Diversity of structures and adaptive methods on an evolutionary hypermedia system

N. Medina-Medina, F. Molina-Ortiz and L. García-Cabrera

Abstract: SEM-HP is a model for the development of evolutionary and adaptive hypermedia systems by means of: a development process that consists in iterative and evolutionary phases; a layered architecture that captures each phase in subsystems divided into two levels of abstraction (system and metasystem); and an author tool that implements the model. The paper focuses on the capability of adaptation of the hypermedia systems developed according to the SEM-HP model, in particular, the learning subsystem that is in charge of performing the user adaptation. SEM-HP offers two important contributions: supporting the process of evolution of the development of adaptive hypermedia systems, ensuring easy, flexible and consistent maintenance; and the orientation support is that inherent to the navigational structure itself.

1 Introduction: the SEM-HP model

SEM-HP [1] is a model for the development of hypermedia systems (HS) with antecedents in the Wang [2] and Stotts [3] approaches. SEM-HP provides to the author the necessary formalisms, techniques and methods to support two essential aspects: evolution and adaptation. The *ability to evolve* eases the creation of the HS and guarantees its long-term life. The evolutionary mechanisms provided by SEM-HP allow the HS to incorporate the needed changes in an easy, flexible and consistent way. The *ability to adapt* the HS to the needs and features of each user is essential if we take into account that there will be users with very different profiles accessing our system. This process can be seen as a particular case of evolution where the system automatically changes its behaviour depending on the user utilising it [4].

SEM-HP model provides the author with three elements for the creation of evolutionary and adaptive hypermedia systems: a development process, an architecture and an author tool. The development process establishes the guidelines to follow for the creation of the HS from a software engineering approach. The architecture describes the representation models used to capture each phase of the development process. Finally, the author tool eases the creation of the system according to the architecture and the development process proposed in the model.

In this paper we firstly outline the SEM-HP architecture, describing with more detail the learning subsystem, which takes care of the user adaptation and maintains a user model, and update and knowledge rules. After that, we describe the adaptive methods and techniques used in SEM-HP (choosing the appropriate subdomain and sets of rules, and personalisation to user knowledge, his interests and goals). Finally, conclusions, related work and further work are discussed.

IEE Proceedings online no. 20050026

doi: 10.1049/ip-sen:20050026

Paper received 8th April 2005

N. Medina-Medina and F. Molina-Ortiz are with the Depto. L.S.I., E.T.S.I. Informática, Universidad de Granada, Spain

L. García-Cabrera is with the Depto. Informática, Universidad de Jaén, Spain

E-mail: nmedina@ugr.es

2 SEM-HP architecture

The architecture proposed by the SEM-HP model (Fig. 1) is structured in layers, performing a double division [5]: vertical and horizontal. The vertical division considers four subsystems: memorisation subsystem, presentation subsystem, navigation subsystem and learning subsystem. The horizontal division distinguishes two layers in each subsystem: system and metasystem. The least abstract level (system) comprises the representation models defined by the author during the corresponding development phase, while the most abstract level (metasystem) includes the evolutionary mechanisms that will permit the integration and propagation of changes performed by the author in the elements of the corresponding subsystem.

The evolutionary mechanisms included in the metasystem are mainly three [6]: evolutionary actions, restrictions and change propagation. The author interacts with the metasystem to build and modify the HS. To perform a change in a subsystem the author must choose the appropriate evolutionary action and run it. To ensure the integrity of the HS, an evolutionary action is only executed if it satisfies a set of restrictions imposed by the model and by the author. In addition, when changing an element in a subsystem, the need may arise to modify other elements in the same subsystem or even in other subsystems. This change propagation is automatically done in the model, guaranteeing, in addition to the consistent evolution in each subsystem, the coevolution of the set of subsystems.

In this paper we focus on the capability of adaptation of the HS developed according to the SEM-HP model, so we will briefly outline the three first subsystems [5], and we will describe in depth the learning subsystem.

The *memorisation subsystem* stores, structures and maintains the conceptual and informational domain of the HS. The representation model used for describing both domains is a conceptual structure, CS_M (Fig. 2), or semantic net with two kinds of nodes: concepts (C) and items (I). Items are informational units (text, image, audio, video, executable, etc.). The concepts are connected between them

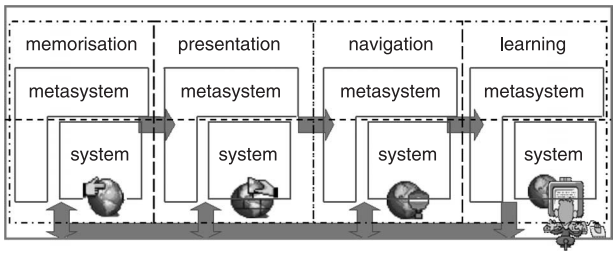


Fig. 1 *SEM-HP architecture for a hypermedia system*

through semantically labelled relations, called conceptual relations. The informational items are associated with concepts; in this way the links between items are established based on the semantic relations among concepts. The association between an item and a concept is called functional association and it is labelled with the role the item plays with respect to the concept. In addition to the role, each item has some properties that determine the main features of the information it contains: author, date of creation, level of difficulty, media (audio, text, video, etc.), language, etc.

In the *presentation subsystem* the author selects different subsets of the CS_M created in the memorisation phase, with the aim of reducing its complexity and size (Fig. 3). Creating different views of the CS_M achieves: (*a*) provision to the user of a navigation structure focused on the knowledge subdomain in which he is interested; and (*b*) reduction of the disorientation problems. Based on the same knowledge domain (conceptual and informational domain captured in a CS_M), the author prepares different conceptual structures of presentation (CS_P). The author is in charge of defining the subdomains and of labelling each CS_P with the subdomain it represents.

In the *navigation subsystem* the author establishes the navigability of the conceptual relations. By default, the conceptual relations will be navigated from the origin to the destination concept. Nevertheless, if the author wishes, he can extend the navigability of a conceptual relation in the other direction, so that it can be navigated in both ways. The author defines the navigational possibilities for each presentation, being able to create different conceptual structures of navigation (CS_N) from the same CS_P.

3 Learning subsystem

The learning subsystem permits the HS developed according to the SEM-HP model to be qualified as *adaptive*. In the two subsystems described above the author prepares different navigation structures for the same knowledge domain. These structures differ in the knowledge subdomain they represent and in the way their conceptual relationships can be navigated. Because of this diversity, the system can offer to the user the navigation structure that best matches his profile this being understood as several general features, such as the subdomain he is interested in, experience in the subject and experience in navigation. Two users with the same profile will obtain the same structure – nevertheless they are different. For example, although both may be interested in the same subdomain, their knowledge goals can be different, or, even if both have similar average experience in the subject, one may know some concepts better than the other. Hence, individual adaptation methods and techniques are needed, which in the SEM-HP architecture are included in the learning subsystem. The elements the learning subsystem uses to apply these adaptation methods are: a user model and a set of learning rules (update rules and knowledge rules). Again, for the same CS_N the author can define different sets of learning rules, creating different conceptual structures of learning (CS_L).

3.1 User model

The user model is the internal representation of the user that the learning subsystem keeps. The information the user model contains is shown in Table 1. Most of the initial information is explicitly set by the user the first time he logs on to the system. Part of that information will stay unchanged until the user requests to change it, while other information will be automatically updated as the user browses the HS. So, we can characterise the HS developed according to the SEM-HP model as adaptable and adaptive (following the definition of both terms shown in [7]).

The user will choose the knowledge subdomain in which he is most interested, selecting one of the subdomains defined by the author in the presentation phase. Whenever he wishes he will be able to choose another subdomain by a direct petition to the system. The initialisation and update tasks are performed automatically for other entries in the user model, such as the knowledge degree and the number of visits. The first time the user logs on to the system the number of visits is 0 and his knowledge about every item in the HS is set to 'null'. While the user navigates, the system counts his visits and updates the degree of knowledge about the visited items and other items related to them using the update rules. There are other entries which are updated by the system upon the user's request. This is the case of the

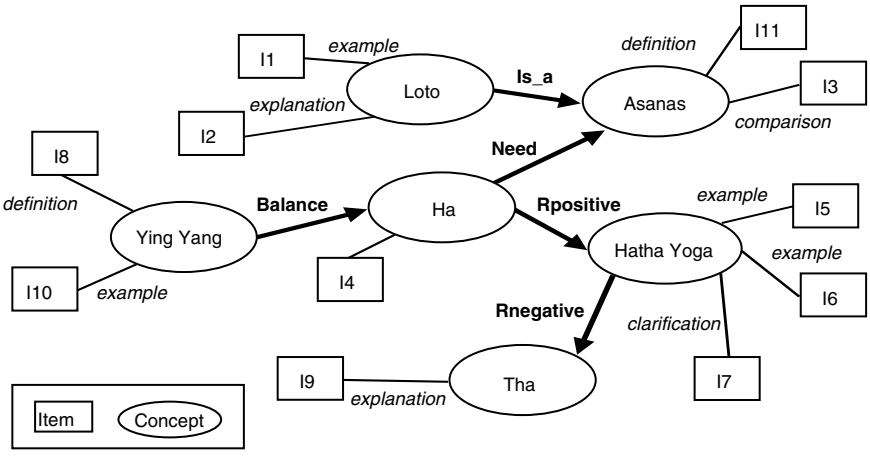


Fig. 2 *Conceptual structure of memorisation*

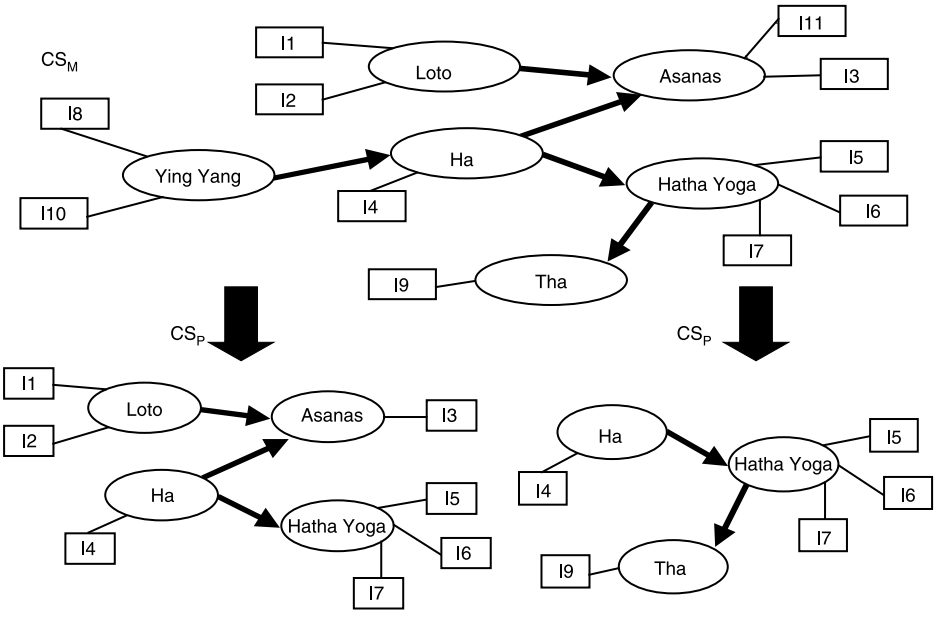


Fig. 3 *Two different views of the same CS_M*

Table 1: User model

		Initialisation	Update
Subdomain of interest		Asked of the user	Performed by the user
Experience in the subject		Asked of the user	Requested by the user
Experience in navigation		Asked of the user	Requested by the user
For each item	Degree of knowledge	Automatic	Automatic
	No. of visits	Automatic	Automatic
Interests / goal		Asked of the user	Performed by the user
Preferences		Asked of the user	Performed by the user
Personal information		Asked of the user	Performed by the user

experience of the user in the subject and in navigation. Initially the system can only obtain this information by explicitly asking the user, but later the system can automatically infer these data from other information in the user model. The subject experience can be approximated as the average user knowledge, and the navigation experience can be inferred from the total number of visits performed.

3.2 Update rules

The degree of user knowledge about an item I, $K(I)$, is represented in the user model using any of the semantic labels ($label_{SEM}$): 'null', 'very low', 'low', 'medium', 'high', 'very high' and 'total', each one with an associated numerical value from one to seven. The update rules (R_u) determine how, after every visit the user carries out, his degree of knowledge about certain items is increased. There will be an R_u associated with the visit to each item in the CS_L. In it, the head is the visited item and the body includes a set of update predicates that will be executed after the visit. Equation (1) shows the generic structure of an R_u:

$$\text{Visit}(I) \rightarrow \text{Update}(I),\ \text{Update}(I'') \ldots \text{Update}(I''') \quad (1)$$

Each one of the predicates updates a different item in the CS_L. The update of the user knowledge about an item can be: incremental or fixed (the degree of user knowledge is set to a fixed degree of knowledge), absolute (it uses an absolute $label_{SEM}$) or relative (it uses the degree of user knowledge about the visited item), each-time or first-time (it is run only the first time the item in the head is visited). Table 2 shows the available update predicates and Table 3 includes examples of updates to the item *I*6 after *I*4 is visited.

To help the author, the system generates by default a set of R_u for each CS_L. In the body of an R_u there is only one

Table 2: Update predicates

		Each-time	First-time
Incremental	Absolute	Ink-abs (*I*, number-of-degrees)	$Ink\text{-}abs_{first}$ (*I*, number-of-grades)
	Relative	Ink-rel (*I*, number-of-degrees)	$Ink\text{-}rel_{first}$ (*I*, number-of-degrees)
Fixed	Absolute	Fix-abs (*I*, $label_{SEM}$)	$Fix\text{-}abs_{first}$ (*I*, $label_{SEM}$)
	Relative	Fix-rel (*I*)	$Fix\text{-}rel_{first}$ (*I*)

Table 3: Some possible updates on the item I6 after the visit to I4

	Update(I6)	K(I6) after visit to I4
Current User Knowledge: $K(I6) =$ 'null', $K(I4) =$ 'medium'	Ink-abs(I6, 2)	$K(I6) = K(I6) + 2 =$ 'low'
	Ink-rel(I6,2)	$K(I6) = K(I4) + 2 =$ 'very high'
Update Rule: Visit $(I4) \rightarrow \ldots,$ Update$(I6), \ldots$	Fix-abs(I6, 'total')	$K(I6) =$ 'total'
	Fix-rel(I6)	$K(I6) = K(I4) =$ 'medium'

update predicate by default, which sets to 'total' the degree of knowledge about the visited item. For example: Ru$(I4)$: *Visit*$(I4) \rightarrow$ *Fix-abs*$(I4,$ *'total'*$)$. The author can make the default update rules evolve, adding updates about other items different from the visited item. All the changes are done through evolutionary actions, so they will not be run unless they satisfy several restrictions, such as not removing the visited item's update or not including two updates about the same item. The restrictions guarantee that the set of R_u satisfies some desirable features, such as that every R_u must permit the user to achieve a 'total' knowledge about the head item after a finite number of visits to it. For example, by performing valid changes, the author can transform the default rule for $I4$ into this other (the changes are in boldface): Ru$(I4)$: *Visit*$(I4) \rightarrow$ Fix-abs$(I4,$ 'total'$)$, **Ink-abs(*I6*, 2), Ink-abs(*I7*, 3)**.

3.3 Knowledge rules

The author defines, for each CS_L, a set of knowledge rules (R_k). These R_k will be used to adapt the user navigation depending on his degree of knowledge. Hence, the R_k associated with an item determines which other items must be known by the user and which degree of knowledge he must have about them in order to access the item. Equation (2) shows the structure of an R_k and Table 4 contains the allowed knowledge restrictions:

$$\begin{aligned} &\text{Knowledge\_restriction}(\text{I}')\ \text{op}_\text{L} \ldots \text{op}_\text{L} \\ &\text{Knowledge\_restriction}(\text{I}'') \rightarrow \text{Visitable}(\text{I}) \\ &\text{where op}_\text{L} \in \{\text{and, or}\} \end{aligned} \tag{2}$$

Table 4: Knowledge restrictions in the body of R_k

	Strict	Not strict
Equality	$K(I) = \text{label}_{\text{SEM}}$	
Larger	$K(I) > \text{label}_{\text{SEM}}$	$K(I) >= \text{label}_{\text{SEM}}$
Smaller	$K(I) < \text{label}_{\text{SEM}}$	$K(I) <= \text{label}_{\text{SEM}}$
Interval	$I1 < K(I) < I2$	$I1 <= K(I) <= I2$
	where I1 and I2 are $\text{label}_{\text{SEM}}$ and $I1 < I2$	
OR_{SET}	$K(I) \propto$ [restriction$_1$, restriction$_2, \ldots,$ restriction$_N$] which means $K(I)$ satisfies at least one restriction in the set	

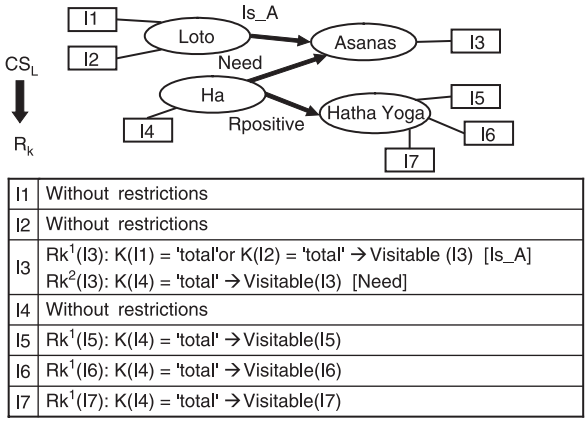
Once more, to assist the author, the system generates an set of default R_k, which can be changed by the author using the appropriate evolutionary actions. For each item in the CS_L the system generates an R_k for each conceptual relation which can be navigated up to the concept to which the item is associated. The body of the rule expresses the requirement of a 'total' knowledge about any of the items associated with the origin concept of the relation. Figure 4 shows the R_k automatically generated in the example.

Some modifications the author can carry out in an R_k are: remove an R_k, change the logical operators that connect the knowledge restrictions, add new restrictions or change the restriction about an item. For example, the author can change the default R_k for I_3 (Fig. 4), producing the R_k shown in Table 5.

Before performing the changes requested by the author, the metasystem checks that the integrity restrictions hold. For example, the set of R_k must guarantee that performing the appropriate navigation steps, every item in the CS_L will become accessible (when the navigation is adapted to the user knowledge, an item is accessible if the user fulfils at least one of the R_k associated with it). This restriction will depend not only on the R_k but also on the R_u. To check it, the system builds a knowledge navigation tree that permits one to check that all the items are accessible, so as a result it

Table 5: Changes in the R_k associated with the visit of I3

$Rk^1(I3)$: $K(I1) >$ 'null' or $K(I2) >=$ 'medium' → Visitable $(I3)$ [IsA]
$Rk^2(I3)$: ($K(I2) >$ 'high' or $K(I6) >$ 'high') and $K(I4) >$ 'high' → Visitable $(I3)$ [Need]



I1	Without restrictions
I2	Without restrictions
I3	Rk^1(I3): K(I1) = 'total'or K(I2) = 'total' → Visitable (I3) [Is_A] Rk^2(I3): K(I4) = 'total' → Visitable(I3) [Need]
I4	Without restrictions
I5	Rk^1(I5): K(I4) = 'total' → Visitable(I5)
I6	Rk^1(I6): K(I4) = 'total' → Visitable(I6)
I7	Rk^1(I7): K(I4) = 'total' → Visitable(I7)

Fig. 4 *Default knowledge rules*

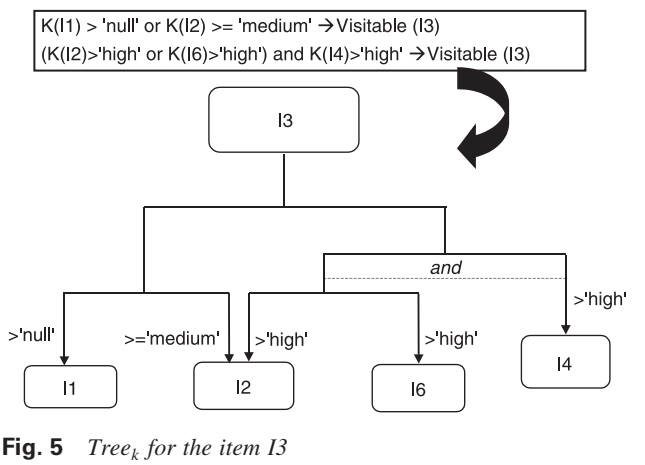


Fig. 5 *Tree$_k$ for the item I3*

is possible to reach a 'total' knowledge about each one of them (more details in Section 4.2).

Every R_k can be expressed as a knowledge restriction tree ($Tree_K$). The head item of the R_k is the root of the $Tree_K$. The composite restrictions (restrictions joined by logic operators) in the body of the R_k are decomposed in the restrictions that integrate it, generating subtrees inside the $Tree_K$. Every single knowledge restriction is represented by a child node labelled with the item implied in the restriction. The branch getting to the node is labelled with the knowledge degree required for the item in the restriction. To represent the *and* operator we join the implied branches with a horizontal dotted line. All the R_k associated with an item can be unified in a single tree using *or* branches, since it is enough that one R_k holds for the item to be accessible. Figure 5 shows the $Tree_K$ that will allow one to determine whether *I*3 is accessible.

4 Adaptive methods and techniques

4.1 Choosing the CS_L

In each development phase the author prepares different CSs, starting from each of the CSs defined in the previous phase. For example, if the author builds two CSs based upon each CS in the preceding phase, the number of CS_Ls available to be browsed by the user will be 2^3 (Fig. 6).

During the development process of the HS the author must label the created CSs so that the system can automatically determine which is the one that best fits the user. There are three labels for each CS_L:

- *Knowledge subdomain*: The knowledge subdomain shown by a CS_L is established in the presentation phase, since in this phase the author perfectly knows which part of the knowledge domain represented in the CS_M he has captured in the presentation.
- *Experience in navigation*: The value of this label is defined in the presentation and navigation phases. For example, a conceptual structure with a considerable amount of items, concepts and extended conceptual relations offers very many navigation possibilities, and consequently users with little experience in navigation are more likely to feel lost.
- *Experience in the subject*: The value of this label is determined by design decisions taken by the author in the presentation and learning phases. For example, with restrictive knowledge rules and update rules that quickly increase the user's knowledge, a high level of experience is required.

Since there are three attributes in the user model with the same name, meaning and value range, the chosen CS_L for each user is the one whose values in these three labels are closer to the values of the corresponding attributes in his user model.

4.2 CS_L personalised to user's knowledge

During the user navigation, the system personalises the CS_L through the annotation of the visited items and the visualisation in the CS_L itself of the user's knowledge state. The annotation of the previously visited items with a special colour permits us to avoid undesired repeated visits. The visual labelling of the items with the knowledge degree the user has about them allows him to check how his knowledge increases as he browses the HS. This makes the user aware of the knowledge prerequisites of the system and of his learning process.

In addition, the system uses the item hiding technique, in which forbidden items are hidden and disabled, ensuring that the user only visits accessible items according to the R_k defined in CS_L. With this adaptive technique the visible items in the CS_L depend on such an individual thing as the knowledge state of the user, restricting the navigation in the function of his knowledge. In Fig. 7 the adaptation of a CS_L after visiting the item *I*4 is shown.

To calculate the accessible items, the R_k must be evaluated on the user's current knowledge state. To efficiently perform this task, the system creates a knowledge navigation tree ($Navigation_K$ Tree) for each CS_L, applying its R_k and R_u. Each node in the tree represents a knowledge state in which the user can be, that is, the degree of knowledge the user has in a particular navigational step about every item in the CS_L. The root node represents total ignorance ('null' knowledge about all the items), and the leaf nodes represent 'total' knowledge about all the items. Each node has as many children as items are accessible from the knowledge state represented by it. The branch that leads to each child node is labelled with the name of the visited item and the child node is built applying the visited item's R_u on the knowledge state represented by the parent node. In Fig. 8 we can see part of the $Navigation_K$ Tree generated for the CS_L in Fig. 7, using the default R_u and R_k, after applying to them the changes shown in Sections 3.2 and 3.3 for $R_u(I4)$ and $R_k(I3)$. The knowledge state of a user who has only visited *I*4 is marked in the Figure.

Once the $Navigation_K$ Tree has been generated for a CS_L, it is used to restrict the navigation by knowledge of all the users who browse it. This tree will only be changed when the R_u or the R_k that were used to generate it change, be it requested by the author through an evolutionary action or caused by an automatic change propagation carried out

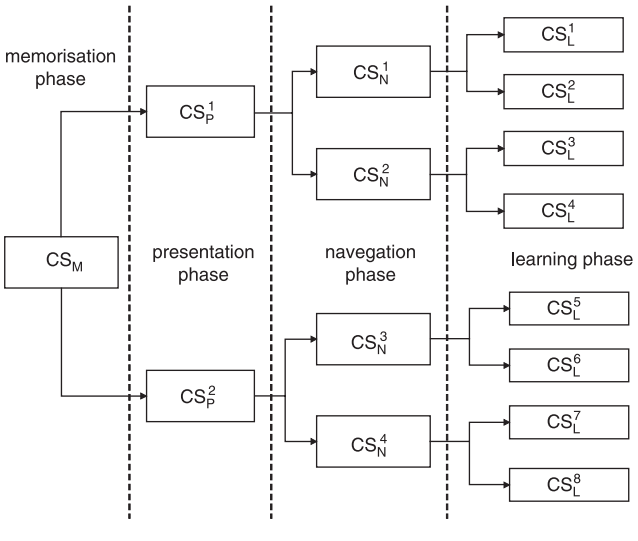


Fig. 6 *Eight different CS_L*

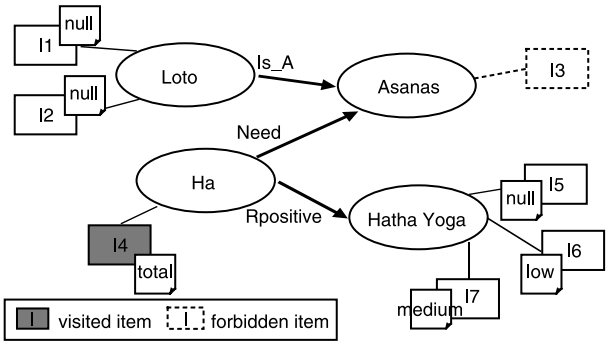


Fig. 7 *CS_L adapted to user's knowledge*

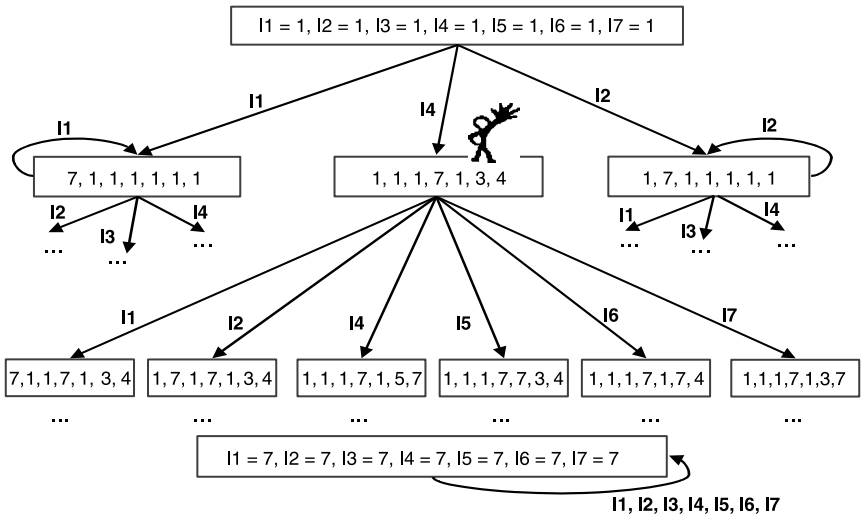


Fig. 8 *Navigation$_k$ tree*

by the metasystem. Every time the tree is generated, the system checks restrictions to guarantee that the set of R_u and R_k is consistent. For example, the presence of leaf nodes representing 'total' knowledge ensures that this knowledge state is reachable by the user.

4.3 CS_L personalised to user interests

During the choice of the CS_L, the system takes into account the knowledge subdomain in which the user is interested, contemplating to some extent his interests and wishes. Nevertheless, inside a knowledge subdomain there can be many items, and the user will probably be more interested in a group of them.

Because of this, the user can specify his interests as the set of items that he is most interested in learning (interesting items). Hence, during navigation, the system will annotate in a special way the items of the CS_L that will help the user to achieve his interests (desirable items). An item is desirable by the user if it satisfies two properties: (i) it is accessible; and (ii) it is interesting or its visit will directly or indirectly contribute to turn an interesting item into accessible. Therefore, when visiting a desirable item the user knows that he is visiting an item he defined as interesting, or visiting an item that will bring him closer to the knowledge state required to be able to visit one or several of his interesting items.

If all the interesting items are accessible, the set of desirable items matches the set of interesting items. If not, to calculate the set of desirable items, the system builds a tree with the knowledge restrictions that are not satisfied by the user and are nevertheless necessary for him to be able to visit the interesting items. This tree is called $Tree_D$ and its leaf nodes represent desirable items. The $Tree_D$ is automatically generated using the R_k represented as $Tree_K$. In the first level of the $Tree_D$, the interesting items specified by the user are joined by an *and* connector. Until all the leaf items are marked as desirable, we proceed as follows: If a leaf item is accessible it is marked as desirable; if not, it is replaced by its $Tree_K$, removing the branches representing knowledge requirements that the user's current knowledge state satisfies.

In Fig. 9*a* we show the $Tree_D$ that is generated for the interests set $Ints = \{I3, I7\}$ being the current user knowledge marked in Fig. 8. We can see how the item $I3$, which is not currently accessible, is expanded using its $Tree_K$ (Fig. 5). Only the requisites not fulfilled by the user are included, so the branch corresponding to the $K(I4) >$ 'high' requisite is not added. Figure 9*b* shows the CS_L adapted to the user interests by annotating the desirable items (the name of the item is underlined and in a larger font).

Obviously, until all the interesting items are marked as desirable, the $Tree_D$ is updated each time the user visits an item and its R_u is run. The knowledge restrictions fulfilled by the new knowledge state are removed, and if a node becomes accessible, it is marked as desirable and the subtree under it is removed.

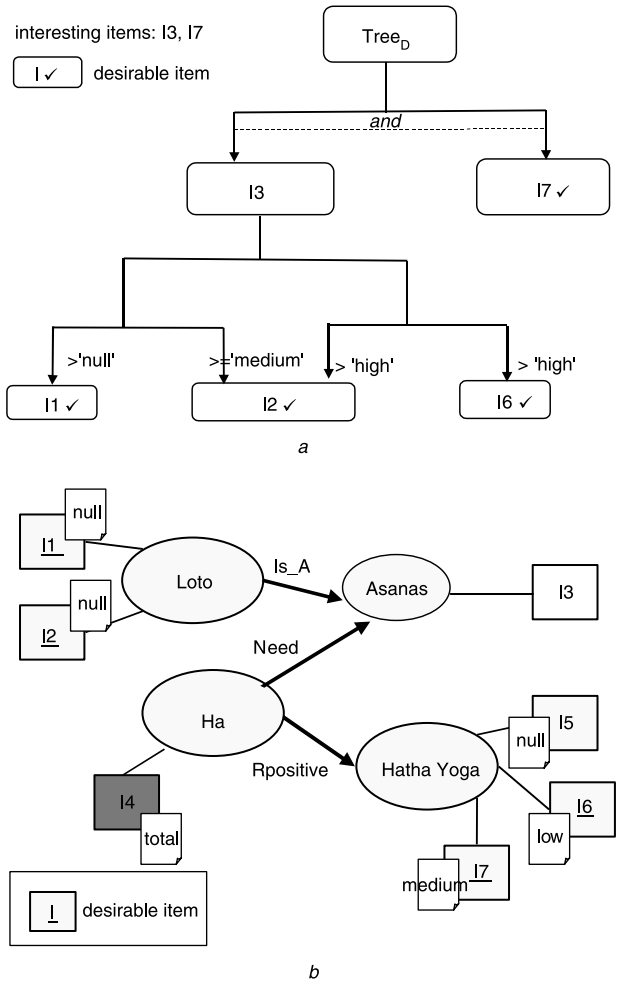


Fig. 9 *Tree$_D$ for $Ints = \{I3, I7\}$ and annotation of desirable items*

The user can also specify a knowledge goal, indicating not only the items he wishes to know but also the degree of knowledge he wants to achieve in each of them. The user will specify his goal as a set of subgoals in the form (I, $label_{SEM}$), where I is an item in the CS_L he browses and $label_{SEM}$ represents the desired knowledge degree. To personalise the CS_L in the function of the user goal, the procedure explained previously is used. The only difference is that, in this case, the branches that get to the nodes placed in the first level on the $Tree_D$ (the items included in the goal) are labelled with the knowledge restriction specified in the corresponding subgoal. Now, the knowledge restrictions in the first level are also evaluated. In this way, every time a subgoal is satisfied it is removed from the tree. When the root node of the $Tree_D$ is removed it will mean that the user goal has been completely achieved.

In addition, the user can request from the system a navigation route that brings him to his goal. This route is shown as a list of items and an order to visit them. To calculate it the system uses the $Navigation_K$ Tree (Section 4.2), taking into account the knowledge state of the user and his preferences on the items' idiom, length of the route, etc. The details of this technique are given in [8].

5 Conclusions and related work

SEM-HP is a systemic, semantic and evolutionary model that permits the development of adaptive and evolutionary HS. The architecture of the developed systems is divided into four subsystems, with two abstraction levels in each subsystem: the higher level (metasystem) provides the necessary mechanisms to evolve the structure and the functioning of the representation models included in the lower level (system). Among the four subsystems, the learning subsystem is in charge of performing the user adaptation and to do this, it defines, among other mechanisms, a set of rules that calculate the user knowledge (R_u) and use it to restrict the user's navigation (R_k). Three types of adaptive methods are applied during the user navigation: orientation support, personalised views and guidance. Table 6 shows the adaptive techniques used to implement each one of them.

Since the first adaptive hypermedia systems (AHSs) appeared in the late 1980s, many authors have contemplated navigation support methods in their systems, models and architectures. The particularity of the SEM-HP model lies mainly in that it offers to the author support during the process of evolution of the developed AHS, ensuring an easy, flexible and consistent maintenance task.

In addition, SEM-HP integrates an important number of navigation adaptation methods described by Brusilovsky in [9], while most of the AHSs we have found focus only on some of them. The way in which SEM-HP implements each adaptive method is also different. For example, while most adaptive architectures place on the navigational structures a method that provides orientation support (for example, abstracts views in AHAM [10]), in SEM-HP the orientation support is inherent to the navigational structure itself, a separate method not being necessary.

Another differentiating aspect of SEM-HP is that the navigation structure displays, in addition to the items of information, the concepts they describe and the relations between them. Although it is true that other systems, such as the one proposed in [11], also incorporate concepts, the difference lies in the homogeneous way SEM-HP treats them. For example, the system is able to determine the knowledge the user has about the concepts using a set of weight rules (R_w) [12] previously defined by the author.

Table 6: Adaptive techniques and methods

Adaptive method		Adaptive technique
Orientation support		Annotation of visited items
		Conceptual structure showing the position of the last visited item and its context in the information/conceptual domain
Personalised views		Choice of the navigation structure according to the subdomain of interest, navigation experience and subject experience
		Annotation of degree of user knowledge about the items
		Hiding the forbidden items
		Annotation of interesting items
		Annotation of goal items
Guidance	Global	Guidance for a knowledge state taking into account user preferences
	Local	Annotation of interesting items and goal items
To the user group		Transition matrices

Each R_w calculates the user knowledge about a concept considering the user knowledge about the items associated with it. It permits using knowledge about concepts in the definition of R_k, user interests, etc.

Regarding the adaptation to the group of users, we follow an approach similar to the one used by Bollen and Heylighen [13], although with some differences. While Bollen and Heylighen apply a set of learning rules (frequency, transitivity and symmetry) to automatically strengthen or to create new hyperlinks, we build transition matrices [14], which reflect the number of times the user goes from one concept to another in each navigation

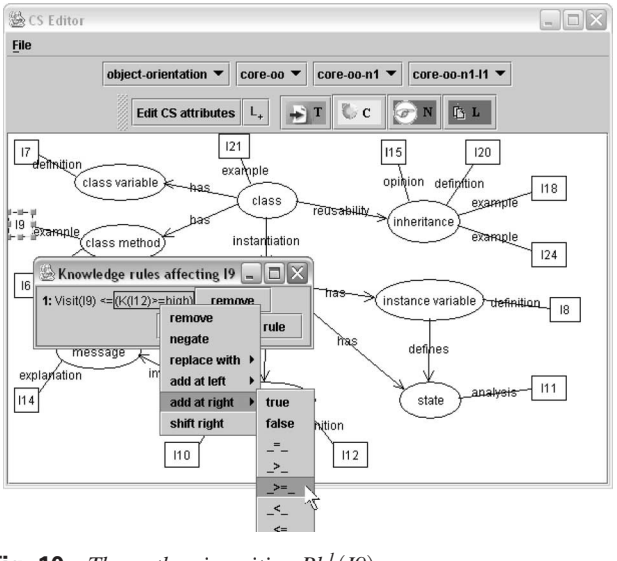


Fig. 10 *The author is writing $Rk^1(I9)$*

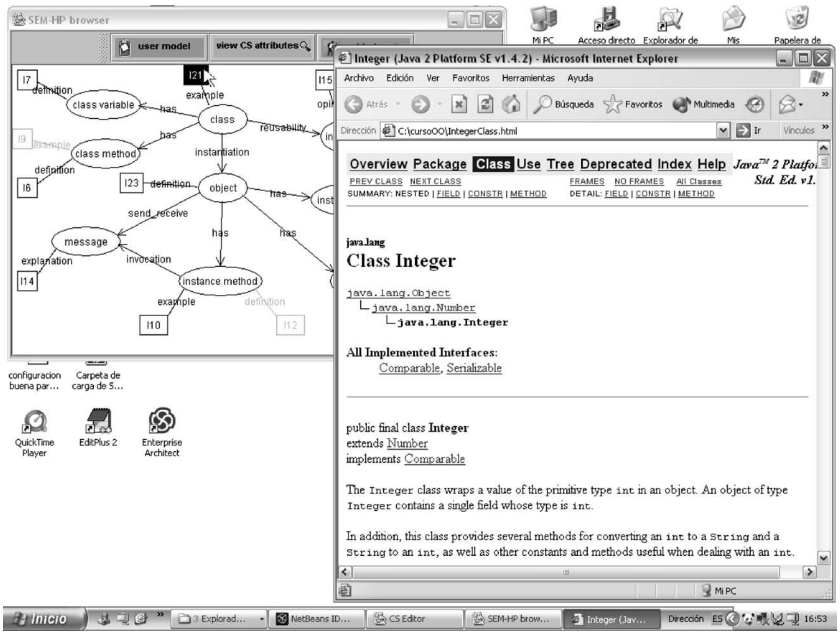


Fig. 11 *The user has selected I21*

structure [15]. Based on a matrix, the author will decide if the corresponding structure will be totally or partially adapted to the navigation pattern defined by its users. For example, he can remove a conceptual relation never used or add a relation frequently followed.

6 Further work: prototype JSEM-HP

Our further work is focused on finishing the development of a prototype that implements SEM-HP. The prototype in development is called JSEM-HP, and it is written in Java. Firstly, we are developing the author tool that will allow the developer to build and evolve adaptive hypermedia systems. JSEM-HP explicitly supports the layered division in systems and metasystems, and for now it implements the memorisation, presentation and navigation subsystems and part of the learning subsystem. For the visual manipulation of graphs it uses the Jgraph library [16]. Figure 10 shows the prototype interface when the author is editing the first knowledge rule associated with the item *I*9 in a conceptual structure whose knowledge domain is the object orientation paradigm.

Figure 11 shows the prototype interface during the user navigation – the moment in which the user has selected the accessible item *I*21. The degree of knowledge about the items is reflected through the colour of the item (darker means higher knowledge).

More details about the existing versions of the prototype are described in [14, 17]. Our aim is to use the prototype to validate the SEM-HP model, mainly in two aspects:

- *Development of AHS*: We are checking that our evolutionary approach actually helps the author to build AHS in a flexible and consistent way, i.e. the four subsystems keep consistent during their evolution, and the evolutionary actions and the restrictions are the appropriate and flexible enough to allow the author to develop and evolve AHS easily.
- *User adaptation*: Our objective is to check the different adaptation techniques implemented and verify that they enhance the navigation for heterogeneous users, improving the comprehension of the offered material and reducing the disorientation problems.

7 References

1 García, L., and Parets, J.: 'A cognitive model for adaptive hypermedia systems'. 1st Int. Conf. on WISE, Workshop on World Wide Web Semantics, Hong-Kong, China, June 2000, pp. 29–33
2 Wang, W., and Rada, R.: 'Structured hypertext with domain semantics', *ACM Trans. Inf. Syst.*, 1998, **16**, (4), pp. 372–412
3 Stotts, P., Furuta, R., and Ruiz, C.: 'Hyperdocuments as automata: verification of trace-based browsing properties by model checking', *ACM Trans. Inf. Syst.*, 1998, **16**, (1), pp. 1–30
4 Medina, N., García, L., Torres, J., and Parets, J.: 'Evolution in adaptive hipermedia systems'. Principles of Software Evolution IWPSE. Orlando, Florida, USA. May 2002, pp: 34–38.
5 García, L., Rodríguez, M.J., and Parets, J.: 'Evolving hypermedia systems: a layered software architecture', *J. Softw. Maint. Evol. Res. Pract.*, 2002, **14**, (5), pp. 389–405
6 García, L., Rodríguez, M.J., and Parets, J.: 'Formal foundations for the evolution of hypermedia systems'. 5th Eur. Conf. on Software Maintenance and Reengineering, Workshop on FFSE, IEEE Press, Lisbon, Portugal, March 2001, pp. 5–12
7 Wadge, W.W., and Schraefel, M.C.: 'A complementary approach for adaptive and adaptable hypermedia: intensional hypertext'. 12th ACM Conf. on Hypertext and Hypermedia, 3rd Workshop on Adaptive Hypertext and Hypermedia, Aarhus, Denmark, August 2001, pp. 327–334
8 Medina, N., Molina, F., and García, L.: 'Personalized guided routes in an adaptive evolutionary hypermedia system', *Lect. Notes Comput. Sci.*, 2003, **2809**, pp. 196–207
9 Brusilovsky, P.: 'Methods and techniques of adaptive hypermedia', *User Model. User-Adapt. Interact.*, 1996, **6**, pp. 87–129
10 Wu, H., and De Bra, P.: 'Link-independent navigation support in web-based adaptive hypermedia'. Web-Engineering Track of the 11th Int. WWW Conf, Honolulu, Hi, USA, May 2002, pp. 74–89
11 Da Silva, P., Durm, R., Duval, E., and Olivié, H.: 'Concepts and documents for adaptive educational hypermedia: a model and a prototype'. 2nd Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, USA, June 1998, pp. 35–43
12 Medina, N., García, L., Rodríguez, M.J., and Parets, J.: 'Adaptation in an evolutionary hyperpedia system: using semantic and Petri nets', *Lect. Notes Comput. Sci.*, 2002, **2347**, pp. 284–295
13 Bollen, J., and Heylighen, F.: 'A system to restructure hypertext networks into valid user models', *New Rev. HyperMed. Multimed.*, 1998, **4**, pp. 189–213
14 Medina, N.: 'An integral and evolutionary model of adaptation for hypermedia systems. The learning system of SEM-HP' (in Spanish). Doctoral thesis, University of Granada, Spain, November 2004
15 Salmerón, L., Cañas, J., Gea, M., Fajardo, I., Antolí A., and Abascal J.: 'Analysis of the knowledge adquisition in hypertext systems from the user's navigation strategies'. (in Spanish). COLINE, Granada, Spain, November 2002
16 Jgraph Swing Component. http://www.jgraph.com
17 Molina, F., García, L., Medina, N., and Hurtado, MV.: 'Evolutionary conceptual structure editor: practical considerations'. (in Spanish). 3rd Jornadas de trabajo DOLMEN JISBD, El Escorial, Madrid, Spain, November 2002, pp: 77–82