Computers in Human Behavior 26 (2010) 210–217

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

Interactive web environment for collaborative and extensible diagram based learning

José Barranquero Tolosa [a], Jose E. Labra Gayo [a], Ana B. Martínez Prieto [a], Sheila Méndez Núñez [a], Patricia Ordóñez de Pablos [b,*]

[a] Department of Computer Science, University of Oviedo, Asturias, Spain
[b] Department of Business Administration, University of Oviedo, 33071 Asturias, Spain

ARTICLE INFO

Article history:
Available online 11 November 2009

Keywords:
Collaborative
Extensible
Interactive
Web environment
Diagram design
Groupware usability
User tracking
Graph visualization
Semantic web

ABSTRACT

Nowadays there is a growing need of ubiquity for learning, research and development tools, due to the portability and availability problems concerning traditional desktop applications. In this paper, we suggest an approach to avoid any further download or installation. The main goal is to offer a collaborative and extensible web environment which will cover a series of domains highly demanded by different kinds of working groups, in which it is crucial to have tools which facilitate the exchange of information and the collaboration among their members. The result of those interactions would be the development of one or several diagrams accessible from any geographical location, independently of the device employed. The environment can be adapted through personalized components, depending on the type of diagram that the user wants to interact with and the users can also create new elements or search and share components with other users of the community. By means of this environment, it will be possible to do research on the usability of collaborative tools for design diagrams, as well as research on the psychology of group interactions, assessing the results coming from the employment of known methodologies, techniques, paradigms or patterns, both at an individual and at a collaborative group level.

1. Introduction

Sometimes, the portability problem among different platforms can be simplified using virtual machines. However, the availability or application ubiquity problem is more complex. In this paper, we suggest an approach which avoids any further download or installation, thanks to the use of graphical web browsers and W3C standard technologies.

From this ubiquity perspective, the main goal is to develop an interactive web environment, where a group formed by several collaborative users, which could be geographically separated, interact synchronously to learn and practice diagram design issues in an easy way. These working groups could be multidisciplinary, embracing approaches like e-Learning (Sigala, 2007), research or development and could also form a virtual community (Fuchs, 2007).

In this paper we treat diagrams as graph structures. A simple definition for graph is: *"representations with nodes and edges to model the relationships within the space represented"*, proposed by Kargar and Schraefel (2006). And the fact is that this kind of graph structured information is present in many areas, like science taxonomies, semantic networks, UML diagrams, organizational charts, navigation diagrams and web maps, biology, chemistry, data structures, Petri nets, data flows, logic programming, circuit schematics, scene graphs, document management systems, conceptual maps, etc. (Herman, Melançon, & Marshall, 2000).

We start discussing some key issues about groupware usability measurements. After that, we will go through a brief overview of graph visualization background. Once explained all this related work, we present our proof of concept prototype and finally we expose our conclusions and future work.

2. Groupware usability issues

Every groupware application can be classified according to a set of taxonomies. One of the best known is the one proposed by Johansen et al. (1991), reviewed by Ellis (1991), and presented in Table 1. Our studies will be mainly focused on *distributed* and *synchronous* approaches, like real-time collaboration among groups; but we also plan to cover *distributed* and *asynchronous* scenarios, like teacher–student interactions or challenges.

As mentioned in the article by Baker, Greenberg, and Gutwin (2001), nowadays there are known problems related to the

* Corresponding author. Tel.: +34 985106206.
E-mail addresses: barranquero@gmail.com (J.B. Tolosa), labra@uniovi.es (Jose E. Labra Gayo), belenmp@uniovi.es (Ana B. Martínez Prieto), sheilamendeznunez@gmail.com (S.M. Núñez), patriop@uniovi.es, patriciaordonezdepablos@yahoo.com (P.O. de Pablos).

0747-5632/$ - see front matter Crown Copyright © 2009 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.chb.2009.10.003

Table 1
Groupware spatiotemporal taxonomy (Johansen et al., 1991).

	Same time	Different time
Same location	Face to face interaction	Asynchronous
Different location	Distributed and synchronous	Distributed and asynchronous

evaluation of the usability of groupware applications. They remark that it is required an appropriate methodology of groupware usability evaluation and propose an adapted version of Nielsen's (1994) heuristics, focusing on *shared visual workspaces* and supported by Gutwin's conceptual framework for the *mechanics of collaboration* (Gutwin & Greenberg, 2000).

Nielsen (1994) defines heuristics as *"general rules used to describe common properties of usable interfaces"*. Baker et al. (2001) adapt this technique to obtain eight low-cost heuristics for specific groupware shared visual workspaces:

(1) Provide the means for intentional and appropriate verbal communication.
(2) Provide the means for intentional and appropriate gestural communication.
(3) Provide consequential communication of an individual's embodiment.
(4) Provide consequential communication of shared artifacts (i.e. artifact feedthrough).
(5) Provide protection.
(6) Management of tightly and loosely-coupled collaboration.
(7) Allow people to coordinate their actions.
(8) Facilitate finding collaborators and establishing contact.

The key issue around these rules is *awareness*, defined as *"the up-to-the-moment understanding of another person's interaction with the shared workspace"* (Gutwin & Greenberg, 2004). They also affirm that awareness information increases robustness, coordination and efficiency of groupware applications, identifying three generic questions to answer in this context and detailed in Table 2.

The three first rules cover intentional and unintentional communication among collaborators, like instant messaging and individual's embodiment representation, which can be successfully achieved with *telepointers* and *avatars*. Those interaction mechanisms allow collaborative users to collect information about each other's movements and what are they doing (Baker et al., 2001; Gutwin & Greenberg, 2004).

The fourth rule is focused on shared artifacts, assuming that users need to know what is happening. This can be accomplished by showing all users the initial, in-action and final state of such artifacts while being manipulated by some user (Gutwin, 1997). Change history is additionally proposed in Baker et al. (2001).

Table 2
Workspace awareness elements and questions (Gutwin and Greenberg, 2004).

Category	Element	Specific questions
Who	Presence	Is anyone in the workspace?
	Identity	Who is participating? Who is that?
	Authorship	Who is doing that?
What	Action	What are they doing?
	Intention	What goal is that action part of?
	Artifact	What object are they working on?
Where	Location	Where are they working?
	Gaze	Where are they looking?
	View	How much can they see?
	Reach	How far can they reach?

The fifth rule deals with security issues about what can do each user within the workspace during a specific session. Margaritis, Fidas, and Avouris (2007) evaluate some of the *floor control* concerns and conclude that the groups adapt their interaction to the workspace constraints, obtaining similar quality and performance results.

The sixth rule evaluates the viewport sharing among users. In this sense, we have adopted a relaxed WYSIWIS (*What You See Is What I See*), described by Stefik, Bobrow, Foster, Lanning, and Tatar (1987). More recent works studies advanced visibility techniques like *radar view*, *over-the-shoulder view* or *cursor's eye view* of the collaborators (Gutwin & Greenberg, 2004).

The seventh rule is intended to solve coordination problems like duplication of actions, overlapping and others explained in more detail on Gutwin's thesis (1997).

The last rule focuses on searching collaborators and establishing contact with them. A good example is the *room metaphor* proposed by Greenberg and Roseman (2003), where it is easy to find out who is online and allows users to freely jump between synchronous and asynchronous interaction.

We are playing special emphasis in the usability of the collaborative and communicative mechanisms among users, trying to apply and evaluate all the eight heuristics as we will discuss later on.

In addition to groupware heuristic evaluation, automatic evaluation of usability can be done by user tracking. Ivory and Hearst (2001) review the state of the art of these mechanisms of automatic evaluation and it is an excellent start point to understand these approaches.

Fidas, Katsanos, Papachristos, Tselios, and Avouris (2007) stress the growing importance of remote usability evaluation, including user tracking. Moreover, the survey by Hilbert and Redmiles (2000) studies potential information to be extracted from user interface events. Finally, aspect oriented programming is suggested by Tarta and Moldovan (2006), focusing this problem from another point of view.

3. Graph visualization background

There are a lot of authors who have revisited this research area, with many excellent overviews of the state of art (Battista, Eades, Tamassiaa, & Tollis, 1994; Battista, Eades, Tamassiaa, & Tollis, 1999). Special mention to the survey by Herman et al. (2000), where the visualization and navigation of graphs is studied through the point of view of graph structured information visualization, like Semantic Web approaches.

The main factor in graph visualization is the size of the graph to be drawn. As Herman et al. (2000) affirm in their survey: *"It is well known that comprehension and detailed analysis of data in graph structures is easiest when the size of the displayed graph is small. In general, displaying an entire large graph may give an indication of the overall structure or a location within it but makes it difficult to comprehend"*.

Related usability issues involve different *cognitive aspects* like environment, color, visual attention, space perception, images, text and others; explained in more detail in Ware's (2000) book. We recommend the study and test of these concepts as qualitative improvements for graph tools usability.

However, there are many other constraints, which affect the visualization of graphs. For example, *planarity* is checked in order to minimize edge crossing, but it can get quite complicated when we have to deal with other *aesthetic rules*, like restrictions about the type of edges (straight lines only, same length, weight distributed, symmetry, etc.), minimization of the full area to be drawn and others (Battista et al., 1999). For further reading about

aesthetic rules, refer to the work of Purchase (Purchase, 1998; Purchase, Cohen, & James, 1995).

Herman et al. (2000) remark that *"minimization of the full graph area might be an important criterion in applications"* and Purchase et al. (1995) demonstrate that *"reducing the crossings is by far the most important aesthetic, while minimizing the number of bends and maximizing symmetry have a lesser effect"*.

Moreover, *predictability* address that two similar graph structures have to lead to two similar graph visualizations, in order to ensure that the mental map of the user is preserved (Herman, Delest, & Melançon, 1998; Misue, Eades, Lai, & Sugiyama, 1995). This can be especially relevant when working with the same graph or incremental versions of the same graph structure.

Finally, time complexity has to be reviewed to guarantee that each kind of graph is visualized with the proper layout algorithm in terms of time consumption (Herman et al., 2000).

3.1. Overview of graph layout algorithms

Battista et al. (1999) and Mutzel et al. (1997) provide a good starting point to layout algorithms for graph representation. They classify graph layout algorithms according to the following factors: rank assignment, crossing minimization, hierarchy, sub-graph extraction, planarity, compaction, augment, edge insertion and layout itself.

There is huge list of possible layout types. The classical tree layout and its variants like H-trees, radial trees, cone trees, balloon trees, tree-maps, onion graphs or spanning trees, usually provide good predictability and time complexity, which is linear in the number of nodes (Herman et al., 2000). Even they can preserve information about hierarchy in most cases.

Taking up planarity again, it can be an important issue to take care about, particularly in circuit based diagrams. Nevertheless, its complexity has been proved to be NP when dealing with *upward planarity* (rectilinear edges or edges with the same direction), and linear on basic undirected graphs (Hopcroft & Tarjan, 1974).

Sugiyama layout (Sugiyama, Tagawa, & Toda, 1989) is intended to cover general directed graphs, applying an approach called layering. This technique consists in gathering all the nodes in several layers, in accordance with their intrinsic properties, like similarity, weight or relevance among them.

Tutte (1963) was one of the first authors to propose a heuristic for edge crossing minimization, based on the premise that *"a node should be kept close to its neighbours"*.

On the other hand, *Force-Directed* or *Spring* layouts are based on physical models, with variable complexity, usually determined by its quality. The major problem with them is that they are non-deterministic and highly unpredictable, thus they are less interesting for information visualization due to subsequent interaction problems (Herman et al., 2000). However, they can be very useful when dealing with interactive workspaces, as well as client–server distributed applications, especially agent based approaches.

Another vast collection of layout algorithms are those based on grid positioning, which focuses on the distribution of the nodes by coordinates (Battista et al., 1994); however, it is claimed (Herman et al., 2000) that this kind of layouts do not play a central role in graph visualization.

3.1.1. Spanning and clustering

Sub-graph extraction techniques are well suited for information visualization because they address the minimization of the size of the data to be viewed. Spanning trees are a good example of layout algorithms using those techniques, where the problem is turned into looking for the better spanning tree of the graph. The complexity of that task can be reduced to *O(N logN)* as is asserted in Herman et al. (1998) and Eades (1992). It can also alleviate predictability issues of graph representation (Herman et al., 2000).

Spanning trees can be optimized by applying a clustering mechanism, which can also improve navigational factors as we will discuss later on. By now, starting from semantic information view, clustering can be done not only over the structure of the graph, but over the semantic relations among nodes and edges. Thus we can achieve a better approach for context and detail together, taking advance of the knowledge wrapped by the whole graph (Kargar & Schraefel, 2006).

With a well balanced algorithm in terms of the relation between the number of cluster and nodes, we can achieve a good performance in processing and navigating over graph structures (Herman et al., 2000). Other approaches are based on hierarchical clustering, like *Statecharts* (Harel, 1987), and node metrics as mentioned in Herman et al. (2000).

3.2. Graph structured semantic information interaction

Nowadays many researchers are discussing about the relevance of Semantic Web in the evolution of the Web itself. We bear out this idea in an optimistic way and propose a simple prototype of a web environment for collaborative and extensible diagram design, supported by Semantic Web techniques for information persistence and supporting visualization, navigation and edition of Semantic Web information as its key feature. Thus, we face up the two main questions in Semantic Web interaction, *"What I want to do/know?"* and *"Is graph visualization the best choice to represent semantic information?"*

The first question deals with bringing at the same time the sufficient and necessary information that the user needs to face up a concrete task (Kargar & Schraefel, 2006). The second one is quite more difficult to answer, because of the absence of profuse and rigorous research in that area. Herman et al. (2000) declare that the question would be *"Is there an inherent relation among the data elements to be visualized?"* On the other side, Kargar and Schraefel (2006) conclude that *"everything can be represented by a graph, and yet we do not use graphs to represent everything"*.

Deeper on, other questions come on scene: Where am I? What is related? Whose interest is being served? What activity is being supported? Is graph visualization the best tool for examining graph and clustering information? Is it useful? Those questions have not been solved yet and need much more research work over them.

Zoom and *Focus + Context* mechanisms have been studied since many years ago and there are a lot of articles published over different fields of human–computer interaction. In this case, zoom fits well in graph navigation, since graph representations are usually based on simple geometrical figures instead of raster images, avoiding aliasing problems (Herman et al., 2000). A broader approach is semantic zooming, where content is revealed increasingly, taking support of clustering techniques like mentioned in previous section.

It is obvious that when we zoom in a graph we lose information about the context and when we zoom out we lost details. The best known solution is to provide an overview of the entire graph, which can guarantee a minimal context, but further research is needed in this area as is required in Kargar and Schraefel (2006). Nowadays, researches are focused on fisheye distortion, interaction between focus distortions and the subjacent layout of the graph, semantic Focus + Context and even *semantic fisheye* (Herman et al., 2000).

4. A prototype for collaborative and extensible diagram design

By now, we have developed a prototype of the environment where a group of users can interact synchronously to design and share general-purpose diagrams, supporting the main web

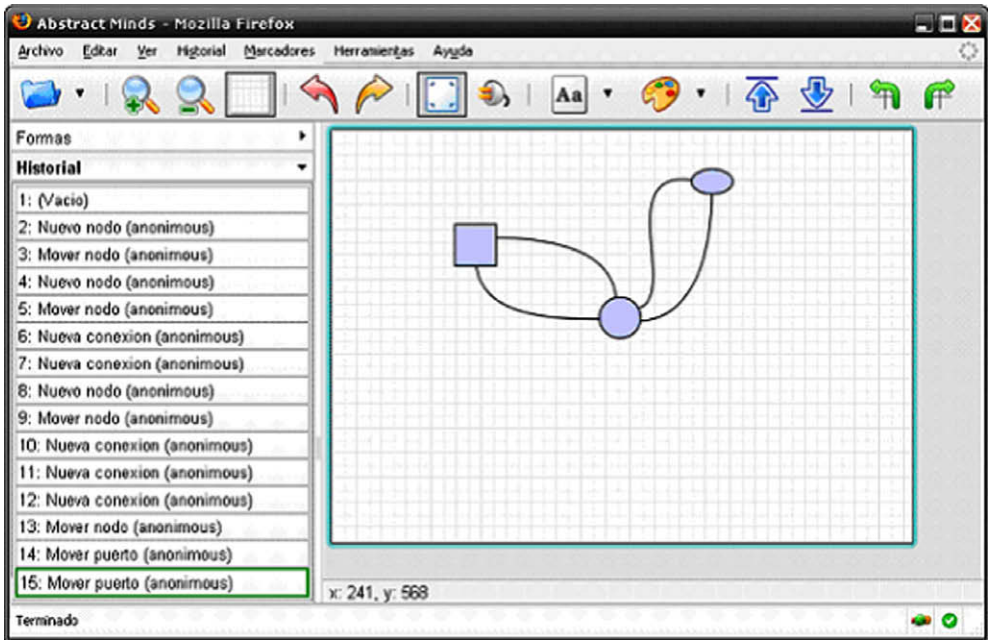

Fig. 1. Diagram change history.

browsers. Objects can be drawn in the workspace design surface and any user can make connections between them, stressing functionalities like action history with undo/redo possibilities and information about authorship, as shown in Fig. 1.

The current prototype supports full scalable vector graphics like SVG and VML and matrix-based geometrical 2D transformations, thanks to the use of Dojo Toolkit 0.9.0 (Dojo Foundation, 2007). Fig. 2 shows a basic example of this kind of graphics.

Since the beginning, we have decided to adopt a relaxed WYSIWIS viewport interaction (Baker et al., 2001; Stefik et al., 1987), because we realize that freedom of interaction is essential for a satisfactory user experience. We have also developed a limited privilege management module; however, export/import and version control is planned to be included in subsequent prototypes, as well as improvements concerning awareness issues mentioned in Section 2.

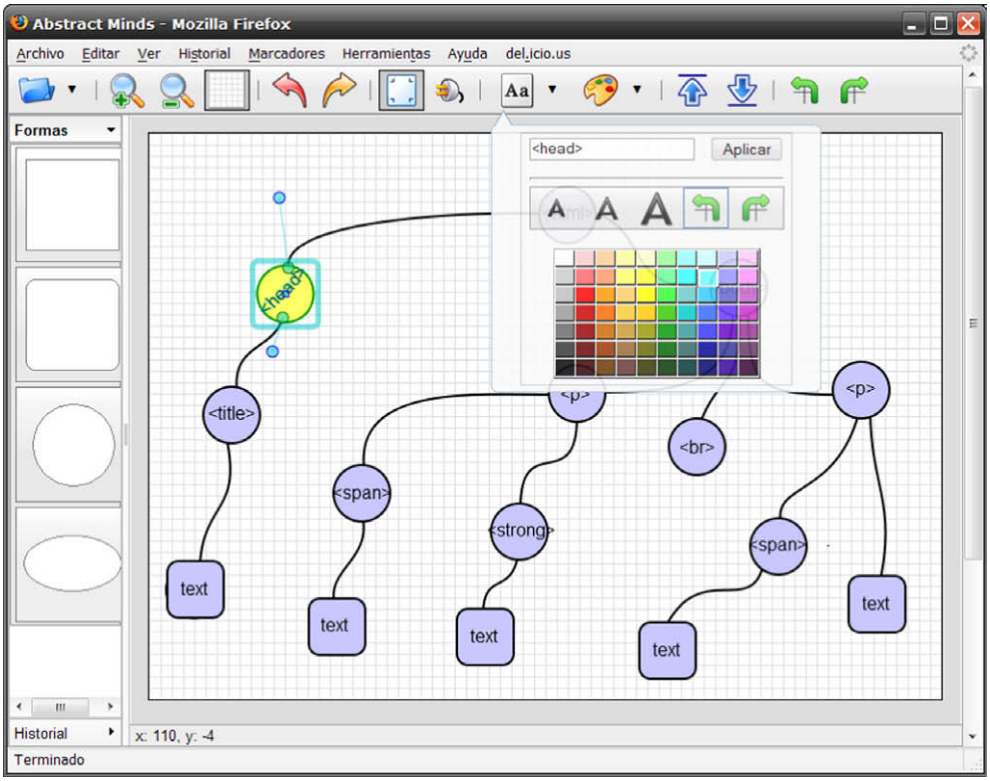

Fig. 2. Basic 2D shapes and transformations.

4.1. System architecture

The system is designed with a modular, extensible and flexible architecture, which allows enlarging and adapting the environmental functionality, thanks to the integration or actualization of new components or the proper management of the existing ones, by means of a specific tool manager, based on Semantic Web technologies.

As a web application, the system can be divided in two independent parts. On the one hand, the server follows an adapted version of the MVC *(Model-View-Controller)* architectural pattern to obtain a correct separation of concerns (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996; Reenskaug, 2003). On the other hand, the client consists of an editor tool based on components, inspired in the JHotDraw framework (Gamma & Eggenschwiler, 1998), trying to take advantage of current dynamic scripting languages of web browsers, as well as AJAX libraries, which facilitate the development of rich internet applications. We justify this choice due to the fact that MVC is a proved and customizable architectural pattern for web applications and JHotDraw is widely referenced sample framework, and even now it is continuously been improved because of its widespread and efficient use for desktop drawing applications.

4.1.1. Server side

The main goal of the design of the Server is to obtain a scalable subsystem regarding the number of users, and extensible in functionality for new research needs, incorporating new services, filters or components which will allow to apply data mining techniques by mathematical models and statistics.

As Fig. 3 shows, we have applied the *n*-layers pattern for the internal architecture of the server, through the combined use of the extended patterns Service Locator and Business Delegate, and thus augmenting its scalability possibilities (Sun Microsystems, 2002).

The view is in charge of transferring information to the client, through web pages and AJAX interactions. The controller manages the working flow, depending on the state and the events that happen, and the model captures the access to the resources and data.

The use of a controller that centralizes access and manages the flow of the application allows reconfiguring the sequence of operations done for each request without the need to directly modify the involved elements. At the same time, the business logic of the model is designed with the goal that the system will functionally evolve in subsequent extensions, avoiding the need to introduce bigger structural changes, thanks to the Business Wrappers layer.

Finally, the Abstract Factory layer, based on DAO (*Data Access Object*) design pattern, uncouples the persistence system employed through a family of objects that encapsulate the data access to the repository. The management of transactions is done in a transparent way through the use of the Transaction Manager module in the infrastructure layer, which is supported in the DAO Abstract Factory layer to abstract the specific implementation details of each type of repository.

4.1.2. Client side

The design of the client is aimed to offer the biggest freedom of configuration, personalization and functional extensibility. The fact that it is based on AJAX technologies facilitates this goal, given that it can load new components at runtime, obtaining them directly from the server, thanks to the employment of dynamic languages in the client. Among other options, the editor can be adapted using personalized toolbars which depend on the type of diagram that the user wants to design, creating new elements for the environment, and searching and sharing components with other users of the community.

We collect an operations record which offers the possibility of undoing and redoing the changes made by any collaborator in real time, including a version control system for each diagram and export facilities to different graphical formats like SVG, VML, etc. We will also offer communicative mechanisms among users to provide useful and efficient collaboration, with successful awareness information about collaborators.

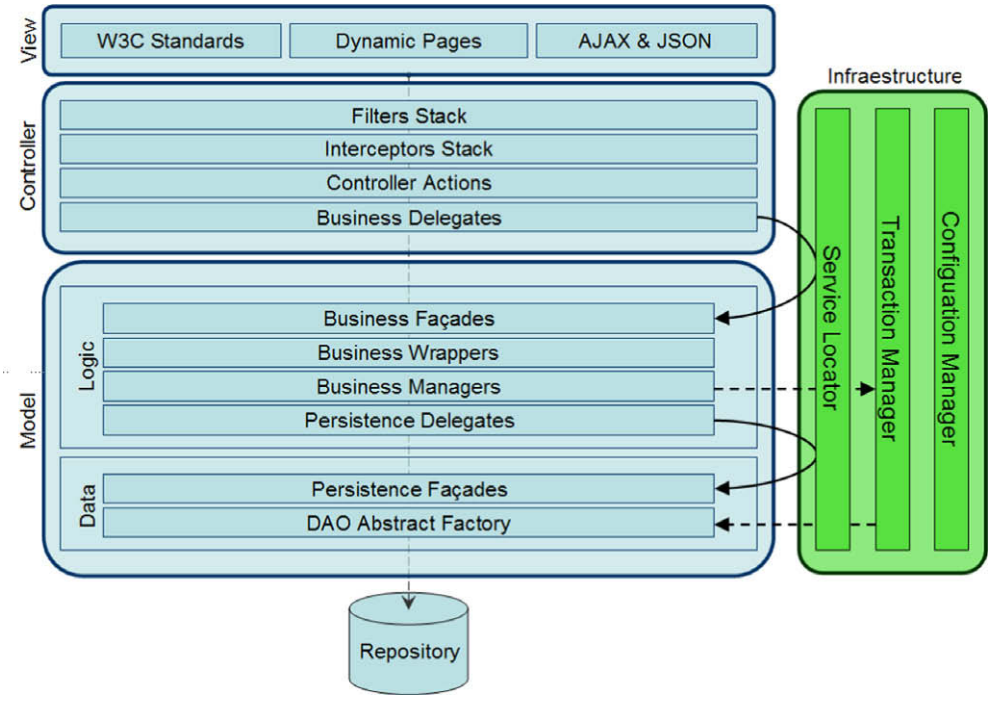

Fig. 3. Server MVC based architecture.

In order to obtain all this, an optimal design is required; hence, we are planning to readapt the best features of the JHotDraw framework (Gamma & Eggenschwiler, 1998) with the support of a dynamic language based on prototypes, like JavaScript, including improvements which can be very useful to conduct usability tests of our tool, like monitoring user steps or the interaction among collaborative working groups. We realized that the architecture proposed in this framework deals very well with our objectives and we are improving it with some issues around distributed browser-based clients.

4.1.3. Client–server communication

There are different information flows between client and server depending on the specific operation; basic data interchange is presented through a global point of view in Fig. 4.

The editor's configuration process involves all those actions that only affect the client that performs them, like modifying zoom factor, managing tool bars, changing visual look and feel, etc. These changes are stored as simple XML documents on server. The issues about this kind of relaxed WYSIWIS interaction are explained in more detail in Stefik et al. (1987).

The synchronization of changes made by a working group on the same resource requires the updating of those changes in real time on every client through the server, assuming the managing of the change history and the resolution of conflicts among versions. We use JSON notation in this scenario because it is interpreted and evaluated automatically by the client's JavaScript engine and the resulting JSON messages have a very small size when compared to other formats (Crockford, 2006).

Moreover, we have designed this client–server interaction to be completely extensible, allowing client improvements without major server changes. This feature is achieved through *duck-typing* possibilities of JavaScript, which enables the development and integration of new drawing primitives efficiently. We can include new client functionalities directly because of the change history module only validates the actions that can be applied over a node or edge, easily extended thought semantic definitions, and the data exchanged is generic, covering only target object identity, action performed and parameters. This approach also simplifies database design and maintenance, storing basic JSON undo and redo information for each action.

In contrast, for the versions control module it may be more appropriate to use an XML format, which is easily extensible and processed by the server, as well as much more portable. The use of such a format would split each version of the same diagram in a separate document, which can be easily validated by XML Schema. In this sense, GraphML, a language for graph representation (Brandes, 2000), stands out and it is used successfully in many commercial products, as yFiles (yWorks, 2004). Notwithstanding, we want to support RDF in order to facilitate the use and edition of this semantic vocabulary.

Finally, to export/import diagrams and components, we are developing filters to accept a variety of formats that may allow different levels of interoperability with other applications, as the already discussed GraphML or others like RDF, SVG vector graphics images, raster images, PDF documents, etc.

4.1.4. User interaction and layout algorithms

Our published prototype does not provide layout features because we have focused our efforts in the usability of the editor for now, but we plan to integrate them by web services, treated as intelligent agents (Etzioni & Weld, 1995; Wooldridge & Jennings, 1995). This approach enables the design of layout algorithms as distributed processes, obtaining a better user experience. Users can invoke these agents as special collaborators, which modify the diagram incrementally and even interact with other human or computer collaborators with a domain specific language to adjust algorithms parameters in real time.

We are always focusing on preserving the *mental map* of the users (Herman et al., 1998; Misue et al., 1995), so the layouts

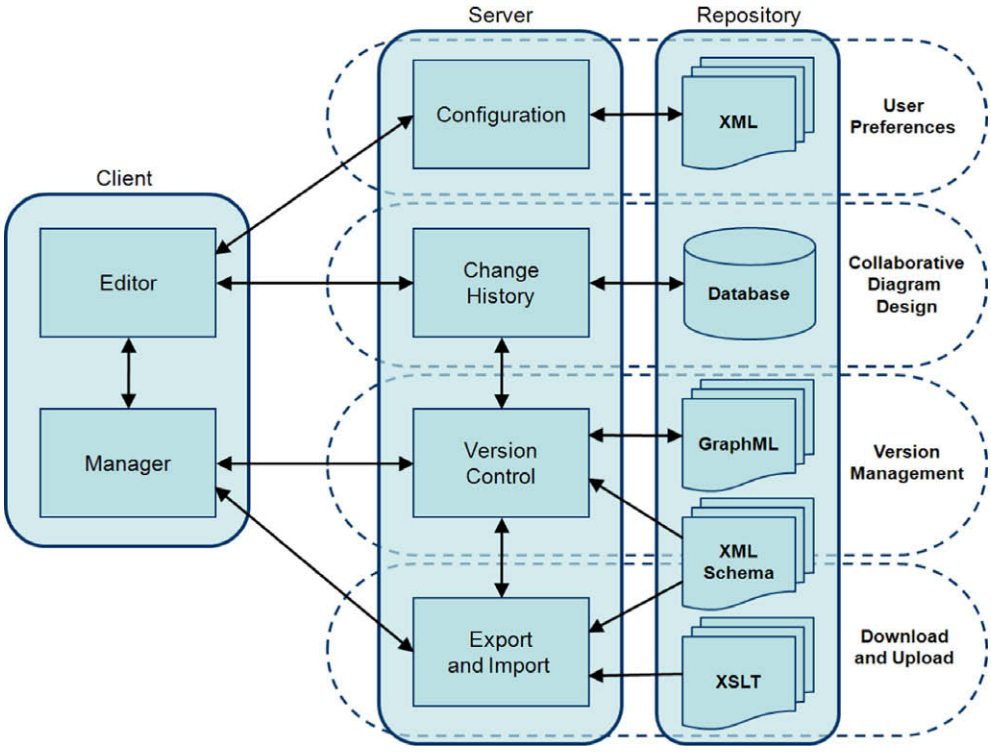

Fig. 4. Data interchange flows.

provided will be adapted to be as deterministic as possible. Moreover, our agent-based approach can be integrated easily with the change history module for undo and redo support; and the layout will be persistent between sessions, economizing resources of the environment.

5. Conclusion and future work

There are a growing number of examples of web applications based on AJAX, although not all could justify their use, given the disadvantages posed. One of them is the overload of requests to the server in case the mechanisms for the exchange and synchronization are not properly planned, which could be worsened due to memory leaks in poorly designed clients, leading to an unsatisfactory user experience. However, there are many advantages within a well-structured and extensible AJAX client. Among others, the chance of fully customize the working environment, share resources and tools, collaborate with other users in real time and access the environment from any system that includes a graphical web browser with a JavaScript engine.

Currently, there are many prototypes, libraries and functional systems like JViews, InfiView, yFiles, mind42, Cumulate Draw, mxGraph, Flowchart, Gliffy, etc.; mainly based on applets, flash and AJAX. These systems enable users to design different types of diagrams by using web technologies, although some of them offer limited functionality or are proprietary solutions. We have also developed a functional Open Source prototype of the system, based on AJAX and JSON, which allows several users from different web browsers to collaborate synchronously in the same diagram (Barranquero & Labra, 2007) and we are performing a deeper study of these examples to publish a thorough classification.

Once achieved an appropriate degree of functionality, we want to extract significant samples of the use of the environment to carry out research on the tool usability, as well as on the psychology of group interactions, assessing the results coming from the employment of known methodologies, techniques, paradigms or patterns, both at an individual and at a collaborative group level.

Furthermore, content-based or semantic clustering and layering requires domain-specific knowledge, which can be supported by collaborative environments and social networks (Ehrlich, 2006), by terms of using AJAX and Web 2.0 technologies. We have realized that the domain experts of each field of application must be encouraged to participate actively in the design of the visualization and interaction components in an easy and intuitive way. The environment would be adapted by means of personalized components, depending on the type of diagram that the user wants to interact with; creating new elements for the environment, and searching and sharing components with other users of the community. However, those components are not limited to information visualization ones, users could customize agent-based layout algorithms or even toolbars and interface views to support community needs.

References

Baker, K., Greenberg, S., & Gutwin, C. (2001). Heuristic evaluation of groupware based on the mechanics of collaboration. In *Paper presented at 8th IFIP working conference on engineering for human–computer interaction (EHCI'01)*, Toronto, Canada.

Barranquero, J., & Labra, J. E. (2007). Web based diagram editor (Alpha version) [Computer software]. Available from Open Source project at Sourceforge.net.

Battista, G., Eades, P., Tamassiaa, R., & Tollis, I. G. (1994). Algorithms for drawing graphs: An annotated bibliography. *Computational Geometry: Theory and Applications, 4*(5), 235–282.

Battista, G., Eades, P., Tamassiaa, R., & Tollis, I. G. (1999). *Graph drawing: Algorithms for the visualization of graphs*. New Jersey: Prentice Hall.

Brandes, U. et al. (2000). *The GraphML file format*. Available from http://graphml.graphdrawing.org Accessed 07.05.08.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture: A system of patterns*. New York: John Wiley and Sons.

Crockford, D. (2006). JSON: The fat-free alternative to XML. *Paper presented at Proceedings of XML 2006*, Boston. Available from http://www.json.org/fatfree.html Accessed 05.05.08.

Dojo Foundation (2007). *The dojo toolkit* (Version 0.9.0) [Software Toolkit]. Available from http://download.dojotoolkit.org/release-0.9.0/ Accessed 21.02.07.

Eades, P. (1992). Drawing free trees. *Bulletin of the Institute for Combinatorics and its Applications*, 10–36.

Ehrlich, D. M. (2006). Social network survey paper. *International Journal of Learning and Intellectual Capital, 3*(2), 166–177.

Ellis, C. (1991). Groupware: Some issues and experiences. *Communications of the ACM, 34*, 39–58.

Etzioni, O., & Weld, D. S. (1995). Intelligent agents on the internet: Fact, fiction, and forecast. *IEEE Expert: Intelligent Systems and Their Applications, 10*(4), 44–49.

Fidas, C., Katsanos, C., Papachristos, E., Tselios, N., & Avouris, N. (2007). Remote usability evaluation methods and tools: A survey. *Paper presented at Pan-Hellenic Conference on Informatics*.

Fuchs, C. (2007). Towards a dynamic theory of virtual communities. *International Journal of Knowledge and Learning, 3*(4/5), 372–403.

Gamma, E. & Eggenschwiler, T. (1998). JHotDraw as open-source project. Available from http://www.jhotdraw.org Accessed 12.05.08.

Greenberg, S., & Roseman, M. (2003). Using a room metaphor to ease transitions in groupware. In M. Ackerman, V. Pipek, & V. Wulf (Eds.), *Sharing expertise: Beyond knowledge management* (pp. 203–256). Cambridge: MIT Press.

Gutwin, C. (1997). Workspace awareness in real-time distributed groupware. Ph.D. Thesis, Dept. of Computer Science, University of Calgary, Canada.

Gutwin, C. & Greenberg, S. (2000). The mechanics of collaboration: Developing low cost usability evaluation methods for shared workspaces. In *Paper presented at IEEE 9th international workshop on enabling technologies: Infrastructure for collaborative enterprises (WET-ICE'00)*, Gaithersburg, Maryland.

Gutwin, C., & Greenberg, S. (2004). The importance of awareness for team cognition in distributed collaboration. In E. Salas & S. M. Fiore (Eds.), *Team cognition: Understanding the factors that drive process and performance* (pp. 177–201). Washington: APA Press.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming, 8*(3), 231–274.

Herman, I., Delest, M., & Melançon, G. (1998). Tree visualization and navigation clues for information visualization. *Computer Graphics Forum, 17*(2), 153–165.

Herman, I., Melançon, G., & Marshall, M. S. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics, 6*(1), 24–43.

Hilbert, D. M., & Redmiles, D. F. (2000). Extracting usability information from user interface events. *ACM Computer Surveys, 32*(4), 384–421.

Hopcroft, J., & Tarjan, R. E. (1974). Efficient planarity testing. *Journal of the ACM, 21*(4), 549–568.

Ivory, M. Y., & Hearst, M. A. (2001). The state of art in automating usability evaluation of user interfaces. *ACM Computer Surveys, 33*(4), 470–516.

Johansen, R., Sibbet, D., Benson, S., Martin, A., Mittman, R., & Saffo, P. (1991). *Leading business teams: How teams can use technology and group process tools to enhance performance*. Boston: Addison-Wesley.

Kargar, D. & Schraefel, M. C. (2006). The pathetic fallacy of RDF. *Paper presented at semantic web user interaction workshop (SWUI 2006) at the interantional semantic web conference ISWC 2006*, Athens, Georgia.

Margaritis, M., Fidas, C. & Avouris, N. (2007). A framework to facilitate building of collaborative learning applications [Special issue]. *Advanced Technology for Collaborative Learning (ATL) International Journal, 4*(1).

Misue, K., Eades, P., Lai, W., & Sugiyama, K. (1995). Layout adjustment and the mental map. *Journal of Visual Languages and Computing, 6*, 183–210.

Mutzel, P., Gutwengwer, C., Brockenauer, R., Fialko, S., Klau, G., Kruger, M., Ziegler, T., Naher, S., Alberts, D., Ambras, D., Koch, G., Junger, M., Bucheim, C., & Leipert, S. (1997). A library of algorithms for graph drawing. In *Symposium on Graph Drawing GD '97*, Springer-Verlag, pp. 456–457.

Nielsen, J. (1994). Heuristic evaluation. In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods*. New York: John Wiley & Sons.

Purchase, H. C. (1998). Which aesthetic has the greatest effect on human understanding. In *Symposium on Graph Drawing GD '97*, Springer-Verlag, pp. 248–261.

Purchase, H. C., Cohen, R. F., & James, M. (1995). Validating graph drawing aesthetics. In *Symposium on Graph Drawing GD '95*, Springer-Verlag, pp. 435–446.

Reenskaug, T. (2003). *The model-view-controller (MVC): It's past and present*. Paper presented at the metting of JavaZONE, Oslo.

Sigala, M. (2007). Integrating web 2.0 in e-learning environments: A socio-technical approach. *International Journal of Knowledge and Learning, 3*(6), 628–648.

Stefik, M., Bobrow, D., Foster, G., Lanning, S., & Tatar, D. (1987). WYSIWIS Revised: Early experiences with Multiuser interfaces. *ACM Transactions on Office Information Systems, 5*(2), 147–167.

Sugiyama, K., Tagawa, S., & Toda, M. (1989). Methods for visual understanding of hierarchical systems structures. *IEEE Transactions on Systems, Man and Cybernetics, SMC, 11*(2), 109–125.

Sun Microsystems (2002). *Core J2EE patterns*. Available from Sun Developer Network Java Blueprints: http://java.sun.com/blueprints/corej2eepatterns Accessed 10.05.08.

Tarta, A. M., & Moldovan, G. S. (2006). Automatic usability using AOP. *IEEE International Conference on Automation, Quality and Testing, Robotics, 2*, 84–89.
Tutte, W. (1963). How to draw a graph. *Proceedings of the London Mathematical Society, 3*(13), 743–768.
Ware, C. (2000). *Information visualization: Perception for design*. San Francisco: Morgan Kaufmann.
Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review, 10*(2), 115–152.
yWorks, The Diagramming Company (2004). *yFiles: Diagramming that works*. Available from http://www.yworks.com/products/yfiles/doc/yFiles_E.pdf Accessed 08.05.08.