The Knowledge plane interface

CHRISTIAN SIFAQUI
Department E-Business & Information Visualization
Zentrum für Graphische Datenverarbeitung e.V.
64283 Rundeturmstraße 6, Darmstadt
GERMANY

Abstract: - A new approach to human-computer interaction and user interfaces based on the concepts of simple and composite unities, is described. A simple unity can be seen as an attribute-based object. The prototype described here uses a three-dimensional graphical workspace for working with the *knowledge planes*. A k-plane is a set of semantically grouped unities and constraints on them. K-planes exhibit a comprehensible interface for both stand-alone and cooperative work.

Key-Words: - User interface, Human computer interaction, Autopoiesis theory, Attribute/value systems

1 Introduction

In a very general view, working with computers implies working with documents. A document will be defined a sequence of bytes that can be distinguished as a logical unit at some moment of time (for a more comprehensive definition of what a document can be, see [1]). Documents are edited, programmed, searched [2], filed, executed, sent to other people, revised and use as support for conversations [3], to name some forms of use.

On the other hand, performing a non-computationally task still leaves either physical or digital traces such as memos, contracts, reports, invoices, and so on. Afterwards, these documents are computationally reated and managed.

Computers confront us with a program-document interaction, where programs use document, for example, a text editor edits and stores textual documents. A program also lies on the file system as a special type of document until it is found by the user and then executed. In fact, both program and document are instances of files.

From the user point of view, in [2, 4, 5] different ways used by users to search for and find files on a personal computer are analized. In these articles the use of a concept like hierarchical files systems (hfs), that is not human based but machine based, is criticized, and it is shown also that all the user activity tries to avoid using hfs.

In summary, the problem lies on the semantical separation between program and document, and on the problems arisen from this gap, for example, the form of organizing them. In order to deal with this gap issue, a new conceptual framework is necessary, that allows to unify these concepts by bringing us a simple but effective approach.

In this paper an user interface that allows browsing, navigating and manipulating enhanced documents (later on called "unities") is presented. This interface is named Knowledge Plane (k-plane) and allows us to combine in it semantically similar unities.

The organization of the paper is as follows: section 2 is concerned with providing the approach that guides this new framework, section 3 presents the design and implementation of this framework and the interface, and some examples of the use of this interface are presented in section 4.

2 Approach

The von Neumann machine is a good example of how the idea about memory, originally used only for data storing and transformation, enabled us also to store the instructions on it. In the same way the object-oriented programming allowed us to combine in a concept –the object– data and functions that were two separated notions in the structured programming. Nowadays programs and documents are considered separated entities. For example, if I find a document in .doc format then I need a program that can process it. In this case the document's extension determines the allowed use on it (generally).

In [6, 7] a framework based on the ideas derived from the autopoiesis theory [8] is presented, where the concepts of unity, observer, organization and structure are fundamental to structure information with meta-information. This new framework deal indeed with joining semantically the document-program concept.

According to the autopoiesis theory, there are some definitions that I will use in this paper:

- **Observer**: "*living system who can make distinctions and specify that which he or she distinguishes as a unity, as an entity different from himself or herself that can be used for manipulations or descriptions in interactions with other observers*" [9].
- **Simple unity** "*is a unity brought forth in an*

operation of distinction that constitutes it as a whole by specifying its properties as a collection of dimensions of interactions in the medium in which it is distinguished" [10].

- **Composite unity** "*is a unity distinguished as a simple unity that through further operations of distinction is decomposed by the observer into components that through their composition would constitute the original simple unity in the domain in which it is distinguished*" [10].
- A composite unity has both **organization** and **structure**. "*Organization denotes those relations that must exist among the components of a system for it to be member of specific class. Structure denotes the components and relations that actually constitute a particular unity and make its organization real*"[8].
- **Property** "*is a characteristic of a unity specified and defined by an operation of distinction. Pointing to a property, therefore, always implies an observer*" [9].

Those previous definitions can be formally defined as:

a simple unity is constituted by an organization O and a structure S, so that

$$U_S := (O, S) \tag{1}$$

Likewise a composite unity also has its own organization and its own structure, in addition to the simple unities that it can contain, O_C is the organization and S_C the structure of the composite unity.

$$U_C := \bigcup_{i=1}^{n} U_{Si} \oplus (O_C, S_C) \tag{2}$$

The organization of a simple unity has a set of attributes A_S that define it, so

$$O(U_S) := A_S \tag{3}$$

Similarly, the organization of a composite unity is constituted of the organization of the simple unities that compose it, its own set of attributes A_C and the constraints C_C that affect the organization of the simples unities and itself

$$O(U_C) := \bigcup_{i=1}^{n} O(U_S)_i \oplus (A_C, C_C) \tag{4}$$

The structure of a simple unity is the set of values V_S of the attributes A_S at the time t, that is

$$S(U_S) := (V_S)^t \tag{5}$$

In the same way, the structure of a composite unity is the set of values V_C of the attributes A_S and the constraints C_C at the time t,

$$S(U_C) := \bigcup_{i=1}^{n} S(U_S)_i \oplus (V_C, C_C)^t \tag{6}$$

Fig.1 shows the structures defined previously in a graphic form. The unities that the observer can distinguish do not correspond necessarily 1:1 with the unities that reside in the model.

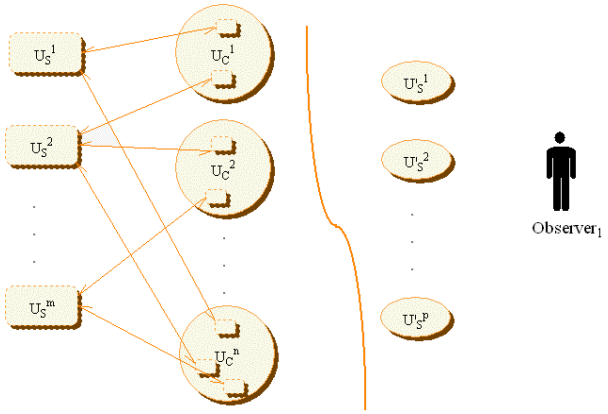

Fig.1: Unities, k-planes and the observed unities

Fig.2 shows an example that clarifies the previous structures. On the repository there are 3 simple unities, "seat", "back" and "leg", one observer with those unities has only defined the composite unity "chair", whereas the other observer defines two dependent composite unities, "stool" and this is part of "chair".

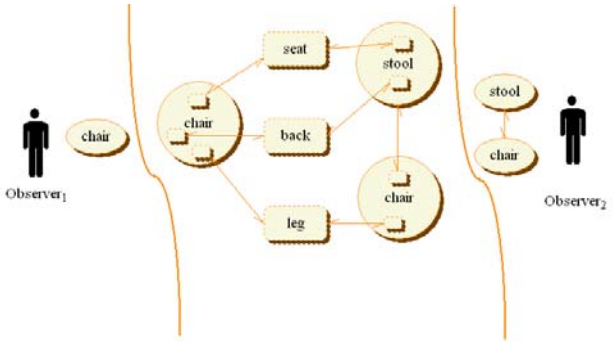

Fig.2: Simple and composed unities and the observed unities

3 Design and Implementation

The architecture of the system is shown at Fig.3.

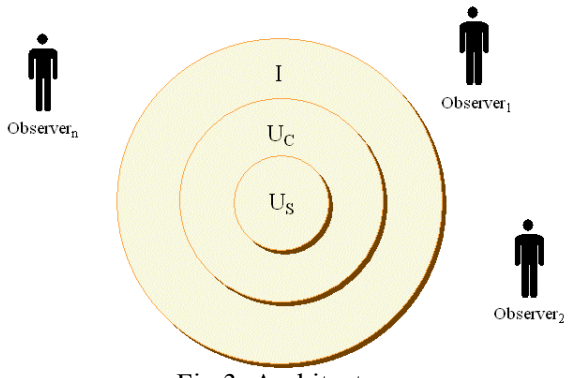

Fig.3: Architecture

This architecture is divided in three layers, the inner layer is a repository of simple unities. Following equations (1), (3) and (5) a simple unity can be realized through a list of attributes. A possible implementation would be to model a document through a file to which a list of attributes are added up as described in [11]. Another possibility would be a list of attributes where the content of the document is stored itself as an attribute, as proposed in [12].

The next layer stores the composite unities. Following equations (2), (4) and (6) a composite unity is the composition of simple unities adding up the constraints that make these units belong to the composite unity. The constraints are logical operators on attributes or functions that are executed on particular attributes.

The last layer deals with the interaction which is seen as interaction between observer and unities but also interaction between observers (for a more detailed discussion about interaction between observers see [7]).

The architecture allows the local gathering of unities to be distinguished by each observer and also a distributed access of shared unities.

I choose harland [13] as support for attributes and Java for the implementation of the prototype. Java offers the possibility of new classes being added up to the system while as it runs, through the adequate use of *Class.forName().newInstance()*. This is an essential issue to allow users to generate new composite unities through additional constraints. The new class must implements the *Kplane* interface.

Fig.4 shows a screenshot of the prototype, where the mostly used frames are shown. On the right side it is possible to see the k-planes interface and on the left side the interaction between observers.

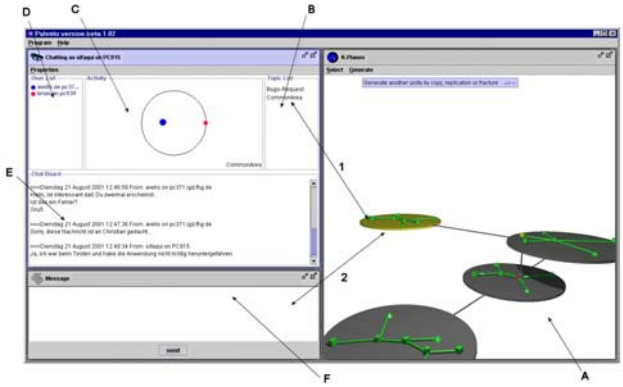

Fig.4: Snapshot of the protoype

On Fig.4A the visualization of the k-planes is shown. Fig.4D shows a list with the connected observers. Fig.4C shows the activity of them. The implementation supports direct manipulation, so communicating k-planes among observers is done by dragging and dropping them from a frame to another. Arrow 1 (from 4A to 4B) or arrow 2 (from 4A to 4F) shows how a unity can be transmitted. Here is a fundamental difference on the sharing, arrow 1 uses a unity as a basis for conversation whereas arrow 2 sends an attached unity as a part of the conversation.

3.1 K-Planes

K-planes are a 3D graphical representation of the composite unities. As previously noted, the k-planes (or composite unities) relate simple unities to each other and have constraints added. Thus, an observer can create a k-plane as a container of simple unities, with no constraints sharing common attributes. Many complex k-planes can have special constraints with regard to the organization of the unities, for example, according to a particular value of the structure of the unity some actions can be carried out.

K-planes are created on-the-fly when some search operation is carried out, looking at all the unities that have been modified in the last two days, for example. Then, a new k-plane containing the found unities is created.

The observers can define and program (in Java) their own k-planes and they can also be added to those already used by a client. K-planes can also be shared among observers.

A draft visualization of this representation is given in Fig.5, where the simple unities are represented by nodes on each k-plane. The navegation is carried out through movements in the hyperbolic space in a style similar to the proposed in [14]. In addition each k-plane can be rotated in its axis which allows a easy access to the unities that it stores.

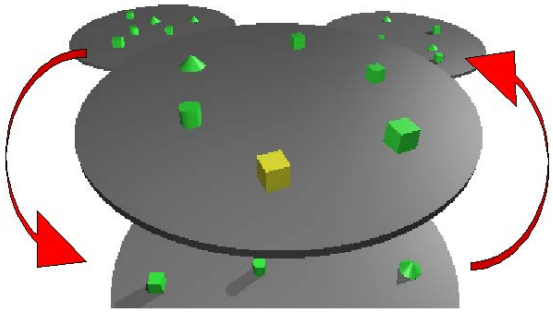

Fig.5: Knowledge planes.

4 Examples

Two examples have been chosen to show the use of k-planes that do different actions based on the attributes of the simple unities.

4.1 Use of the k-planes for e-mail

Nowadays, it is almost impossible to conceive using k-planes without having electronic mail facilities, since most of the tasks and information that daily arrives to us originates from it. In Fig.6 two k-planes are shown, the nearest shows all the unread incoming mail and the behind one contains all the mails. The first k-plane connects with a IMAP server and for each mail a unity is created and added to this k-plane. This initial configuration can be extended by the observer later, creating more k-planes that "filter" the unities according to the values of the structure of each unity.

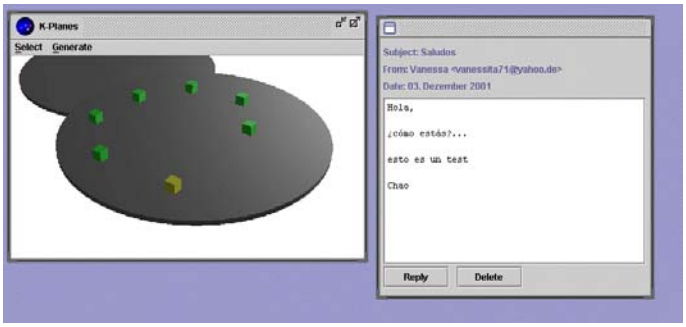

Fig.6: A k-plane showing the inbox mail with a spiral visualization of the incoming mails and a frame showing the structure of the selected unity.

The organization of these unities is composed of the attributes *from*, *subject*, *content*, *status*, *sentDate*, *receivedDate*, *size* and so on. When the structure of a unity has the attribute *status* with the value *recent*, then this unity is shown in the first k-plane. Once the observer reads the unity, the *status* changes to another value (possible values are *seen*, *answered*, *deleted* among others) and "disappears" from the first k-plane and "goes" to the k-plane of the mails.

In this last k-plane other search or refinements operations can be carried out or new k-planes can be created from this, for example, a k-plane can have the constraint that the attribute *from* must have a specific value and that the date must be ordered in ascending way, as shown in Fig.7.

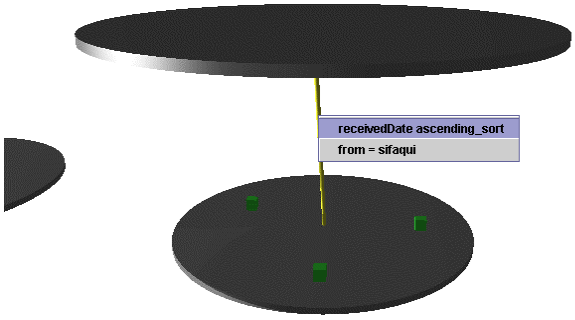

Fig.7: A K-plane showing a k-plane created from the email k-plane.

4.2 Use of the system as workflow

The k-planes can also be used to implement workflows. A workflow allows us to automate work steps, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

This workflow has been designed to support a process of internal review of articles. In order to improve the quality of papers, the authors use a review system where the paper is sent to a anoymous reviewer according to the areas of knowledge of this reviewer. In order to determine who should revise the paper, it is necessary that each unity (paper) has the key words as attributes in its organization.

For this workflow each observer has two k-planes, one for exit and another one for entrance. In the first one the articles are placed for their review. In the other k-plane articles from others authors arrive due to the key words corresponding to the area of knowledge of the observer.

Fig.8 shows the k-plane for a person who is waiting for the result of the reviewing process. In this plane, two unities are not been evaluated and the another unity is waiting for the author to read the comments.

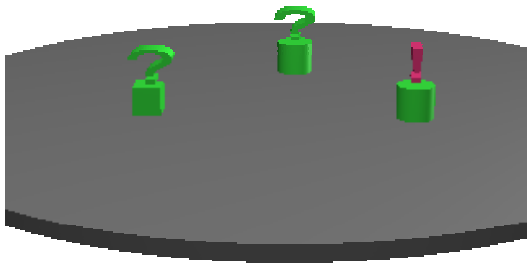

Fig.8: A K-plane showing a workflow

5 Conclusions

In this article a new framework for interacting with computers and an appropriate user interface is presented. As theoretical basis is used the autopoiesis theory. This theory provides the concepts for the new framework and a new human-computer interaction approach based on concepts from this theory is modelled.

The usual document-program interaction style is replaced with a observer-unity interaction, so to enable a more effective way of interacting and working.

K-planes show a new way of information structuring and visualization. In the case of k-planes for mail they act as filters of a mail server that stores the information in a hfs-like system, with the use of k-planes the hfs data model becomes non-relevant to the user.

In addition, a unity organization allows us to model workflows and therefore to work with k-planes so reducing the mental overhead.

In summary, k-planes exhibit a well-defined interface for stand-alone and cooperative work.

References:

[1] M. K. Buckland, What is a "document"?, *Journal of the American Society of Information Science*, Vol. 48, No. 9, 1997, pp. 804-809.

[2] D. Barreau and B. Nardi, *Finding and Reminding: File Organization from the Desktop*, in *SIGCHI Bulletin*, vol. 27, 1995, pp. 39-43.

[3] S. Whittaker, D. Frohlich, and O. Daly-Jones, Informal Workplace Communication: What is it like and how might we support it?, in Proceedings of CHI'94 Conference on Computer Human Interaction, Boston, USA, 1994.

[4] E. Freeman and S. Fertig, Lifestreams: Organizing your Electronic Life, in AAAI Fall Symposium: AI Applications in Knowledge Navigation and Retrieval, Cambridge, Mass., 1995.

[5] B. Nardi and D. Barreau, *"Finding and Reminding" Revisited: Appropiate Metaphors for File Organization at the Desktop*, in *SIGCHI Bulletin*, vol. 29, 1997.

[6] C. Sifaqui, The ICS model: simultaneous support to stand-alone and cooperative work, in 1st International Conference on Universal Access in Human-Computer Interaction, New Orleans, LA, 2001.

[7] C. Sifaqui, A process- and product-centered approach to Knowledge Management, in Applied Informatics (AI 2002), Inssbruck, Austria, 2002 to be published.

[8] H. Maturana and F. Varela, *The Tree of Knowledge: The Biological Roots of Human Understanding*, Revised Edition ed. Boston, Shambhala Publications, Inc., 1998.

[9] H. Maturana, Biology of Language: The Epistemology of Reality, in *Psychology and Biology of Language and Thought: Essays in Honor of Eric Lenneberg*, G. A. Miller and E. Lenneberg, Eds. New York: Academic Press, 1978, pp. 27-63.

[10] H. Maturana, Ontology of Observing: The Biological Foundations of Self Consciousness and the Physical Domain of Existence, in Conference Workbook: Texts in Cybernetics, American Society for Cybernetics Conference, Felton, CA, 1988.

[11] P. Dourish, W. Edwards, A. LaMarca, and M. Salisbury, Presto: an experimental model for fluid interactive document spaces, *ACM Transactions on Computer-Human Interaction*, Vol. 6, No. 2, 1999, pp. 95-132.

[12] T. Jones, *Attribute Value Systems: an Overview*, 1998.ftp://jones.tc/pub/hotos.ps.gz.

[13] Xerox, *Harland (har23)*. Palo Alto, CA, 2000.http://www.parc.xerox.com/harland.

[14] T. Munzer, Visualizing the Structure of the World Wide Web in 3D Hyperbolic Space, in Proceedings of the First Annual Symposium on the VRML Modeling Language, San Diego, CA, 1995.