifications on other components of the application model, besides the information base, to a limited extend. For example, adding a new table in an SQL system is easily done. However, changing the arity of a table, or some of its attributes, will result in a time consuming table conversion. This conversion also leads to loss of the old table!

Evolving information systems should, however, support update of all these specifications. In an evolving information system, all components of the application model are allowed to change in the course of time. Besides, evolving information systems are supposed to react online on update requests reflecting the evolution of all these components.

2.2 The Architecture

The application model can be subdivided further into two major sub-models. The *world model*, encompassing the combination of information structure, both static and dynamic constraints, and a population conforming to these requirements. Traditionally, a world model is provided as a fixed data model conforming to a modelling technique like NIAM ([NH89]), ER ([Che76]), or for more complex applications a schema conform IFO ([AH87]), or PSM ([HW93], [HPW92]) together with an information base conforming to this data model.

The other part of the application model is the *action model*. The action model is a set of action specifications, describing the transitions that can be performed by the system. An action model is usually modelled by means of petri-net like specifications, for instance ExSpect ([HSV89]) or Task Structures ([WHO92]), describing transitions on populations. Note that there exist (unified) modelling techniques, such as Telos ([JMSV92]), and TMT ([Hof93]), which intend to specify both world and action model in one single consistent model.

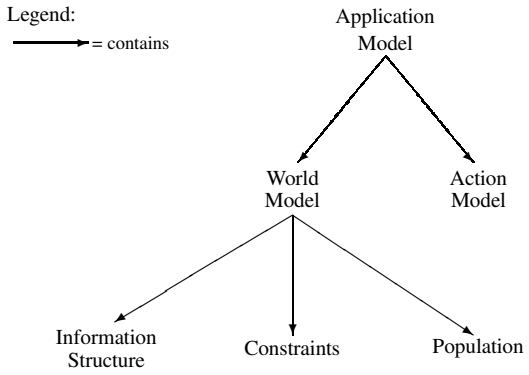

Figure 3: A Hierarchy of Models

The above definitions, result in the hierarchy of models, as denoted in figure 3. There the distinct models, their interrelationships, and their respective components are depicted. In practice, the application model describing a universe of discourse is denoted in terms of *object types*, *constraints*, a *population*, *action specifications*, etc. As a collective noun for these modelling concepts the term *application model element* is used ([FOP92a]).

The main difference between a traditional information system and an evolving information system, is now best explained by means of the general architecture for evolving information systems as depicted in figure 4 ([FOP92b]). In an evolving information system, the only fixed part is the *meta model*. A meta model is *time-* and *application independent*, it contains (and is restricted to) all rules about the languages used to model the application model and to formulate user *requests*. The complete application model can be updated in an evolving information system. These changes are performed by the information processing activities (done by the information processor) as a result of update requests from the users of the system. Since the history of the application model is maintained, the information processing activities operate upon the history of the application model, and not just on one version.

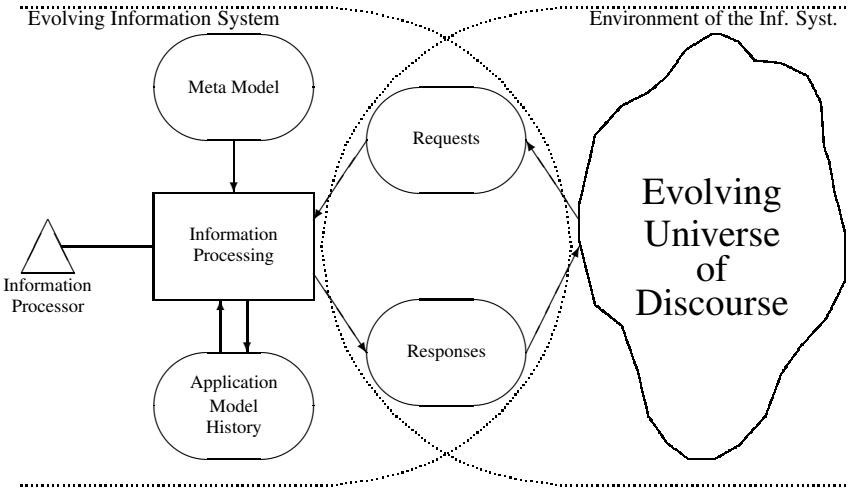

Figure 4: Evolving Information System Architecture

2.3 The Approach to the Modelling of Evolution

The three ER schemata, together with the associated action specifications, as discussed in section 1, correspond to three distinct snapshots of an evolving universe of discourse. Several approaches can be taken to the modelling of this evolution. In [PW93], these alternative approaches are discussed in more detail.

In our approach, we treat the evolution of an application model as a distinct concept. Furthermore, we model the evolution of an application model as the evolution of its elements, thus keeping track of the evolution of individual object types, instances, action specifications (also referred to as *methods*), etc. This has been illustrated in figure 5. Each dotted line corresponds to the evolution of one distinct element.

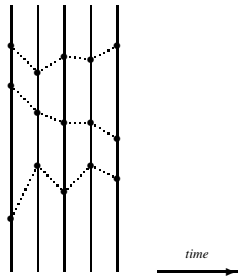

Figure 5: Evolution modelled by functions over time

The major advantage of this approach to the modelling of evolution is that it enables one to state rules about, and query, the evolution of distinct application model elements. Furthermore, we are able to derive a snapshot view from the set of element evolutions, by constituting the application model version of any point of time from the current versions of its components. This derivation is examplified in figure 6.

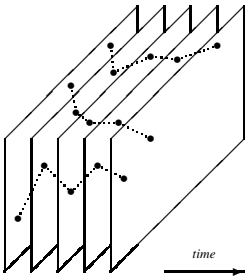

Figure 6: Deriving snapshots from element evolutions

The above described approach to the evolution of elements, has a parallel in the description of the history (evolution) of the world. Many approaches are possible. One may choose to describe the evolution of the world as a sequence of snapshots, where each snapshot contains all facts valid at a given point in time. This (higly inefficient) way of describing the history of the world is not used in practice, as one always wants to know the distinct evolution of persons, municipalities, families, laws, countries, etc. Consequently, if one wants to make a complete description of the evolution of the world, the evolutions of all relevant components of the world (persons, mountains, tectonic plates, ...) have to be described seperately.

3 Evolving Information System Shells

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*. 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 (ER, NIAM, Task Structures, etc.) defined in the meta model of the evolving information system shell (EIS-Shell). In this section we focus on the information processing in an EIS-Shell.

The architecture for evolving information systems as depicted in figure 4 is on a conceptual level. The architecture does not take the notion of recording time ([SA85]) into consideration. When indeed taking recording time into consideration, the information processing has to be refined conform figure 7 (page 6). Due to the possible corrections, and the maintenance of recording time, several *versions of application model histories* can be distincted. When performing requests from the users, the proper application model history must be selected by the *version manager*.

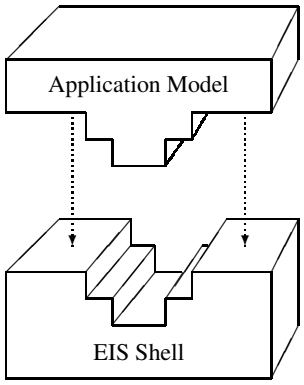

Figure 8: The EIS shell: independent of any application model

An EIS-Shell is independent of any universe of discourse. As a consequence, application models describing different domains can be 'plugged' into the EIS-Shell. This principle is illustrated in figure 8. 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 operatating system. This is illustrated in figure 9.

4 Methodological aspects of Evolving Information Systems

In the process of the development of an EIS-Shell, three subobjectives ([FOP92b]) are distinguished:

1. The design of a meta model and a language (based on that meta model) for the specification of the *application model*. Furthermore, this language must be able to support all aspects of evolution.
2. The implementation of an evolving information system shell based on that meta model and language.
3. The development of a suitable procedure for the process of designing, building up and maintaining the application model.

The use of an EIS-Shell, together with the design/maintanance procedure, in an organisation can be seen as the use of a method for the modelling of the evolution of organisations. The three objectives as stated above, can together be regarded as an *evolving information systems method*. This can be motivated by relating these objectives to the components of a method.

Several definitions of the concept of method exist. An elaborated discussion can be found in [Wij91]. According to [Wij91] a method can be dissected in the following five aspects:

1. The *way of thinking* should provide a paradigm. It should define the assumptions made by the method with respect to the elements that consitute an information system, the function of an information system in relation to its environment, the environment of the information system, and the major characteristics of the components of the information system and its environment.
2. The *way of working* should structure the way in which an information system is developed. It defines the possible tasks, including sub-tasks, and ordering of tasks, to be performed as part of the development process. Furthermore, guidelines and suggestions (heuristics) on how these tasks should be performed.
3. The *way of modelling* provides the modelling concepts and their interrelationships. It structures the models which can be used in the information system development, i.e. it provides a (formal!) language in which to express the models.
4. The *way of controlling* deals with the managerial aspects of the information system development. It includes such aspects as planning and evaluation of plans, i.e. the overall project management.
5. The *way of support* of a method, refers to the support of the method by (automated) tools.

These five components of a method have been illustrated in figure 10, which is taken from [HW92], and based on [Wij91].

The way of thinking of an evolving information system has been discussed briefly in the previous sections. Objective 3 provides the way of controlling, and some indications for the way of working. The exact definition of the way of working depends on the way of working of the chosen modelling techniques for the components of the application model (the data/information model, the specification of activities to be performed by the information

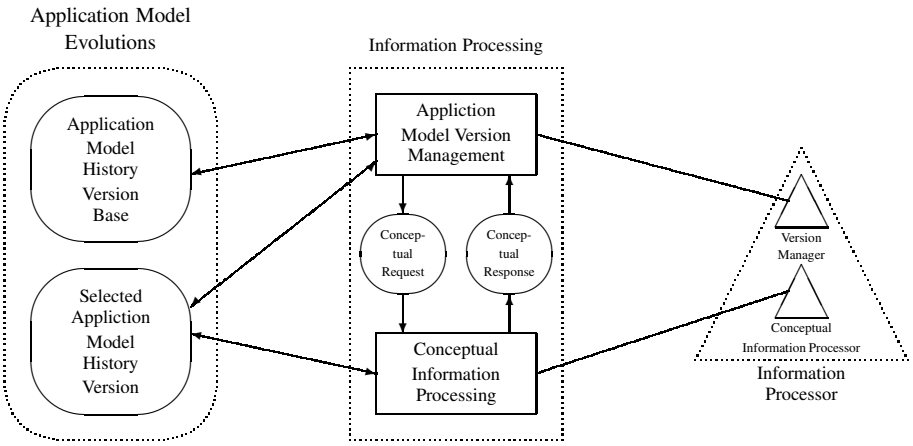

Figure 7: Information processing in an EIS shell

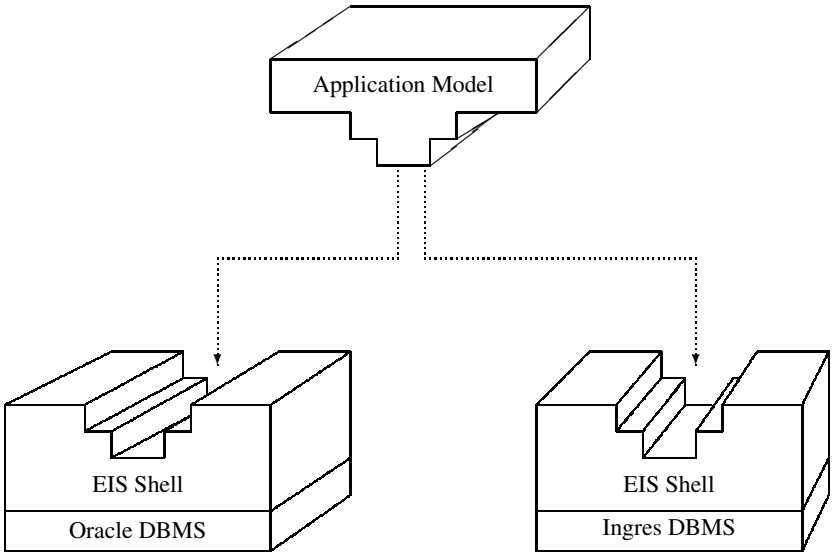

Figure 9: The EIS shell: independent of any software environment

system, etc.) The result of objective 3 can only provide an umbrella (framework) for the way of working.

Objective 1 provides the way of modelling of an EIS-Shell. The exact way of modelling in an EIS-Shell partially depends on the chosen modelling techniques for the application model. Finally, the EIS-Shell itself (objective 2) provides the way of support of the evolving information system method.

In [PW93], the focus is on objective 1, i.e. the way of modelling of an evolving information system. In this article we focus on the way of support as provided by an EIS-Shell, and how it can be enhanced by the support of a CASE-Tool. Furthermore, we look at the reverse, and relate the notion of CASE-Shell ([HVWB90]) to an EIS-Shell.

The way of controlling/working of an EIS-Shell will be closely related to the prototyping approach (see for instance [Som89]) to system development, as an EIS-Shell allows for quick changes of the application model. The prototyping approach has been illustrated in figure 11 (page 7).

5 CASE Support for Evolving Information Systems

When an application model evolves, the changes have to be communicated to the EIS-Shell by means of update requests. For changes to the population, this is rather straight forward as this can be done in a traditional way by means of an ADD, DELETE or CHANGE. For the other

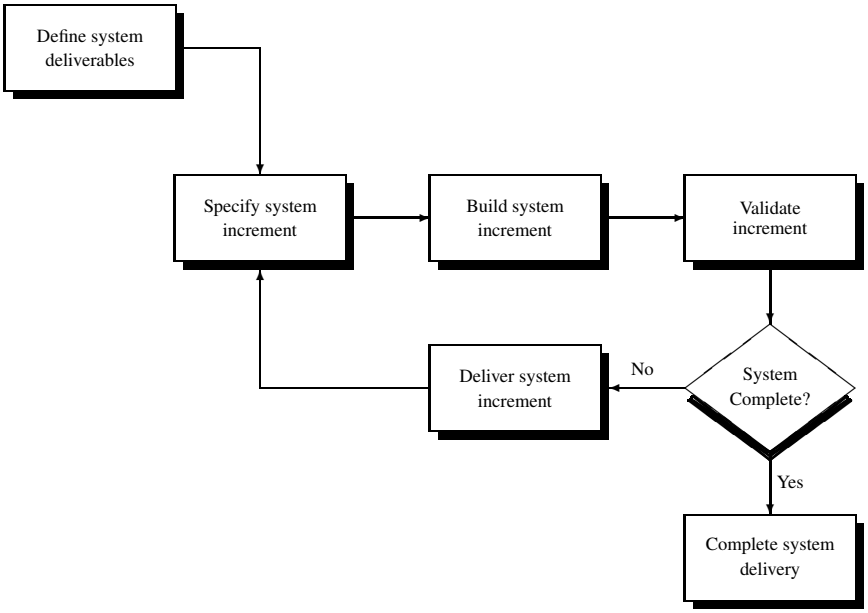

Figure 11: The Prototyping Approach

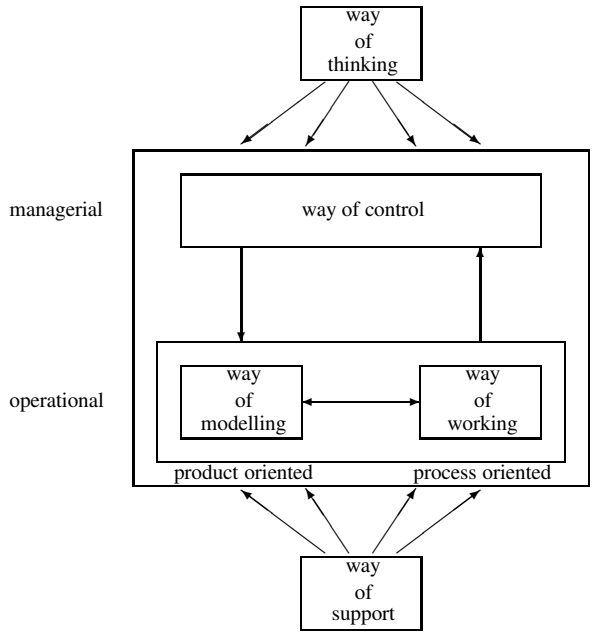

Figure 10: The components of a method

components, however, this is more difficult. Consider for example the addition of a serie of new object types, or the adding of an action specification. Such an operation would require a series of ADD statements, adding every single component.

5.1 Integration

When the information structure has to be modified, or an action specicifcation in the action model has to be changed, it is not advisable to do this by means of a (large number of) traditional ADD or CHANGE update request. Rather, one would like to specify such a modification by means of a CASE-Tool interface, and even *test* the envisioned modification in some way before comitting it to the EIS-Shell. This *test* may even consist of a prototyping phase, in order to asses the actually desired change to the application model present in the EIS-Shell.

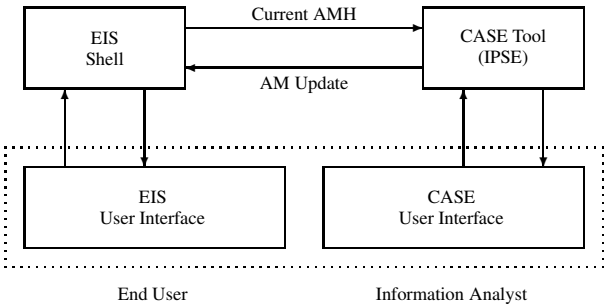

Figure 12: CASE and EIS integration

The integration between a CASE-Tool and an EIS-Shell is illustrated in figure 12. When a change to the application model is needed, the current (or older) version of the application model history (minus the population) is transferred to the repository of the CASE-Tool. This transfer does not have to be a *physical* transfer. Parts of the CASE-Tool's repository may indeed be *shared* with the EIS-Shell. Note that the 'CASE-Tool' in this context

should not just be an Analyst's Workbench, but a complete IPSE ([Bri90]), at least covering all the modelling techniques used for the application model.

The modifications to the (most recent) application model are made through the CASE-Tool, and on completion communicated to the EIS-Shell. It should be noted that no changes can be made to past versions of the application model, as this would lead to the falsification of the history. If such changes need to be made, this has to be done by means of a correction ([FOP92a]).

The consistency of the changed application model will be checked by the CASE-Tool, before communicating the changes to the EIS-Shell. Furthermore, the well-formedness of the evolution of the application model will be maintained. In order to do this, the updated application model must be related to the application model history as whole (see [PW93]). This is also the reason why the entire application model history has to be communicated to the CASE-Tool.

One of the requirements for a set of (integrated) CASE-Tools, is that it should have a consistent user interface, i.e. the user interfaces have a similar *look-and-feel*. Therefore, it is only obvious that the interfaces of the EIS-Shell, and the CASE-Tool should have a similar *look-and-feel*.

5.2 Hypertext Browsing of Information Structures

For users of traditional information system, it is already difficult to maintain an overview of the information structure of the stored information. This has lead to the idea of building a hypertext browser for such traditional information systems ([BPW93]), enabling an improved information disclosure. This browser allows for *query by navigation*, i.e. building a query whilst navigating through the information structure. Similar browsers have also proved their usefullness in CASE-Tools ([Big88], [GS90], [Hag92]).

For an EIS-Shell, the need for a hypertext-like browsing mechanism for the current, as well as past information structures, of the application model history is even more pressing. When an evolving information system changes fast, it is hardly possible for most users (and information analysts) to maintain an overview of the kind (structure) of the stored information of the past and present.

This observation leads to the refined architecture of an integrated EIS-Shell and CASE-Tool as depicted in figure 13.

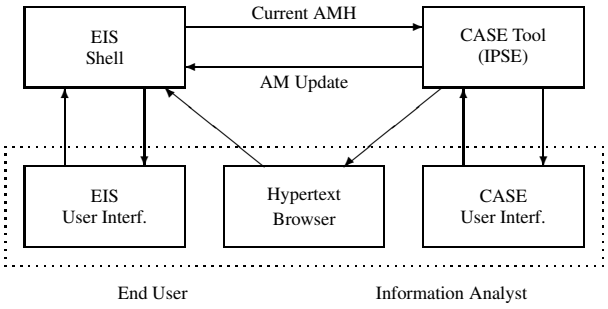

Figure 13: Hypertext Browser for an EIS-Shell and CASE-Tool

6 Second Order Evolution and Evolving CASE Shells

In [OHFB92] the notion of *evolution of the second order* is defined as the ability to change the used modelling techniques for the application model of an evolving information system in the course of time. This need arises when new kinds of applications are needed, requiring different modelling techniques then the applications which where used in the organisation thus far. For the support of second order evolution, a CASE-Shell ([VHW91]) is needed which supports the evolution of the modelling technique modelled in the CASE-Shell.

In the EIS research thus far we aim to use modelling techniques with a high level of expressiveness. The modelling techniques used for meta modelling as used in the SOCRATES project, indeed have a high level of expressiveness ([HW93]). As a result, these modelling techniques are also very usefull for the modelling of non-meta application models ([Hof93]). Therefore, we will use these modelling techniques for the application model of the EIS Shell. Currently, we are developping adopted versions of these modelling techniques supporting the evolution of models conforming to these techniques.

As a direct consequence, we will not only be able to model traditional (non-meta) applications, but meta-applications as well. Consequently, an EIS shell supporting these techniques will be an evolving CASE Shell as well! Such an Evolving CASE shell is indeed a proper vehicle to be able to support method engineering, in particular evolution of the second order. The relation between the three levels of abstraction recognised in a CASE Shell ([VHW91]), and the evolution theory is depicted in figure 14.

7 Conclusions

In this article we have provided a short definition of an evolving information system, and the accompanying EIS-Shell. We related evolving information systems and EIS-Shells to CASE-Tools. There obviously exists a close relationship between an EIS-Shell and CASE-Tools. We delfed into this relationship from two angles. From one angle we looked at the functionality of a CASE-Tool with respect to an EIS-Shell, and from the other angle we looked

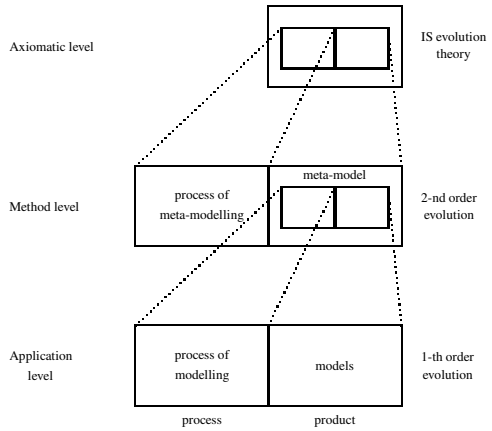

Figure 14: Three levels of abstraction

at the possibility of using an EIS-Shell as an (Evolving) CASE-Shell.

In the future, CASE-Tools may indeed support some aspects from the theory of evolving information systems, such as the well-formedness of the evolution of an application model.

References

[AH87] S. Abiteboul and R. Hull. IFO: A Formal Semantic Database Model. *ACM Transactions on Database Systems*, 12(4):525–565, December 1987.

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

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

[Big88] J. Bigelow. Hypertext and CASE. *IEEE Software*, 5(2):23–27, 1988.

[BKKK87] J. Banerjee, W. Kim, H.J. Kim, and H.F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *SIGMOD Record*, 16(3):311–322, December 1987.

[BPW93] C.A.J. Burgers, H.A. Proper, and Th.P. van der Weide. Organising an Information System as Stratified Hypermedia. In H.A. Wijshoff, editor, *Proceedings of the Computing Science in the Netherlands Conference*, pages 109–120, Utrecht, The Netherlands, EU, November 1993.

[Bri90] S. Brinkkemper. *Formalisation of Information Systems Modelling*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1990.

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

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

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

[GS90] P.K. Garg and W. Scacchi. A Hypertext System to Manage Software Life-Cycle Documents. *IEEE Software*, 7(3):90–98, 1990.

[Hag92] T.M. Hagensen. Hyperstructure CASE Tools. In B. Theodoulidis and A. Sutcliffe, editors, *Proceedings of the Third Workshop on the Next Generation of CASE Tools*, pages 291–297, Manchester, United Kingdom, May 1992.

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

[HPW92] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Data Modelling in Complex Application Domains. In P. Loucopoulos, editor, *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, volume 593 of *Lecture Notes in Computer Science*, pages 364–377, Manchester, United Kingdom, EU, May 1992. Springer Verlag, Berlin, Germany, EU. ISBN 3540554815

[HSV89] K.M. van Hee, L.J. Somers, and M. Voorhoeve. Executable Specifications for Distributed Information Systems. In E.D. Falkenberg and P. Lindgreen, editors, *Information System Concepts: An In-depth Analysis*, pages 139–156. North-Holland/IFIP, Amsterdam, The Netherlands, 1989.

[HVWB90] A.H.M. ter Hofstede, T.F. Verhoef, G.M. Wijers, and S. Brinkkemper. The SOCRATES project. In S. Brinkkemper and G.M. Wijers, editors, *Proceedings of the First Workshop on the Next Generation of CASE Tools*, Noordwijkerhout, The Netherlands, April 1990.

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

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

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

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

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

[KBC+89] W. Kim, N. Ballou, H.-T. Chou, J.F. Garza, and D. Woelk. Features of the ORION Object-Oriented Database. In W. Kim and F.H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications*, ACM Press, Frontier Series, pages 251–282. Addison-Wesley, Reading, Massachusetts, 1989.

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

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

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

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

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

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

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

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

[VHW91] T.F. Verhoef, A.H.M. ter Hofstede, and G.M. Wijers. Structuring modelling knowledge for CASE shells. In R. Andersen, J.A. Bubenko, and A. Sølvberg, editors, *Proceedings of the Third International Conference CAiSE'91 on Advanced Information Systems Engineering*, volume 498 of *Lecture Notes in Computer Science*, pages 502–524, Trondheim, Norway, May 1991. Springer-Verlag.

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

[WHO92] G.M. Wijers, A.H.M. ter Hofstede, and N.E. van Oosterom. Representation of Information Modelling Knowledge. In V.-P. Tahvanainen and K. Lyytinen, editors, *Next*

Generation CASE Tools, volume 3 of *Studies in Computer and Communication Systems*, pages 167–223. IOS Press, 1992.

[Wij91] G.M. Wijers. *Modelling Support in Information Systems Development*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 1991. ISBN 9051701101