Hyperstories: A Model to Specify and Design Interactive Educational Stories*

Jaime Sánchez, PhD. & MauricioLumbreras, MSc.
Department of Computer Science
University of Chile
Blanco Encalada 2120, Santiago
CHILE
jsanchez@dcc.uchile.cl
mlumbrer@dcc.uchile.cl

***** Paper to be presented in the XVII International Conference of the Chilean Computer Science Society, November 13 - 15, 1997, Valparaíso, Chile.
* also LIFIA - Fac. Cs.Exactas - Universidad Nacional de La Plata - Argentina

ABSTRACT

We introduce a conceptual model for building interactive hypermedia literary stories. Hyperstories extend and plasticize the idea of «branching games» and the classic «choose your own adventure» stories. The model combines static and dynamic objects embedded in nested contexts to allow flexible navigation through virtual worlds. Hyperstories include the concept of object migration between nodes, underexplored in hypermedia environments. As a result, the model supports objects manipulated by the learner, autonomous objects, characters representing entities that behave independently from the user, and a clear separation between content and interface representation. We discuss different aspects involved in the implementation of hyperstories. Finally, we analyze some further trends and issues in this growing line of research.

1. Introduction

Conventional models of educational software engineering are based on the tutorial mode of instruction by focusing on book-like presentation design guidelines, stemming from a cycle of presentation of a learning content-questioning-presentation-questioning [1,6]. This model is seen as *presentation of information/knowledge* [28]. Even though this model is somehow useful, action, control, rhythm, and interaction are given by the software demanding a very passive user. Using this type of software has been described as «clicking arrows back and forth «. As a result, no added value is included in the software besides presenting information more dynamically and attractively.

A slight variation of this type of software is the model for *representing information/knowledge*. The idea here is to mimic the human mind by presenting information/knowledge in a way to match with a given model of human memory. Some examples of this software are hypermedia educational software that include concept maps or semantic networks for designing and structuring content, navigating, and even evaluating user's performance [27].

A much more flexible and learner-centered model of educational software is the type of software to *construct and reconstruct knowledge*. This is mainly recognized by the fact that users have to make things, construct, reconstruct, resolve, create, correct, build from errors, etc. Educational games, stories, comics, editors, and some developing systems are illustrations of this type of software. They incorporate key cognitive strategies motivating and fully involving learners by giving them control over the learning task, challenging, engaging, interacting, and adapting to the player's level, ranging from beginning to advanced ones [1,2,10,11,19,30,31,36]

This study introduces a conceptual model for developing educational software to help learners to construct and reconstruct knowledge. A model for building interactive hypermedia literary stories. We name this *hyperstories* [33,34]. Stories are narratives of true or fiction events that intend to capture and involve learners actively. Hyperstories extend and plasticize the idea of «branching games» and the classic «choose your own adventure» stories. The model combines static and dynamic nested contexts to allow flexible navigation through virtual worlds. Thus supporting of objects manipulated by the learner, autonomous objects, characters representing entities that perform independently from the user, and a clear separation between content representation and interface management.

The literature on the design and specification of hypermedia applications focus primarily on navigational and structure design [38]. It is uncommon that current methodologies put emphasis on traces of interaction between objects in the application through the time. They often take into account only media synchronization and similar ideas. On the contrary, in hyperstories it is critical at the semantic level to capture the trace of events (giving the plot of the hyperstory) and the interaction between objects (giving the roles). As a consequence, we believe that it is desirable to have a well grounded methodology to design this type of applications. The scope of this paper is to describe the specification model for hyperstories and to outline the major steps involved in the design of hyperstories. One of the added value of the proposed specification is a compiler designed in order to run hyperstories written within the framework of this model.

Finally, the target focus of hyperstories is cognitive development. Hyperstories were built to enhance the development of cognitive structures that determine tempo-spatial relationships and laterality in early age children. These intellectual structures include learner consciousness of his own position in time and space, the position of others, the position of others in relation to them, the position of the things and objects that surround him, time duration, succession, and the recognition of his and other's left and right [34]. Hyperstories are mainly based on spatial navigation and time-events, presenting a strong metaphor to deliver these ideas.

2. What is a Hyperstory?

There are several disciplines, computer environments, and applications that come together to the concept of hyperstory. One type of such environments are MUDs (Multi-User Dungeons) and their variations (MOOs, etc.). In the original version, these text-based systems allow many users to connect simultaneously to virtual «worlds» composed by rooms, objects, and people. Depending upon the design of a particular system, themes vary from fantasy environments with dragons and wizards, to futuristic exploration with spaceships and aliens.

Our model extends these ideas by including the elements of a story. These elements are: plot, roles and character. The main idea is to capture these elements in the representation [17]. Plot is a temporal sequence of actions involving a set of individuals. A plot and its constituent actions may be quite abstract. i.e.: A meets B, A loves B, A loses B, A wins B. Role is a class of individuals, whose prototypical behaviors, relationships, and interactions are known by both actors and audience. For example, the plot outlined above ordinarily is instantiated with alternative roles, for example: the boy in love and the girl he loves. Character is a personality defined as a coherent configuration of a psychological trait. For example: any of the characters in the present scenario might be: shy and sensitive, silly and affectionate.

We start the construction of the concept of Hyperstory by first introducing the definition of a Hypermedia Virtual Environment (HVE) as:

$$HVE := hypermedia_{I} + dynamic\ objects_{II} + characters_{III} \quad (eq.1)$$

Where:

I. Hypermedia is:

- In charge of modeling the virtual world composed by several navigable environments connected between them by links. This is a special case of hypertext, in which each node basically represents a container of objects and a potential scenario of the hyperstory. The connection is rendered by physical gates, portals and doors which are represented as links. Thus hypermedia models the spatial relationship and connection of environments. The concept of associated hypertext as underlying model to describe spatial navigable metaphors is discussed in [9]

  - A modeling technique to provide basically the branching in the course of the story. But while this definition might be sufficient, it fails to convey a significant semantic aspect of the structures embedded in the hyperstories. It is also complex to model interaction patterns among entities by using just the node-link model.

II. Dynamic Objects are:

  - In charge to represent the objects of the virtual world. They are entities that have behavior in time and react to the events produced by the user and other entities.

III. Characters are:

  - The entities that carry on the main course of the events and have a very complex behavior.

There is a distinguished character that is called the protagonist, manipulated by the user and representing the connection of him with the system. If the protagonist is third-person viewed, an avatar will be in charge of this representation. In addition that characters are special cases of dynamic objects, they are very important to the story level. Characters represent the main plot and elicit the content of the story. For example, in a film the most interesting events happen to the characters and they develop the actions that emotionally impact to the audience.

Up to this point a MUD or an adventure computer game fall in the scope of HVE. We then envision a Hyperstory (HS) as an extension of this concept. A Hyperstory is composed of:

HS = HVE + narrative (eq. 2)

Our model extends the idea of HVE by introducing the idea of narrative, including an intentional sequence of events, based on plot, roles and characters. Other differences between HS and MUDs arise from the idea of closure or explicit final, described as a good feature in narrative [23]. Thus hyperstories are intentional in a greater degree rather than a casual scenario. Additionally, the plot in a hyperstory is not linear, is a hyper-plot. Then action, object activation and dialog can trigger changes in the flow of the story. To do this, we borrow ideas from hypertext/hypermedia technology, by including narrative in a hypermedia context [4].

2.1 Hyperstories and branching games

One form of interactivity that is underlying similar to HS is referred as «branching games», first popularized in «choose your own adventure» stories. In these games, the player experiences short, linear story segments. At the end of each segment are a small number of (say, two to four) choices, each leading to a new linear segment, which leads to further choices, and so on. Sometimes the pathways converge, other times they diverge to different endings. As Joiner points out [18], the advantage of a branching game is that allows to write a small (or at least finite) number of alternate story paths. The disadvantage of this technique, however, is that the full power of interactivity is diminished. The player can only choose paths that have been anticipated by the designer.

Hyperstories grow from this concept by introducing the idea of opportunity. Real world stories have an ample variety of opportunities. These chances occur throughout the time. Characters may decide whether or not to take an opportunity. This concept was previously explored in [35] by presenting the powerful idea of temporal link, available only in some time window. We move forward by including in the behavior of entities some rules that can be triggered for a given period of time without being aware of the beginning of the time window. For example, a treasure map flying freely in several environments can be found only if it is perceived by the character. If the character refuses to explore the map, the opportunity could be lost forever. As a result, even though the flight of the map was predefined in the authoring stage the encounter map-character is a non-deterministic scenario. This gives an unpredictable performance of the story by presenting a powerful extension of the concept of opportunity or a kind of floating link such as in a custom version of the Storyspace authoring environment [21]. We call this feature *blur link*.

In addition, hyperstories have improved conventional literary stories by allowing a «dynamic binding» between characters, the world where they move and the objects they act on [34]. This binding is performed by the learner, thus allowing a greater flexibility in the learning process. In other words, a hyperstory is a combination of a virtual world where the learner can navigate, a set of objects operated by the learner, and the pattern of interaction between entities [33].

A particular feature of a hyperstory is that two different learners may experience different views of the same virtual world. Slight changes introduced by the learner to the object's behavior can produce different hyperstories in the same world. Learners when manipulating a character, can also interact with other characters to solve a given problem. Familiar environments such as schools, neighborhoods, squares, parks, and supermarkets, can be interesting metaphors for building virtual worlds.

It is interesting to notice that conventional hypermedia authoring tools do not provide an adequate set of facilities for building hyperstories as we have conceptualized them. The static environments involved in a hyperstory -virtual world- can be simulated easily, but several aspects such as the behavior of dynamic objects and complex interactions between the protagonist and characters exceed the conventional «nodes and links» model, as seen in [12,14,24].

3. The model

Our model is Object Oriented, providing a framework in order to model diverse building blocks of a hyperstory. The model supplies a framework made up of three foundational classes as described in OOD techniques [32]. It is important to realize that each one of these classes has customized syntax and associated semantic. The classes are: context, link and entity. In addition to classes, there is an associated constructor called channel. Contexts model the static world and links model the connection between contexts. Entity is the abstract class that captures any object or character definition and channels work as a broadcast media of events in a fan-in or fan-out fashion to the subscribed entities.

Each base class has a predefined behavior and a set of attributes that differentiate from each other (e.g. a link knows about the movement of entities between contexts). Another example of specialized behavior arises from contexts: if an entity sends an event to a context, it sends the event to all contained objects. Thus a context works as a diffuser of events. All these base classes have certain behavior, based on a modal programming. Objectcharts is the formalism to specify it. To give rigorousness to our model, we adapt a semantic borrowed from the StateMate specification [16].

3.1 The state-based behavior specification paradigm

A hyperstory is based on the idea of a mixture of reactive objects and an intentional plot mapped to predefined behaviors, extending the concepts of [37]. But reactive systems owe much of their complexity to the intricate nature of the reactions. From these systems arise the notion of reactive behavior, whereby the system is not adequately described by specifying the output that results from a set of inputs. Rather, it requires to specify the relationship of inputs and outputs throughout the time [15]. Typically, such descriptions involve complex sequences of events, actions, conditions and information flow. They have often explicit timing constrains that combine to form the system's overall behavior. To deal with this problem we found that the standard structured analysis and structured design methods do not adequately deal with the dynamics of a HS, since they were proposed to deal primarily with nonreactive, data-driven applications. Moreover -and at a higher level of design- standard design methodologies for hypermedia applications such as OOHDM [38], do not convey dynamic behavior of objects that navigate in a hypermedia environment.

On the contrary, functional approaches fail to model HS, because they do not specify dynamics: it is not clear when and why the activities are activated, whether or not they terminate on their own, and whether they can be carried out in parallel. In short, a functional view says virtually nothing about how the activities are done during the HS flow. For this reason, we start to model the behavior of hyperstories' entities based on some reactive specification language because the nature of a HS is basically animated by events and states. There are several languages to deal with this feature like Esterel [5] and others, but they do not include explicit foundational concepts associated with hyperstories such as navigation.

4. Main conceptual design building blocks

A hyperstory specification can be splitted in two interrelated conceptual parts by using the following classes:

- static scenarios (contexts and links),
- objects (entities) and the explicit routing mechanism (channel).

4.1 The static world

When a hyperstory involves several scenarios, they are organized according to their physical connection (linking). For this purpose, we can describe the virtual world as a kind of nested context model. A virtual world is defined as a set of contexts that represent different environments. Each context contains an internal state, a set of contained contexts, a set of objects, links to other contexts, and a specific behavior. Different relationships may be held between two different contexts, such as:

- neighborhood (there is a link from one context to the other),
- inclusion (one context is included in the other),
- none (contexts are «disjoints»).

The idea of a context is the same as in standard hypertext technology: a node or container. Different «real world» metaphors can be implemented easily with this simple model, such as a town, a house and a room (or houses within a town and rooms in a house). All these metaphors are designed in such a way that can be freely navigated. The main difference between our model and traditional hypermedia models is that nodes (contexts) may be nested, in some way like [7]. Another important concept about context is perception: a context is a spatial container that can be perceived as a whole rendered as a unity at the interface level. In this stage of modeling, context and link are used to build new navigational elements through the use of the inheritance mechanism. At this point of the designing, we are dealing with the first term of the *eq. 1*.

4.2 Populating the world

In order to bring life to the hyperstory, we populate the environment with objects, some active, some passive, orthogonally composed of a navigational dimension. To avoid misunderstandings we briefly define some terms related to objects in this context.

- **Passive**: the object answers only a simple events like «Who am I?»
- **Active**: the object has a noticeable behavior while the time progress -continuous or discrete- or they respond to events with some algorithm that reflects some behavior.
- **Static**: the object always belongs to the same context.
- **Dynamic**: the object can be carried to contexts by some entity or may travel autonomously.

Any object or character (even the protagonist) can be a subclass of an entity. Therefore we need to extend the basic attributes and behavior of an entity. Basically an entity can be viewed as an object that has a set of attributes that define an internal state and a behavior. The object behavior is specified by using a special made state-based script. In each state, there are a set of rules containing a triggering event, a pre-condition, and a list of actions that must be performed when the event arrives and the pre-condition holds.

		Navigational Dimension	
		Static	**Dynamic**
Behavioral Dimension	**Passive**	▪ a tree ▪ a table	▪ a key ▪ a fly
	Active	▪ a fire that grows ▪ a bank vault with a combination lock	▪ the character

Table 1. Characterization of some common objects in hyperstories according to navigational and behavioral characteristics

Each rule plays the role of a method in OOD jargon. But if we try to capture the nature of the narrative and the diverse branches of a hyperstory, the model must consider this. Certain entities in a story can respond to the same event (message in OOD jargon) in a different way according to the story stage. For example, according to the stages of the hyperstory a person can respond: «fine» or «tired» related to the question «How are you?». In short, an object can behave differently to the same message received in its life stage. This concept is called programming with modes [39] or state-based programming. To capture this feature the rules are not specified in a flat way, they are blocked and grouped according to the entity life stage. We use state-based scripts in order to deal with this feature. Therefore, for the same event there are different rules according to the state of the hyperstory. The post-condition embedded in each rule activates or de-activates these blocks of rules, managing the stages of the HS. At this stage, the behavior specification enables us to describe the nature of characters and objects. We map narrative in the behavior by means of dialog, dynamic behavior, blur links, etc. As a result of this, we satisfy the *eq. 2* .

We thought that the reuse of entities from previous HS by using inheritance would be an interesting feature to avoid repetitive work. But the more complex objects -characters- are very difficult to reuse due to their custom nature, very related to the original HS. Thus this feature only applies to simpler objects.

4.3 How to specify the behavior

Objectcharts is a visual formalism for describing the behavior of an object class as a state machine. They extend the idea of Statecharts [15], which are an improvement of the standard finite state machines. The Objectcharts incorporate the notion of hierarchy, orthogonality (concurrence), composition, a broadcast mechanism for communication between concurrent components, composition, aggregation and refinement of states. The Objectcharts provide an effective and concise notation for the specification and design of complex reactive systems [8].

The object behavior performs the hyperstory content and is critical to capture the narrative into objects. Because narrative is guided by behavior and changes in a reactive manner, we based the specification on the Objectcharts formalism. It fits well at the behavior level of the hyperstories' entity. Previously, Statecharts have been used to model hypermedia applications [40,41].

5. Writing and modeling a hyperstory

When we write a hyperstory, the first idea is to write a flowchart or state diagram [20] describing different paths and the main events that may modify the story. This is a naive method that can easily produce a monolithic piece of code, but is unable to specify temporal opportunities. It is also complex to maintain and debug. A more natural way is to think about the behavior of each object or character and to map the main events that can change the story to rules, such as writing a script for each drama actor. These rules describe the object's behavior. As a result, we have fragmented the whole story into each object. Therefore maintenance and debugging is easier, because the cause-effect rule is contained in the object´s behavior.

5.1 The specification language

The hyperstory specification language satisfy the followings requirements:

- A way to represent scenarios, connection among them, routing events and a way to specify changing behaviors according to different stages of the hyperstory.
- Isolation of the interface from the content of the story in such a way that at the moment of story specification we can avoid representation problems and deal exclusively with the content of the hyperstory.
- Independence between the specification and the implementation language.
- Ability to embed objects with dynamic and some kind of autonomous behavior. This characteristic enables us to create «virtual creatures» or robots that perform activities by themselves

- Synchronous and asynchronous communication among entities in order to simulate concurrence

This language tries to be clear and intuitive, and thus amenable to be generated, inspected and modified by humans, as well as precise, and capable of validation and execution. Moreover, our language is an executable specification.

As a consequence, this model allows the use of the same virtual world for different learning purposes. (e.g.: modifying the interface -not the content-, you can build a hyperstory presenting all the information represented by sound to build a hyperstory for a blind child [26]).

5.2 Modeling stages

At the beginning of our work we thought that describing isolated behavior of objects could directly generate a hyperstory. But when we tested some prototypes with object interactions without an embedded narrative, the hyperstory appeared somehow misleading because objects alone only generate useless interactions without generating story.

Step-by-step guideline	Explanation
Write the literary story	Conceptualize the theme of the story and the topics involved.
Craft the skeleton of the hyperstory	This phase defines the soul of the narrative. Here the main story nodes are designed and the events that trigger the navigation in the story space are specified. Character features and their behavior are critical aspects for the harmony of the narrative. The structure of the behavior of each entity appears at this stage.
Describe the static world	It comes from the hyperstory essence and the environment description. We deal with context and link as foundational classes to model the spatial navigational structure.
Construct the character-dialog interaction	It is necessary to display motivation, emotion, and personality. This can end up with lovable or detestable characters that can be liked or hated during a performance. This phase increase the number of rules -methods- embedded in each entity.
Enrich the story	By aggregating some characters with autonomous navigational behavior, the story can be enriched by providing a scheme for non-determinism
Split the whole	We partition the monolithic hyperstory into much more manageable atomic objects by extracting it from the skeleton. At this point a graphical specification arises by using the Objectcharts formalism. In that way, we describe the interactions and timing in an object oriented fashion.
Compile the Objectchart specification	We have built a special compiler that interprets a textual Objectcharts-based specification and makes an executable kernel. This module interacts with some interface through a well established protocol. The interface will render the final look of the hyperstory and can be implemented in any environment capable of understanding MS Windows messages.

Table. 2. Issues identified in the writing of the content for a hyperstory.

6. Implementing Hyperstories

The specification model captures the hyperstory independently from the machine leaving a mechanism precisely defined to map this specification to the target executing machine. A small excerpt of a hyperstory displayed below intends to show the main features of the specification language. This example does not try to show how dialog and higher level narrative are mapped. It only illustrates the underlying computational elements of a given hyperstory.

6.1. Illustration of the Model

Imagine a killer robot originally placed on context A traveling each *n* units of time to the neighbor context in clockwise direction as seen in figure 2. When the robot finds the character tries to shot the character. The character -originally placed on context B- can protect himself by taking the shield located in context C. Thus the shots of the robot do not make any effect on the character.

Below, the related code shows how the specification is directly executable by using our compiler. The example does not show formally most of the main characteristics of the language, but it intends to be an intuitive example. Each rule is defined as: *event; pre-condition; list of actions; post-condition*. The post-condition is in charge of activate and de-activate states. Synchronous events are noted as *receiver<—event* and can return a value. Comments are between braces.

```
CLASS directedLink IS A LINK
.... definition of a directed link, is a link that enables to be crossed only in one direction
CLASS Robot IS AN ENTITY
ATTRIBUTES
... some attributes of the robot
BEHAVIOR
STATE global

PAR {allows parallel analysis in two contained substates: main and navigation }
STATE main
        (sys_init; true; TIMER<—setEvent('navigate',100,true);  NULL)
        { sys_init is an initialization event. Here we set the timer to generate a cyclical 'navigate'
event every 100 ticks }
        (entityin; sender<—are_you_a _person?; sender<—kill ;  NULL)
        { if a person enter to my context, I try to kill the character}
        (crossed; myContext<—is_there_a_person?;
 LOCAL aperson:hobject; aperson:=mycontext<—  getAPerson; aperson<—kill;null).
        { when crossing a link, I receive a 'crossed' event from the link. Then I ask to my new
context if there is a person, if it is true, I try to kill the character }
END { main}

STATE navigation
STATE A
(navigate; true; linkAtoB<—crossby(self);B)
        { Hardwired coding of the navigation between contexts.. Each time I receive an event
'navigate' I change my state and navigate to the neighbor context }
        {To navigate, I tell the adequate link I want to cross the next context. The last
parameter in the rule indicates the change of state}
END {A}
STATE B
```

```
(navigate; true; linkBtoC<—crossby(self);C)
END { B }
STATE C
(navigate; true; linkCtoA<—crossby(self);A)
END { C }
END { navigation}
 END { PAR }
END { global }

CLASS human IS an ENTITY
STATE main
(objTaked(obj);obj<—getname='shield';; IMMORTAL)
        { if at the interface level I take the shield, then I will get       immortal }
STATE living
(kill; true; self<—die;DEAD)
        { due to the nesting structure, this state is automatically activated with the previous one }
{ somewhere I need to model the state DEAD }
END { living }
STATE immortal
(kill; false; ;NULL)
        { redefinition of the behavior to attend the «kill» event.      Thus I never die}
END { inmortal }
END { main}

CLASS world is a CONTEXT
{ define my virtual world }
ATTRIBUTES
  aRobot :Robot
  theUser :Human
CONTEXTS
{ these are the containers of the objects }
   A,B,C ARE CONTEXTS.
LINKS
 { my virtual world has this type of links}
 AtoB,AtoC ARE directedLink
 AtoC IS A link
INITIALIZATION
 { linking the contexts between them }
  A LINKED TO B BY AtoB
  B LINKED TO C BY BtoC
  A LINKED TO C BY AtoC
 { loading the virtual world with entities}
 A INCLUDES aRobot
 B INCLUDES theUser
END { of the world definition }
```

We notice that there are two hyperspaces running simultaneously:

•• The static environment that supports the navigation, and
•• The story space.

In the story space there are links and nodes, but in the standard way, because one node can be conceptualized as a determined state of the instance variables, behavior of objects, and characters. The instance variables and behavior can change due to certain events generated by the user or some objects. A subset of these events can change the course of the story. In this case, a story-link has been activated.

6.2 System Architecture

Basically the system is composed of some way like [3,25] by:

- The kernel
- The interface manager

The kernel maps every interface object -such as characters, contexts, etc.- to an internal object The kernel's object can send messages to the equivalent interface object in the interface. The communication between internal and interface objects is made through a special protocol. The interface has the capability to receive and show the incoming messages sent by the kernel. Furthermore, the interface manager captures the user activity and dispatches it to the kernel.

As stated previously, conventional hypermedia authoring tools reflect a lack of facilities to implement the requirements of hyperstories. Our architecture solves this drawback because timing and complex object behaviors (managed by the kernel) are implemented by using an adequate language that manages this constraint. In addition, there are several tools that provide an easy user interface management. Therefore we made the system to profit from the best tools available. Different interfaces and networked hyperstories can be easily implemented with this global design.

A hyperstory is finally made as follows: a cross-compiler generate Pascal code from the hyperstory code defined as showed in section 6. The Pascal code is again compiled and linked to build the executable kernel to be run under MS Windows 3.X. By using some multimedia authoring environments -such as Asymetrix Toolbook and Borland Delphi- or a plain textual interface, we can run the system obtaining the final performance of the Hyperstory. The interface and the kernel link the internal and interface objects through a configuration file generated at the compilation time.

7. Final Discussion

We have introduced Hyperstories as a new concept for producing interactive software. Our idea is supported by a model for building highly interactive stories and a language to satisfy complex requirements of Hyperstories. As a result, a new way for producing interactive educational software is delineated. Actually, we believe that our proposal is an alternative metaphor to produce software that involves the learner more deeply, thus giving control, adaptability, constructability, and plasticity, therefore, a truly rich interaction. Furthermore, we have extended key attributes of branching games and MUDs with an intentional non-linear narrative embedded in a hypermedia virtual environment.

Our model covers most requirements to engineer quality OOD software for learning purposes. We believe that in the way presented here our concept can be standardized to enrich the area of educational software engineering.

Hyperstories give learners control over stories, access to diverse tools, and materials to construct with. These features may help to develop strategies and abilities to test hypothesis with the implicit idea of fostering the development and use of tempo- spatial relationships and laterality. As as consequence, our model for building Hyperstories supports complex user requirements in order to assist constructivist learning.

Diverse efforts have to be made to improve our concept. We are now working on several directions to enrich our model. First, we are testing Hyperstory prototypes supporting most of the features at the kernel level and a sub-set at the interface level. We intend to define different user-centered interface styles in order to adapt Hyperstories to learners with different communication access requirements. In addition, our research group is studying the modeling and implementation of collaborative hyperstories, ludic hyperstories, web-based hyperstories, and hyperstories for blind children.

Finally, the main belief behind the concept of Hyperstories is that they can contribute to make the interaction with computers much more enjoyable and learnable to learners. Children like stories and get involved in them easily. If we add non linearity, navigability, interactivity, flexibility and plasticity, we end up with an interesting software to learn and to construct with.

8. Acknowledgment

This work was partially supported by the Chilean Science and Technology Fund (FONDECYT) grant No. 1950584 and CYTED Project VII-8. We would like to thank to Guillermo Capelli for his work with the programming.

References

[1] S. Allesi , S. Trollip. *Computer-based instruction: Methods and development*. New Jersey: Prentice Hall Inc., 1991

[2] S. Alessi. The design of educational software: state of the art. *Proceeding of the Educational Software Workshop,* Valdivia, Chile, pp.1-18, 1996..

[3] P. Appino, J. Lewis, L. Koved et al. An Architecture for Virtual Worlds. *Presence*, Volume 1, Number 1, Winter 1992, MIT Journals, 1992.

[4] M. Bernstein. Conversation with friends: hypertext with characters. *Lectures on Computing,* Springer Verlag, forthcoming.

[5] G. Berry, G. Gonthier. The Esterel Synchronous Programming Language:Design, Semantics, Implementation. *Science of Computer Programming*, Vol 19 #2: 87-152, 1992.

[6] A. Bork .*Personal computer for education.* New York: Harper & Row Publisher, 1985.

[7] M. Casanova, L. Tucherman, J.L. Neto,
N. Rodriguez, L. Soares. The nested context model for hyperdocuments, *Proceedings of Hypertext '91*, 1991.

[8] D. Coleman, F. Hayes, S. Bear. Introducing object charts or how to use state charts in object-oriented design. *IEEE Transactions on Software Engineering*, Vol.18, No.1: 9-18, 1992.

[9] A. Dieberger. Browsing the WWW by interacting with a textual virtual environment - A framework for experimenting with navigational metaphors. *Proceedings of Hypertext '96*, pp.170-179, 1996.

[10] A. Druin. A place called school. *Interactions*, January 1996, pp.17-22, 1996.

[11] A. Druin, C. Solomon. *Designing multimedia environment for children: computers, creativity and kids.* New York: John Wiley and Sons,1997.

[12] R. Furuta, D. Stotts. The Trellis Hypertext Reference Model. *Proceedings of the Hypertext Standarization Workshop, National Institute of Standards and Technology*, pp. 83-93, 1990.

[13] F. Garzotto, P. Paolini, D. Schwabe. HDM: A Model for the design of Hypertext applications. *Proceedings of Hypertext '91*, 1991.

[14]. F. Halasz, M. Schwartz. The Dexter Hypertext Reference Model. *Proceedings of the Hypertext Standarization Workshop,* National Institute of Standards and Technology, pp. 95-133, 1990.

[15] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, Vol 8: 231-274, 1987

[16] D. Harel, A. Naamad. The STATEMATE Semantics of Statecharts. *Technical report CS95-31*.The Weizmann Institute of Science. Available at http://www.wisdom.weizmann.ac.il/Papers/trs/CS95-31/abstract.html, 1995.

[17] B. Hayes-Roth, R. v. Gent, D. Huber. Acting in character. *Technical Report KSL-96-13*, Knowledge Systems Laboratory, March 1996.

[18] D. Joiner. Real Interactivity in Interactive Entertainment. *Computer Graphics*, Vol 28, number 2: 97-99, 1994.

[19] Y. Kafai, E. Solloway. Computational gifts for the barney generation. *Communications of the ACM*, 37(9): 19-22., 1994.

[20] M. Kelso, P. Weyhrauch, J. Bates. Dramatic Presence, *Presence*, Volume 2, Number 1, MIT journals, Winter 1993.

[21] R. Kendall. Hypertextual Dynamics in A Life Set for Two. *Proceedings of ACM Conference on Hypertext ´96*, pp. 74-83, 1996.

[22] H. Kenneth, M. Guzdial, S. Jackson, R. Boyle, E. Soloway. Students as multimedia composers. *Computers Education*, vol 23, 4: 301-317, 1994.

[23] G. Landow. *Hypertext: the convergence of contemporary critical theory and technology.* Baltimore: The John Hopkins University Press, 1992.

[24] D. Lange . A Formal Model of Hypertext. *Proceedings of the Hypertext Standarization Workshop,* National Institute of Standards and Technology, pp. 145-166, January 1990.

[25] J. Lewis, L. Koved, D. Ling. Dialogue structures for virtual worlds. *CHI'91 Conference Proceedings.* New Orleans, USA. pp. 131-136, 1991.

[26] M. Lumbreras, J. Sánchez, M. Barcia. A 3D sound hypermedial system for the blind. *Proceedings of the First European Conference on Disability, Virtual Reality and Associated Technologies,* pp. 187-191, Maidenhead, UK, 1996.

[27] J. McKendree, W. Reader, N. Hammon. The «homeopathic fallacy» in learning from hypertext, *Interactions*, pp.74-82, july1995.

[28] W. Nelson. Efforts to improve computer-based instruction: the role of knowledge representation and knowledge construction in hypermedia systems. *Computers in the schools,* 10(1-4) ,1994.

[29] M. Reed, S. Giessler. Prior computer-related experiences and hypermedia metacognition. Ed-Media 94. *Unpublished paper,* 1994.

[30] M. Resnick. Behavior construction kits. *Communications of the ACM*, July 1993, Vol. 36, 7:†66-71, 1993.

[31] M. Resnick, F. Martin, A. Bruckman. Computational construction kits. *Interactions* (3), 5:.40-50, 1996.

[32] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy , W. Lorensen. *Object-Oriented Modeling and Design.* Englewood Cliffs, NJ:Prentice Hall, 1991.

[33] J. Sánchez , A. Mallegas, L. Cernuzzi, G. Rossi, M. Lumbreras, A. Díaz, A conceptual framework for building hyperstories. *Ed-Media '94 Conference Proceedings*, p.761, 1994.

[34]. J. Sánchez, M. Lumbreras. Interfaces for learning. In Anzai, Y., Ogawa, K. And Mori, H(Editors). *Advances in Human/Factors: Human-Computer Interaction*., Symbiosis of Human and Artifact : Future Computing and Design for Human-Computer Interaction, 20A, pp. 865-870. New York : Elsevier Publishers, 1995..

[35] N. Sawhney, D. Balcom, I. Smith. HyperCafe: narrative and aesthetic properties of Hypervideo. *Proceedings of ACM Conference on Hypertext '96* pp. 1-10, 1996.

[36] R. Schank. Learning via multimedia computers. *Communications of the ACM,* May, 36(5): 54-56, 1993

[37] R. Schank, R. Abelson. *Scripts, Plans, Goals and Understanding*, Chapters 1-3:1-68, Hillsdale, NJ: Erlbaum, 1977.

[38] D. Schwabe, G. Rossi, S. Barbosa. Systematic Hypermedia Application Design with OOHDM. *Proceedings of Hypertext ´96* , Washington DC, pp. 116-128, 1996.

[39] A. Taivalsaari. Object-Oriented programming with modes. *Journal of Object Oriented Programming*, Vol. 6, No. 3: 25-32, June 1993.

[40] M. Turine , M. Ferreira de Oliveira, P. Masiero. A Naviga tion-Oriented Hypertext Model Based on Statecharts. *Proceedings of Hypertext '97*, pp. 102-111, 1997.

[41] Y. Zheng, M. Pong. Using Statecharts to Model Hypertext. *Proceedings of ACM Conference on Hypertext Technology*, Milan, pp.242-250, 1992.