Journal of Universal Computer Science, vol. 19, no. 7 (2013), 873-893
submitted: 31/8/12, accepted: 31/1/13, appeared: 1/4/13 © J.UCS

On the Development and Usability of a Diagram-based Collaborative Brainstorming Component

Diogo Azevedo, Benjamim Fonseca, Hugo Paredes
(INESC TEC – INESC Technology & Science, Porto, Portugal
UTAD - University of Trás-os-Montes and Alto Douro, Vila Real, Portugal
{diogoa, benjaf, hparedes}@utad.pt)

Stephan Lukosch, Jordan Janeiro
(Systems Engineering Section, Delft University of Technology, Delft, The Netherlands
{s.g.lukosch, j.janeiro}@tudelft.nl)

Robert Owen Briggs
(MIS Department, San Diego University, California, USA
rbriggs@mail.sdsu.edu)

Abstract: The need for computer-supported collaboration has grown over the last years and made collaboration an important factor within organizations. This trend has resulted in the development of a variety of tools and technologies to support the various forms of collaboration. Many collaborative processes, e.g. strategy building, scenario analysis, root cause analysis and requirements engineering, require various collaboration support tools. Data flow, fishbone and brainstorming diagrams, play an important role within these synchronous collaborative applications to create, evaluate, elaborate, discuss, and revise graphical models. Currently, the necessary tools are not integrated and flexible enough to support such processes. In this paper, a synchronous collaborative brainstorming diagram editor integrated in a flexible group support system is described. This approach goes beyond the current state of the art as it can be seamlessly integrated with other collaboration support tools such as text-based brainstorming or voting. The usability of the taken approach is evaluated within a case study on collaborative learning.

Keywords: Brainstorm, Diagram Editor, CSCW, Collaboration, ActionCenters, Diagram-based
Categories: D.2.1, D.2.2, D.2.13, H.5.3, J.0

1 Introduction

Computer Supported Cooperative Work (CSCW) aims at improving the performance of a group in the execution of tasks by providing suitable information and communication technologies. Groups can become even more productive when supported by Group Support Systems (GSS) [Davison, 01] [Briggs, 10]. It is decisive that GSS adopt techniques for the development of groupware applications – A group of people working within the same system or application, no matter where they happen to be – that meet non-functional requirements (quality attributes) such as interoperability, integration, reliability and usability [Pelegrina, 10]. Currently, there is lack of support on GSS for such processes. GSS must therefore offer users collaborative environments where they can interact [Duque, 09], however many of

these systems fail when providing the right tools for effective collaboration [Grudin, 94]. Analysing how groups work and evolve is necessary when the social dimension of collaborative work is considered [Grudin, 88]. Therefore, it is useful to support collaboration engineers (CE) – teach practitioners just the techniques they need to conduct their own mission-critical work practice successfully – in designing a collaborative work practice, and then use that design to develop a collaborative software application tailored specifically for that task, with the right tools, communication channels, data, and most importantly, collaboration guidance for each activity [Buttler, 11]. Collaboration engineering researchers are seeking ways to package collaboration expertise with collaborative technology in a form that practitioners could reuse without training on either the tools or the collaboration techniques [Briggs, 10], [Buttler, 11], [Mametjanov, 11]. So following this approach, by combining the CLSD with the ActionCenters, and with the results of this paper, practitioners can experience the potential benefits of collaboration technology without having to take special training.

ActionCenters is a web-based platform to develop and use effective collaborative practices. In detail, ActionCenters provides:

- A rapid development studio that collaboration engineers can use to create task-specific collaborative application called ActionCenters;
- An online library of ActionCenters for use by collaboration facilitators;
- A run-time platform where an ActionCenter presents practitioners with tools, communication channels, information, and collaboration guidance for each step of their task.

Collaborative graphical systems support a group of people concurrently editing graphical processes over the network. In the case of object-based graphical editing, the central shared information space is a unique scene of objects shared among users. Previous approaches have applied and classified them into locking, serialization and multi-versioning [Ignat, 06]. In the locking approach adopted by systems such as Aspects [Biel, 91], Ensemble [Newman-Wolfe, 92] and GroupDraw [Greenberg, 92] concurrency is restricted, and concurrent editing is allowed only if users are locking and editing different objects, and moreover responsiveness – the capability to answer on time – is affected due to delays for lock acquisition [Ignat, 06].

No matter how many capabilities a system provides, it is likely that there will be work practices that require domain-specific capabilities the system does not yet support, e.g. specialized editors for different modeling language such as UML or BPMN [Buttler, 11]. Therefore, the Collaborative Line-and-Symbol Diagramming - CLSD Component presented herein offers a collaborative environment to manage graphical models and thereby their related collaborative processes. To create such a collaborative environment the techniques and diagram types that could be used to support collaborative diagramming efforts were taken into consideration, and how the features and functions of a single-user differ from a multi-user diagramming tool in order to optimize the values that groups can create through collaborative diagramming. The CLSD Component is developed as component for ActionCenters and is integrated as a plug-in component within the Computer Assisted Collaboration Engineering platform (CACE) of ActionCenters. Thereby, the component can be used in various different processes. ActionCenters presents a step forward in supporting collaborative processes, since it allows collaboration engineers to design a

collaborative work practice, and then use that design to develop a collaborative software application tailored specifically for that task [Briggs, 10][Buttler, 11]. Furthermore, CACE provides collaboration engineers with a rapid application development environment in which they design collaborative components into applications to support those work practices [Buttler, 11]. It embeds collaboration expertise with collaboration technologies [Briggs, 10], so that participants can gain the same benefits without any special training [Mametjanov, 11].

In the remaining of this paper, the requirement analysis that a GSS system has to fulfil is presented (section 2.1) followed by the requirements gathered from other existing diagram-editors (section 2.2) and the requirements that the CLSD Component demands (section 2.3). In section 2.4 the technical requirements that the chosen GSS system (ActionCenters) requires to be taken in consideration are also described. In the next section the architecture (section 3.1) and features (section 3.2) of the CLSD are presented, as well as the scope description (section 3.3). Before concluding and pointing to future directions, a case study (section 4) to test the system usability of the CLSD Component and the related results analysis (section 5) are presented.

2 Requirement Analysis

Appropriating GSSs to better represent a process is a complex task, mainly because such systems have to be flexible enough to be personalized according the process [Buttler, 11]. Therefore, this section addresses the requirements that a GSS system has to fulfil in order to support the CLSD Component. Following, the requirements gathered from other existing diagram editors are presented and the features that fit our collaborative diagram editor are selected. The functional requirements that the CLSD Component has to fulfil in order to allow collaboration engineers to configure synchronous collaborative applications that actually fit specific collaborative processes, such as strategy building, scenario analysis, root cause analysis and requirements engineering are described. Finally, the scope and the technical requirements of the CLSD Component are addressed.

2.1 GSS Requirements

To illustrate the requirements that a GSS system has to fulfill in order to support the CLSD Component and the interoperability needed between components a scenario of a collaborative strategy building process that uses collaborative diagramming and other collaborative applications, e.g. a text-based brainstorming – performed within a group of people to generate ideas to solve a problem [Peter, 11] is presented. Three activities that can be considered in this scenario are [Buttler, 11]:

1. Text-based brainstorming for strategy building (e.g. Outliner Component), where groups will use a shared-outlining component to review, comment on, and revise a taxonomy of system requirements to assure that all key concerns will be addressed in the requirements negotiation;
2. Diagram-based brainstorming (e.g. CLSD Component) to generate a collection of possible requirements;
3. Diagram-based brainstorming (e.g. CLSD Component) to organize, connect and manage strategies based on the data gathered in the previous activity, and to reduce

them to a set they deem worthy of further attention, and to organize them under the categories of their requirement taxonomy.

In this scenario, the GSS system has to support collaboration engineers in designing collaborative processes (R1), such as strategy building, root cause analysis, and design suitable collaboration support (R2). For that the GSS needs to support the integration of components that support collaborative processes (R3), by allowing re-using of existing components [Pelegrina, 10]. Furthermore, it must be able to share and exchange data efficiently (interoperability) between components (R4) [Pelegrina, 10], [Hofte, 95], [Simone, 99], in order to reuse the data gathered for example from the first activity (text-based brainstorming) into the second activity (diagram-based brainstorming). Additionally, since all the support that is needed is not known the set of components must be extensible (R5) [Pelegrina, 10]) by software developers and an API (R6) to support them [Riehle, 00] should be provided. Finally, our scenario requires collaborative diagramming (R7), and for that additional requirements have been identified.

The ActionCenters supports the design of collaborative applications (R1), and allows components (as our CLSD Component) to be assembled by Collaboration Engineers into the CACE editor (R2). These components have access to shared data (R4), are configurable (R3) and can be (re)-designed by other Collaboration Engineers (R5). They usually consist of a user interface for displaying data shared in a group, some input mechanism, and business logic. ActionCenters is published under the BSD license, and is therefore open to anyone who wants to add new components [Buttler, 11]. Thus, ActionCenters fulfils the requirement R5. Furthermore, the ActionCenter provides two JavaScript objects to manage data and their updates – ActionCenterListener, and an ActionCentersAPI (R6) that offers services to create and support the development of collaborative components. Additionally, the data is managed through dynamic communication channels using CometD[1] to a Universal Data Model (UDM) [Mametjanov, 11], to dynamically create and store arbitrary relational data. The UDM and the two JavaScript objects offer some mechanisms to manage contribution, such as *modifiedBy* to know who changed the data, and *lockedBy* to edit-lock entities and their attributes to provide single-user editing. The ActionCenters does not have all the necessary tools. As alternative, these tools are plugged into the ActionCenters as components to simply make them available in the runtime system [Azevedo, 11].

2.2 Requirements gathered from other existing Diagram Editors

According to [Pelegrina, 10] there are GSS systems addressing some of the requirements described above, however for our approach ActionCenters were chosen because it addresses all of the requirements and it fits with our purpose. However, ActionCenters does not address all requirements needed for Collaborative Diagramming. For that purpose, our (R7) CLSD Component that consists of an XML wrapper and an implementation in JavaScript with Ext JS[2] and an extended library

[1] The Dojo foundation. Cometd. More information can be found in http://cometd.org/, 2011.

[2] Ext JS is a javascript framework for developers. More information can be found in http://www.sencha.com/, 2011.

called Joint JS[3] was implemented. The JointJS library is used to create diagrams that can be fully interactive for both implementing a diagramming tool (as our CLSD Component) as well as simply for publishing diagrams (R7.1, R7.2 and R7.3) [Azevedo, 11]. Some mechanisms to know who changed the data (R7.4), and to edit-lock entities and their attributes to provide single-user editing (R7.5) make part of the requirements of the CLSD Component. The last requirements (R7.6 and R7.7) have been implemented in the XML wrapper, which is the CLSD Component, and the API provided by the ActionCenters was used as support to implement it.

The list of requirements is based on the analysis of other existing Diagram Software, such as Banxia[4] [Venable, 05], Smart Ideas[5] [Kyriakou, 10] and Ext Designer[6]. In this case, the requirements address the interaction that Collaborative Diagramming has to provide to groups while they participate in collaborative environments. It must be possible for group members to insert, import (text-based) and manage ideas into a diagram-based format (R7.1), like our previous strategy-building scenario. Then, ideas are diagram-based organized (clusters and colour manager) (R7.2) and connected through arrows (R7.3). Furthermore, group members can unintentionally provoke data conflicts between contributions and therefore it is required to provide remote field of vision - awareness with the scope (who has been doing what) of other members' activities [Dix, 93], and data with their information and the resources that are nearby [Segal, 95] (R7.4), and furthermore triggered locking mechanisms when updates occur (R7.5). Moreover, when changing from text to model based the CLSD Component should allow consensus building such as the organization of concepts even when their position is not defined (R7.6) (this can happen when data is imported from other components, such as the Outliner Component). Finally, the CLSD Component needs to implement a set of rules so that it would be possible to identify conflicting relations, such as arrows that will be connected at least at 2 concepts and also concepts that can change automatically from colour when moving from categories.

The features and requirements that the CLSD Component should support to create a collaborative modelling tool (R7), a scenario identifying the main and necessary functionalities of diagram editors was done. In this scenario the diagram editor should generate (blocks) concepts based on text (R7.1**)** (previously inserted on the database or at runtime), and furthermore it should converge on key concepts allowing users to merge sub-categories from main categories (R7.2). To converge on the key concepts the diagram editor needs first to allow the connection (link) between concepts to (R7.3), which also leads to the organization of the diagram (R7.2) (model relations). It should also have the necessary features and tools in order to provide awareness to other users in the collaborative environment, such as telepointers, remote field of vision and so on (R7.4). Furthermore, concurrency control is very

[3] Joint JS is a JavaScript library developed by David Durman, More information can be found in http://www.jointjs.com/, 2011.

[4] Banxia (Decision Explorer) is a proven tool for managing software issues. Structure and analyse of qualitative information. More information can be found in http://www.banxia.com/dexplore/, 2011.

[5] Smart Ideas concept-mapping software brings the power of visual learning to classrooms, through interactive white boards. More information can be found in http://smarttech.com/, 2011.

[6] Ext Gui Designer is a graphical user interface builder for web applications. Developed by Sierk Hoeksma. More information can be found in http://www.projectspace.nl/, 2011.

common in collaborative scenarios since there is more than one user trying to manipulate the same data, which leads to the requirement of locking contributions upon their manipulation (R7.5). Moreover, in this scenario the diagram editor should provide consensus building on relations when moving from text to model (R7.6), as well as identify conflicting relations (R7.7) (an arrow will be connected at least at two concepts).

The requirements gathered from COMA, SmartIdeas and Banxia [Azevedo, 11] were compared to the above scenario to validate the behavior of the scenario itself and to see if these requirements fit with it. This comparison was done to see if it is possible to implement it in collaborative diagram editors, such as the CLSD Component. After making such analysis and concluding that the above scenario match with the approach used in the diagram editors studied, the main features that the CLSD Component should provide and the requirements mentioned in the above scenario that must match with the requirements implemented in the CLSD Component where described in the following section.

2.3 Collaborative Line-and-Symbol Diagramming Requirements

The detailed analysis of diagram editors and their features revealed the most important requirements: add, edit and delete concepts (brainstorming concepts, ideas, words, blocks and so on); add and delete arrows to connect concepts; add, edit and delete notes; and auto save concepts, arrows, notes and diagrams every time any change is made. Furthermore other important requirements are: cluster concepts by displaying different colours for each category; export diagrams through XML files to be displayed out of the ActionCenters; add and delete telepointers (limited to 1 per user). To create a collaborative environment awareness that [Dourish, 92] defined as an understanding of others activities, which provides a context for your own activity should be concerned. According to [Gutwin, 04b] group awareness information includes knowledge about who is on the collaborative environment, where they are working, what are they doing and their subsequently intentions [Pelegrina, 10]. Therefore, locking mechanisms, remote field of vision and telepointers extend the requirements of working in collaborative environments.

Furthermore the studied requirements were compared with the requirements that our CLSD Component support. CLSD Component generate concepts based on text fetched from other existing components or from the database, or at runtime (R7.1); it is possible to connect concepts with arrows (R7.3) and differentiate them from categories and subcategories by highlighting their path (R7.2); it also allow users to know who changed the data, and to edit-lock entities and their attributes to provide single-user editing (R7.4) (R7.5), such features have been supported by the ActionCentersAPI; the organization of the diagram (R7.6) is possible in the CLSD Component by dragging concepts and dropping them in the right position, however it is only possible to do it manually, so it is not possible to randomize automatically their position; finally it has a set of rules to identify conflicting relations (R7.7) (e.g. each arrow will have at least two concepts) and to maintain consensus building when moving from text (e.g. Outliner component) to model (CLSD Component) based.

2.4 Technical Requirements

There were some issues to implement the CLSD Component since the GSS system – ActionCenters has some technical requirements that need to be followed to have a successful implementation and workability of the CLSD. These requirements start with concurrency control because of the locking mechanisms used to prevent data conflicts between users and it proceeds with the structure and features of the components, where it is mandatory for all components to be a XML wrapper implemented in JavaScript.

Locking mechanisms have different levels of optimism, such as Non-optimistic, Semi-optimistic, and Fully-optimistic (Table 1) [Greenberg, 94]. Therefore in our approach the level of optimism that has been used and implemented has been a Non-optimistic level because users cannot manipulate contributions while waiting for its lock whereas the lock must be triggered and then the user who triggered it can manipulate the contribution [Azevedo, 11]. It is mandatory to wait for the lock in order to manipulate the contribution and users cannot release the changed contributions while waiting for its lock because the lock must be previously triggered. This level of optimism was used because it fits better with the implementation requirements of the ActionCenters, which does not allow developers to manage the access to contributions without using locking mechanisms, so that it has been defined that the manipulation of contributions can only be done after they have been locked and by who locked it. The changes made at contributions are first released to all users and then the contribution is unlocked [Azevedo, 11].

Level of optimism	Can manipulate the object while waiting for its lock	Can release the changed object while waiting for its lock
Non-optimistic	No	No
Semi-optimistic	Yes	No
Fully-optimistic	Yes	Yes

Table 1: Optimism level of locking mechanisms

XML was created so that richly structured documents could be used over the web; furthermore a XML wrapper is a mechanism to identify structures (markup language) where the XML specification defines a standard way of adding markup to documents [Walsh, 97]. These structured documents contain content and some indication of what role that content plays. Moreover, since the name of the attribute, its value and the corresponding data type has to be defined a XML Schema was used to describe them [W3C, 11][Buttler, 11]. This decision is based on the large adoption of XML and XML Schema as standards to describe XML-based generic structures [Buttler, 11].

The CLSD Component handles users' contributions based on the XML Schema, so each contribution may have different attributes identified by an *id*, its semantic type (*key*), data type and its value (*value*). Besides, the attribute also has a reference back to the contribution that contains it [Buttler, 11]. Furthermore, a tag *<js><code></code></js>* is defined where all JavaScript code is written including the JavaScript

libraries that are going to be used. In the CLSD Component the JointJS library and our implementation development in this tag were included. The other tag categories define the layout of the component and are used by the ActionCenters to implement and define the attributes that each component will have. These attributes can be manipulated directly in the CACE editor (after uploading the XML wrapper into ActionCenters), instead of making changes in the XML wrapper, users just need to access the properties of a component to display the attributes defined in the XML wrapper.

The ActionCenters and the CLSD Component summary requirements are addressed at Table 2:

ActionCenters	CLSD
R1: Design collaborative processes	**R7:** Collaborative diagramming
R2: Design support on collaborative applications	**R7.1:** Insert and import text-based ideas, e.g. generate concepts based on text
R3: Plug-in and re-use of components (CACE Platform)	**R7.2:** Converge on key concepts: diagram-based organized (cluster and colour management)
R4: Share and exchange data efficiently (interoperability between of components)	**R7.3:** Connected arrows (connection between concepts)
R5: Extensible components	**R7.4:** Context awareness
R6: API to support the component development	**R7.5:** Locking mechanisms
-	**R7.6:** Consensus building (from text to model based)
-	**R7.7:** Identify conflicting relations

Table 2: Summary requirements of the ActionCenters and of the CLSD Component

3 CLSD Approach

This section presents a collaborative diagramming tool – CLSD developed allowing users to participate simultaneously in a brainstorming session in different computers through the network. Therefore, in the following section the architecture of the CLSD Component is explained in detail. In the next section the features of the CLSD Component are presented, and finally, a scope overview of the CLSD Component is addressed.

3.1 CLSD Architecture

The CLSD Component is a web-based application that supports the cooperation of group participants towards group work [Azevedo, 11]. For example, it might support the group in a text-based or a diagram-based brainstorming, or even support the transition between text-based to diagram-based brainstorming, and vice-versa. Figure 1 shows the overall architecture of our approach.

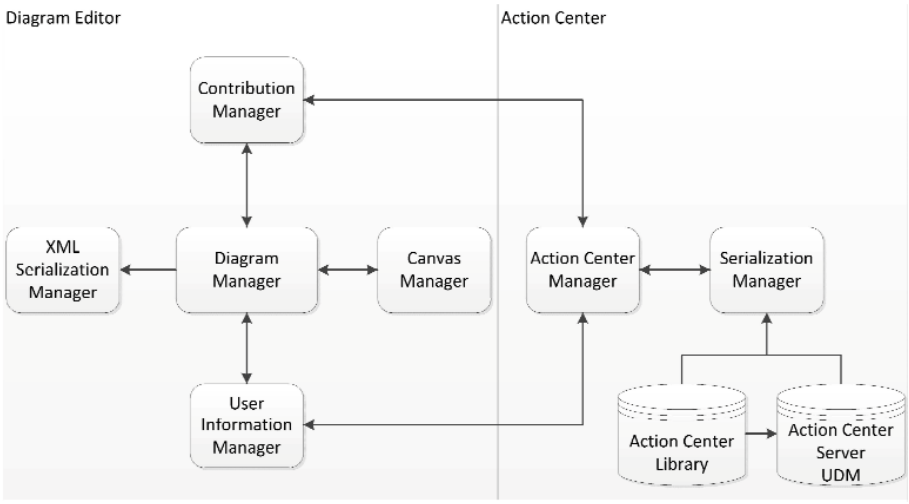

Figure 1: The CLSD Component Architecture coupled to ActionCenters [Azevedo, 11]

The CLSD Component architecture is divided in two environments:

1. Diagram editor: XML file with JavaScript code;
2. ActionCenters: web-based platform with the following technical attributes: MySQL database server, Jetty web container, Back-end Technologies – Java programming language, spring framework, and hibernate for database-Java object mapping, and Front-end Technologies – JavaScript programming language, CometD for messaging from client to server and ExtJS 3 (now sencha.com) for the UI library.

The Diagram Manager is the core manager of our CLSD Component and it is responsible for all processes of input and output, their distribution through the overall system, and for all connections inside the Diagram and between the ActionCenters and the Diagram [Azevedo, 11]. Additionally, it connects with the Canvas Manager that is the bridge between the core manager of our system and the user. The User Interface (UI) influences its degree of acceptance since it allows communication, collaboration and coordination activities among several users interacting with the system. The Canvas Manager manages the CLSD Component design, the concepts and their connectors, and the collaborative tools / awareness mechanisms required, such as the list of users in the session, telepointers – support actions, intentions and location awareness, and remote field of vision – actions of a particular user can be

shown to other users that collaborate in some task [Dix, 98][Gutwin, 04b][Penichet, 08].

The Contribution Manager can also be called of Diagram Database Manager since it is responsible for adding, fetching and updating contributions to the ActionCenters Database. These contributions can include concepts, arrows, JSON[7] messages or objects and are triggered through notification mechanisms [Azevedo, 11]. To manage the information of users that are working in the diagram, such as listening online users, giving personalized information of each of them, and the scope of their activities - group awareness becomes a critical component in successful coordination [Gutwin, 04a] - the entity User Information Manager was implemented. Finally, another feature developed was the XML Serialization Manager, which is an output file that allows users to visualize their diagrams out of the ActionCenters.

3.2 CLSD Features

The CE is responsible for uploading the component to the ActionCenters in order to make it (CLSD component) available at ActionCenters and then use it to create new diagrams. After that he chooses specific users to participate in the session and defines rules, both features through the ActionCenters. When the component is ready for the session it loads information about the users that will be in the session so that specific data can be collaboratively displayed. On one hand, features like Edit Concept, Delete Concept, Delete Arrow and Colour Manager trigger the Locking Mechanism to avoid conflicts, since many users often manipulate the same data objects at the same time [Mametjanov, 11]. Additionally, the remote field of vision is also triggered to help users to identify the scope of other users. Awareness mechanisms were used, allowing the creation of a collaborative environment where users can interact with each other. So multiuser editors make use of awareness widgets that show the working area of other users to avoid conflicting changes in a shared artefact [Schümmer and Lukosch, 07]. Consequently, it is at this point that the need to couple collaborative components with awareness was meet and understand, without them even an expert will have difficulties to implement a diagram in a group, as well as successfully interact with it, or even in helping the rest of the group participants. When a user inserts a new Concept (idea), telepointer or an Arrow he is creating it for the first time, so there is no concurrent access to data and consequentially there is no need to have locking mechanisms at this point, neither the user scope. To call user's attention the CLSD Component allows users to use telepointers, 1 per user, each time they want to inform another user about a particular or independent situation. Finally, at any point of the session users can manually save the diagram, otherwise the contributions that were previously added and automatically saved to the database already contain the full representation of the diagram session.

3.3 CLSD Component

ActionCenters in combination with CLSD Component (Figure 2) allows us to support various different processes that require different forms of collaboration, such as a strategy building process where data is gathered from a text-based brainstorming and

[7] JSON (JavaScript Object Notation) is a lightweight data-interchange format.

then fetched and organized in a diagram-based brainstorming (CLSD Component). The union of the text-based with the CLSD Component creates an ActionCenter, where the data, which is selected (identified) based on their relationship types and attributes by the ActionCenters, is forward fetched (import) from the UDM – Universal Data Model and loaded (insert) to the CLSD Component. CLSD Component transforms it into a diagram-based format where group members can manage and organize data as collaborative processes. Each single user controls the selection and manipulation of data and until he or she has finished nobody else can have access to manipulate that specific data. For this purpose, at each moment when concepts are being changed it shows a locking icon and a scope of action (remote field of vision) of the user who is manipulating it. One possible approach for our CLSD Component was to send notifications to inform the users about the conflicts, but instead locking mechanisms when updating contributions to the database was decided to use, which stops possible conflicts between users and unnecessary notifications. Users don't have advantages for knowing that a conflict has occurred, such unnecessary notifications could disturb their work and focus.

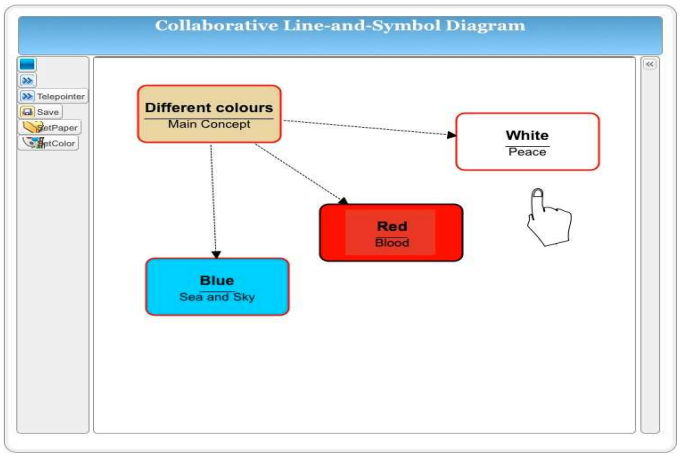

Figure 2: Collaborative Line-and-Symbol Diagramming Component

CLSD interface (Figure 2) has a *Menu* in the left side of the screen with six buttons, which functions are:

- 1st button (■) to create *Concepts* (they have a title and a description);
- 2nd button (>>) to create *Arrows*;
- 3rd buttons (>>Telepointer) creates Telepointers;
- *Save* button to save the current diagram (*Canvas*);
- *SetPaper* button to change the size of the canvas area (*Canvas*);
- *SetColour* button to change the default colour upon the creation of *Concepts*.

To use the first three buttons (creation of concepts, arrows and telepointers), the user needs to drag and drop it into the *Canvas* area, which is the place where the model-based concepts and the brainstorming are created and organized. All concepts, arrows

and telepointers dragged to areas outside the limits of the canvas cannot be seen, so they cannot be used to represent the diagram.

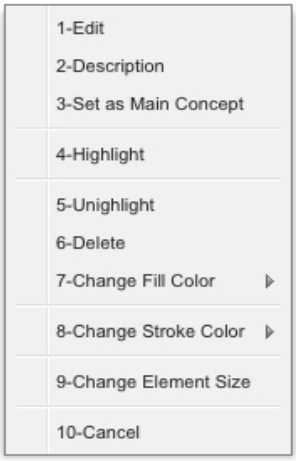

Figure 3: Context Menu properties of concepts

Users can have access to the features of concepts by double-clicking or right clicking the desired concept. This action triggers the context menu (Figure 3) for the current user, awareness widgets to the other users and locking mechanisms to the current concept. The context menu provides a tool to users so that they can change the title of the concept (1-Edit), their description (2-Description) and set them has main concepts (3-Set as Main Concept). The latter property will change the description of the concept to "Main Concept" so that it can be identified as the principal concept or main idea, and to prevent having multiple main concepts. If the main concept was already defined, a warning message will inform the user about this and asks to change it. Furthermore, the concepts that are linked to other concepts by arrows can be highlighted (4-Highlight and 5-Unhighlight) to focus their path and consecutively improve visibility. The only common tool of concepts, arrows and telepointers after their creation is the option to delete them (6-Delete), which erases the contribution and attributes created for concepts, arrows, telepointers and canvas. It is possible to change the fill (7-Change Fill Colour) and stroke (8-Change Stroke Colour) colour of concepts, or also by selecting a colour for default from the boxes or by tipping the colour code into the text area. Finally, in case that the user did not like the size of the concept compared to the text inside of it he can always change it manually (9-Change Element Size).

Considering a two activities scenario, the first activity is the use of the Outliner for the gathering of text-based ideas and the second activity the conversion to model-based ideas with CLSD Component. In case of the first activity – Outliner it has three main features: the text-based area where the ideas are presented, a text area to input ideas and a navigation bar to change between activities. Tools to manipulate text, such as edit, insert, and remove of ideas, are part of the set of features presented in the outliner. All these attributes are different and made for different proposes when

compared to the CLSD Component, so to use these two components (CLSD and outliner) and make them work with interoperability and in parallel it is needed to have communication and sharing of data between them, which will allow users to be working on a text-based (Outliner Component), and other users at the same time manipulating and organizing the same ideas in a model-based approach (CLSD Component). For both components it is mandatory to have the same population rule and data set, and to manipulate the attributes of the contributions used; for the CLSD Component the attributes were defined and for the outliner the developer provided the attributes: *id, fromdate, thumbprints, type* and *name*. These attributes were not changed in the outliner instead the developer manipulated the CLSD attributes making possible to have interoperability between components. Therefore it is possible to create a project at ActionCenters with these two components, invite users to make an exercise and see their progress independently of the component that is being used, since they are both collaborative and the ideas introduced are automatically synchronized between the CLSD and the Outliner. Furthermore, it is possible to see the activity that was more productive and that had more positive feedback from users, allowing to improve even more the interaction and integration of users with the collaborative widget.

4 Case Study

A case study to validate the usability of the CLSD Component and to assess the student performance gains when using the CLSD Component was conducted, with 41 students of three distinct subjects and courses from the University of Trás-os-Montes e Alto Douro: BSc in Communication and Multimedia, BSc in Humans Rehabilitation and Accessibility Engineering, and PhD in Informatics. The aim of this study was to analyse the CLSD usability. Therefore a quantitative questionnaire named Computer System Usability based on [Lewis, 95], extended with a set of qualitative questions, was given to the students after they performed their tasks on the CLSD. The CLSD was used in 3 different collaborative learning scenarios, to help the student groups to perform their current tasks, and to find solutions to current daily problems at their university.

1. The first scenario in which CLSD was used was within the Social and Cooperative Platforms course from the BSc in Communication and Multimedia. The course focuses on the use of social networks, multiuser 3D spaces and Web 2.0 developing platforms for communication and cooperation strategies. Here 7 groups of 4 students and 1 group of 3 students have used CLSD to record the general ideas (Figure 4) and then used these ideas as a guide in the developing process of their work.
2. The second scenario using the CLSD was within the Telematics Applications for Inclusion course from the Humans Rehabilitation and Accessibility Engineering Degree to help solving accessibility gaps at UTAD (Figure 5). In this case, a group of 8 students have made a brainstorming session, in which they first have identified and discussed the existing accessibility problems (Figure 5). After gathering all the current problems, they have made other discussions at the same brainstorming session but now with the purpose of gathering solutions and linking them with problems.
3. In the third scenario, PhD students of Informatics conducted a brainstorming session to identify the problems of a specific PhD research. The CLSD was used after

a brief presentation of the research topic and the group of students have presented their perspectives and scientific approaches for the current problem. In this case, 4 PhD students held the brainstorming but only 2 questionnaires were collected.

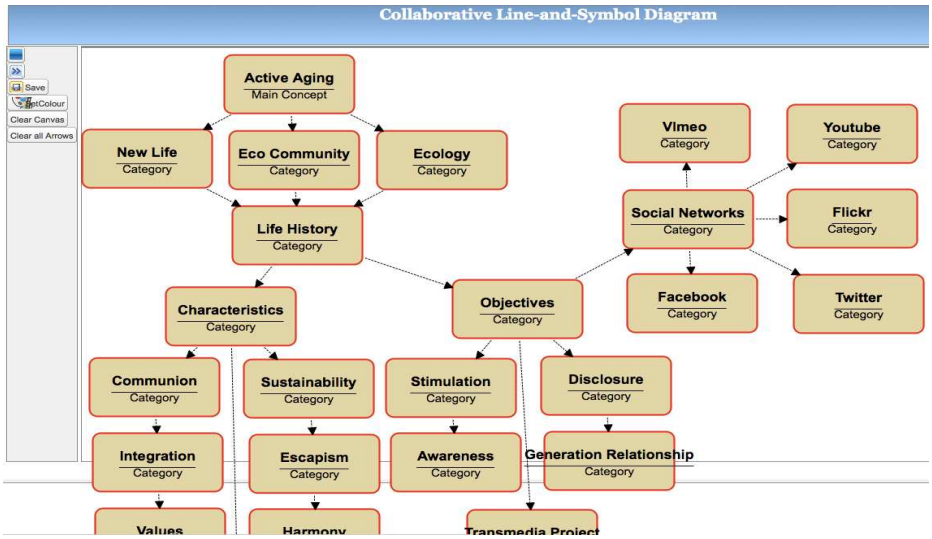

Figure 4: Brainstorming made by a group of students upon the realization of a task

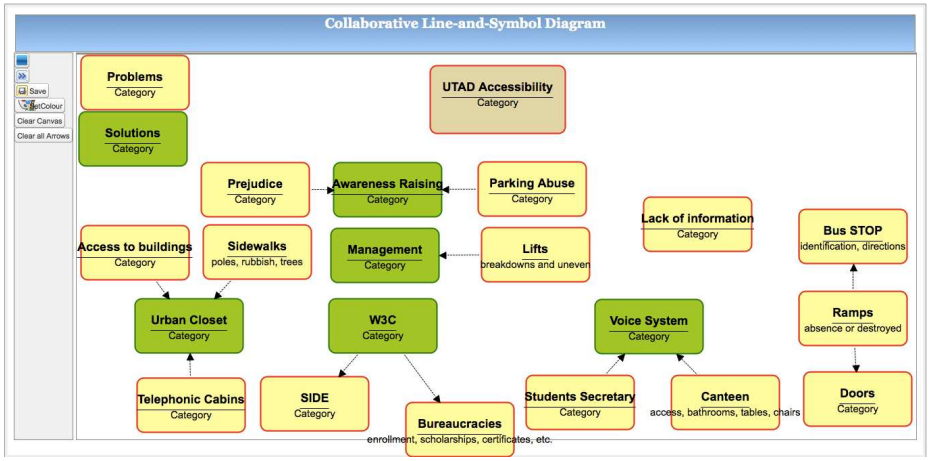

Figure 5: Brainstorming made by a group of students that focus on the problem solving of the accessibility gaps at UTAD

In terms of achievements of the current tasks 80,48% of the students have agreed that they could effectively complete their work using CLSD, where 78,0% were able to complete their work quickly, and 65,9% could efficiently complete the current work. Though 73,2% of the students believe that they became productive quickly

using the CLSD. Such achievements and productivity may be related to the simplicity of CLSD because 90,2% of the students claims that it was easy to learn how to use CLSD. This information can be found in more detail in Table 3.

Computer System Usability	
Overall, I am satisfied with how easy it is to use CLSD	75,6%
It was simple to use CLSD	63,4%
I can effectively complete my work using CLSD	80,5%
I am able to complete my work quickly using CLSD	78,0%
I am able to efficiently complete my work using CLSD	65,9%
I feel comfortable using CLSD	78,0%
It was easy to learn to use CLSD	90,2%
I believe I became productive quickly using CLSD	73,2%
The CLSD gives error messages that clearly tell me how to fix problems	29,3%
Whenever I make a mistake using the CLSD, I recover easily and quickly	41,5%
The information provided with CLSD is clear	65,9%
It is easy to find the information I needed	82,9%
The information provided for CLSD is easy to understand	80,5%
The information is effective in helping me complete the tasks and scenarios	75,6%
The organization of information on CLSD screens is clear	82,9%
The interface of CLSD is pleasant	82,9%
I like using the interface of CLSD	61,0%
CLSD has all the functions and capabilities I expect it to have	43,9%
Overall, I am satisfied with CLSD	78,0%

Table 3: Computer system usability questionnaire [Lewis, 95] results

The students from the first subject were able to efficiently complete the proposed work and moved to the next activity. The second subject has gathered the accessibility problems at UTAD and then related it with proven solutions to their current problems. The PhD students have done one brainstorming session were all their feedback, related to a mobile applications PhD topic, was given to help improving the research objectives and issues of the current research.

5 Results

The quantitative and qualitative questionnaire delivered to students during the case study has generated several results about the usability of the CLSD and the related features, such as: ease of use, interface and organization of CLSD, the effectiveness and efficiency to complete the proposed work, the productivity gained, usability issues and so on. The results of the quantitative questionnaire based on [Lewis, 95] are described and presented by percentage of the most relevant results. These results can be found at Table 3. Furthermore, a more specific analysis to the quantitative questionnaire was done by crossing qualitative data, such as the degree, experience using collaborative tools, age of the users, and so on.

In terms of qualitative analysis the most used collaborative tools were Facebook with 51% and Twitter with 44% and the daily intention of the students upon the use of collaborative tools is related with entertainment, work and research especially for the younger students, which ages converge between 20 and 22. This last result goes beyond the extreme growth of social networks in the business, teaching and learning fields. However, the majority of the students have answered that the purpose of using collaboration tools was for work and research, but when asked which collaboration technologies they most used they have answered Facebook and Twitter, so the lack of information and knowledge of the existing collaboration technologies for work is notorious. They are not yet fully integrated with the collaborative technics and technologies that are extremely helpful in the daily life in terms of work, mobility, problem solving, and decision-making. The fact that they were using the power of social network in their subject with the goal of expansion, divulgation, propagation and so on, it might have influenced the majority of the students to answer that the most used collaboration tools were Facebook and Twitter. There are collaboration technologies that could be integrated into social networks in order to call students attention and extend the use of such tools. This could be a good approach to be put in practice and at the same time have a bigger impact in the collaboration field. However, for now the CLSD will just be supported by ActionCenters. At first view and with the first set of analysis from the questionnaires, it had revealed that the CLSD is on the right path to fulfil the demands of the users. However some issues still need to be taken care to allow a better interaction and solving problems for users. Furthermore, the features that the CLSD embrace will be extended with the feedback provided by the students.

90,22% of the group of students claimed that it was easy to learn how to use the CLSD Component, which may be related to the fact that they effectively completed their work using CLSD (80,5%). However, the experience using collaborative technologies is an important role to understand how they are used to work with such collaborative tools and ascertain their performance when using CLSD. So, both data was crossed (easy to use and experience with collaborative technologies) at Figure 6, and it was concluded that CLSD is in right path, since the most experience students are happy and had agreed that the CLSD is in fact easy to use. Finally it was concluded that the CLSD is effective in complete the tasks and scenarios (75,6%). Such results analysis makes the CLSD a powerful tool in the achievements of the students current work goals, which allows them to be more productive upon the use of the CLSD (73,1%).

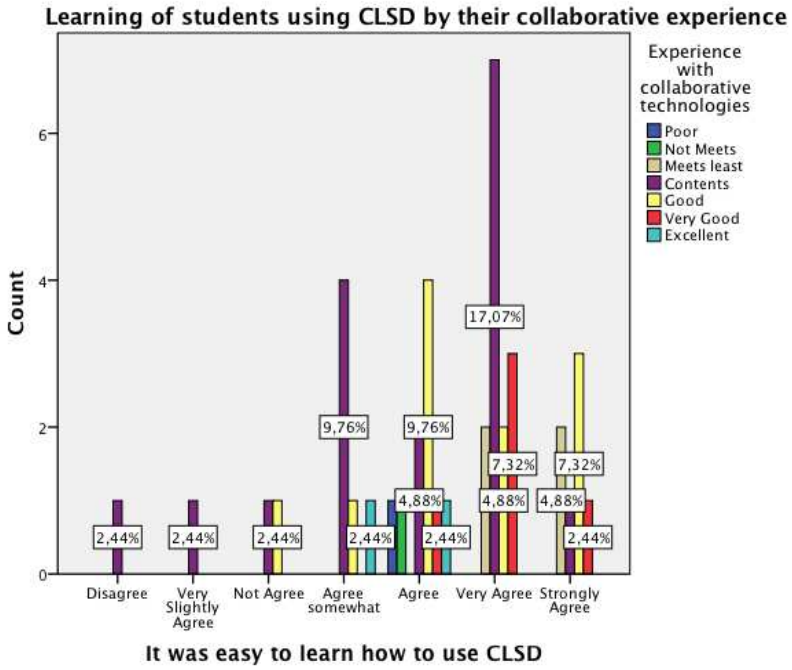

Figure 6: Learning of students using CLSD compared to their collaborative experience

6 Conclusions

In this paper, is presented a Collaborative Line-and-Symbol Diagramming Component – CLSD Component assembled in a CACE editor and supported by ActionCenters.

The requirements that the GSS system ActionCenters address are:

- (R1) Design collaborative processes;
- (R2) Design support on collaborative applications;
- (R3) Plug-in and re-use of components (CACE Platform);
- (R4) Share and exchange data efficiently (interoperability between components);
- (R5) Extensible components;
- (R6) API to support the component development.

CLSD is a collaboration support tool that consists of a XML wrapper and an implementation for creating diagrams that can be fully interactive for both implementing a diagram-based brainstorming session to manage collaborative processes as well as simply for publishing diagrams. The purposed requirements that CLSD Component address are:

- (R7) CLSD Component allows the creation and elaboration of a brainstorming session (collaborative modeling);

- (R7.1) Group members can generate concepts based on: insert, import and fetch data from other components, into a diagram-based format through a collaborative environment provided by the CLSD Component;
- (R7.2) Converge on key concepts: merge sub-categories from main categories, by highlighting their path; Organization of the resulting diagram: the concepts and linking arrows can be dragged and dropped around the canvas area, which provides the necessary feedback to help users organizing/structuring the resulting diagram. Other feature implemented that also allows the structuring of the resulting diagram is the colour manager;
- (R7.3) Connection between concepts (arrows): users can connect concepts through arrows allowing the creation of a hierarchical diagram;
- (R7.4) Context awareness: the CLSD Component provides the necessary awareness, in order to make users aware of the scope area that other users are working on, therefore he supports remote field of vision, telepointers and so on;
- (R7.5) Locking mechanisms: using the provided API from ActionCenters the CLSD Component locks concepts upon the manipulation of them preventing therefore data conflicts;
- (R7.6) Consensus building: transition from text to model based has different levels because they can be: text inserted at runtime, text inserted in other component that must be fetched by the CLSD Component, and text that is already in the database. The CLSD Component has been implemented taken into consideration these transitions levels, which provides a transparent environment to users (they do not need to know from what source the text comes from, they just need know who create it, and for that the awareness tools have been implemented);
- (R7.7) Identifying conflicting relations: set of rules that forbid the wrong use of the provided tools, such as arrows that must have at least two concepts connected, and so on;

The conduced case study revealed not only the most important issues of usability of the CLSD, but it also revealed how students can become productive in their work using this system. So, the conducted case study was not only productive for testing the usability of the CLSD, but instead it was also used to test the system in different learning scenarios. The result questionnaires obtain have allowed the crossing of all these data and is clear that the CLSD has provided the necessary environment and features to make users more productive and effective in different learning scenarios. Some of the issues revealed by this analysis and that must be passed by in a new version of the system is the information that must be clear, and a way to contradict problems, by giving more information messages or even accurate tutorials to make users aware of the existing features and the better way to use them. Concluding, the students have learned fast how to use the CLSD, so they can become productive very quickly, which allows them to conclude their work and have height percentage of success upon the realization of their works and tasks.

7 Future Work

In future, the effect of exchanging data between components will be investigated, e.g. when changing from a text-based brainstorming to a diagram-based brainstorming. In

this line, the plan is to compare traditional text-based approaches with diagram-based approaches. To easily support different diagram types, such as Fishbone Diagrams, the CLSD component will be extended. This will allow users to choose the diagram type that fits best their current needs. Moreover, the awareness support within CLSD and add features will be extended, e.g. list of online users, chat, and so on that influenced the usability of CLSD within the recent case study. Based on the resulting CLSD component, new case studies will be conducted to assess the new or updated features, and actually ascertain if users have become even more productive using CLSD. Here, the plan is a more extensive analyses including the analyses of videos showing the users interaction with each other and CLSD.

Acknowledgements

This work is funded (or part-funded) by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project <<FCOMP – 01-0124-FEDER-022701>>.

References

[Azevedo, 11] Azevedo, Diogo, Janeiro, Jordan, Lukosch, Stephan, Briggs R. O. and Fonseca, Benjamim (2011): An integrative approach to diagram-based collaborative brainstorming, *Proceedings of the ECSCW 2011 Workshop on Collaborative usage and development of models and visualizations*, scientific publication.

[Biel, 91] Biel, V. (1991): V. Groupware Grows Up. *MacUser*, June 1991, 207-211.

[Briggs, 10] Briggs, R. O., Kolfschoten, Gwendolyn L., Vreede, Gert-jan, Albrecht, C., and Lukosch, S. (2010). Facilitator in a Box: Computer Assisted Collaboration Engineering and Process Support Systems for Rapid Development of Collaborative Applications for High-Value Tasks, *Information Systems*, pp. 1-10.

[Buttler, 11] Buttler, T., Jordan Janeiro, Stephan Lukosch, and Briggs R. O. (2011): Beyond GSS: Fitting Collaboration Technology to a Given Work Practice, Collaboration and Technology – 17th International Conference, *CRIWG'11*, Paraty, Brazil, October 2011.

[Davison, 01] Davison R. M. (2001) A Survey of Group Support Systems: Technology and Operation, City University of Hong Kong, China. Sprouts: Working Papers on Information Systems, 1(12). http://sprouts.aisnet.org/1-12

[Dix, 93] Dix, A., Finlay J., Abowd G., and Beale R. (1993): Human-Computer Interaction, *Prentice Hall*.

[Dix, 98] Dix, A., Finlay J., Abowd G., and Beale R. (1998): 'Human-Computer Interaction', *Prentice Hall*.

[Duque, 09] Duque R., Manuel Noguera, Crescencio Bravo, José Luis Garrido and Maria Luisa Rodríguez (2009): Construction of interaction observation systems for collaboration analysis in groupware applications, Advances in Engineering Software 40(12): pp. 1242-1250.

[Dourish, 92] Dourish, P. and Bellotti, V. (1992): Awareness and coordination in shared workspaces, *Conference proceedings on Computer-supported cooperative work*, volume 0, pp. 107-114.

[Greenberg, 92] Greenberg, S., Roseman, M., Webster, D. and Bohnet, R. (1992): Issues and experiences designing and implementing two group drawing tools. *Proceedings of the 25th Annual Hawaii Intl. Conference on the System Science*, Kuwaii, Hawaii, January 1992, 138-150.

[Greenberg, 92] Greenberg, S. & Marwood D. (1994): Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface, Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work, 1994, 207-217.

[Grudin, 94] Grudin J. (1994): Computer-supported cooperative work: history and focus, *IEEE Comput*, pp. 19-26.

[Grudin, 88] Grudin J., (1988): Why applications fail: problems in design and evaluation of organization or organizational interfaces, *Proceedings of the ACM conference on computer-supported cooperative work*, pp. 85-93.

[Gutwin, 04a] Gutwin, C., Reagan Penner, Kevin A. Schneider (2004a): 'Group Awareness in Distributed Software Development', *CSCW*, pp. 72-81.

[Gutwin, 04b] Gutwin, C., Schneider, K., Paquette, D., Penner, R. (2004b): Supporting Group Awareness in Distributed Software Development, *Engineering Human Computer Interaction and Interactive System*, vol. 3425, pp. 383-397.

[Hofte, 95] Hofte, H., ter, Maurice A. W. Houtsma, Hermen J. van der Lugt, (1995): CSCW Infrastructure Research at TRC, *ACM SIGOIS Bulletin*, vol. 15.

[Holzinger, 05] Holzinger, A. (2005): Usability engineering methods for software developers, (C. Stephanidis Ed.) *Communication of the ACM* vol. 48(1), pp. 71-74 ACM. Retrieved from http://portal.acm.org/citation.cfm?id=1039539.1039541

[Ignat, 06] Ignat, C., and Moira C. Norrie (2006): Draw-Together: Graphical Editor for Collaborative Drawing, CSCW 06 Proceedings of the 2006 20th anniversary conference on computer supported cooperative work, ACM, pp. 269-278, ISBN: 1595932496.

[Kyriakou, 10] Kyriakou, P. Hatzilygeroudis, I, Garofalakis, J. (2010): A Tool for Managing Domain Knowledge and Helping Tutors in Intelligent Tutoring Systems, Journal of Universal Computer Science, vol. 16, no. 19 (2010), 2841-2861 submitted: 1/3/10, accepted: 29/9/10, appeared: 1/10/10 J.UCS.

[Lewis, 95] Lewis, J., R. (1995): Computer Usability Satisfaction Questionnaire IBM Psychometric Evaluation and Instructions for Use, *International Journal of Human-Computer Interaction*, 7:1, pp. 57-58.

[Mametjanov, 11] Mametjanov, A., Kjeldgaard, D., Pettepier, T., Albrecht, C., Lukosch, S., and Briggs, R. O. (2011): ARCADE: Action-centered Rapid Collaborative Application Development and Execution, *Hawaii International Conference on Systems Sciences,* volume 0, pp. 1-10.

[Newman-Wolfe, 92] Newman-Wolfe, R. E., Webb M., and Montes, M. (1992): Implicit locking in the Ensemble concurrent object-oriented graphics editor. *Proceedings of CSCW*, Toronto, Canada, pp. 265-272.

[Pelegrina, 10] Pelegrina, Ana B., Rodríguez-Dominguez, C., Rodríguez, María Luisa, Benghazi, Kawtar, and Garrido, José Luis (2010): Integrating Groupware Applications into Shared Workspaces, *RCIS*, pp. 557-568.

[Penichet, 08] Penichet, V. M. R., Maria Dolores Lozano, José A. Gallud, Ricardo T., Maria L. Rodríguez, José L. Garrido, Manuel Noguera, Maria V. Hurtado (2008): Extending and Supporting Featured User Interface Models for the Development of Groupware Applications, *J. UCS 14(19)*, pp. 3053-3070.

[Peter, 11] Peter Dolog, Frederico Araujo Durão, Karsten Jahn, Yujian Lin, Dennis Kjaersgaard Peitersen (2011): Recommending Open Linked Data in Creativity Sessions using Web Portals with Collaborative Real Time Environment. J. UCS – Journal of Universal Computer Science 17(12): 1690-1709 (2011) submitted: 15/10/10, accepted: 28/1/11, appeared: 1/8/11 © J. UCS.

[Riehle, 00] Riehle D., (2000): Framework Design: a Role Modeling Approach Dissertation, *ETH Zurich*.

[Schümmer and Lukosch, 07] Schümmer, T. and Lukosch, S. (2007): *Patterns for Computer-Mediated Interaction*, John Wiley & Sons, Ltd.

[Segal, 95] Segal L., (1995): Designing Team Workstations: The Choreography of Teamwork, *Local Applications of the Ecological Approach to Human-Machine Systems*, pp. 392-415.

[Simone, 99] Simone, C., Mark, G. and Giubbilei, D., (1999): Interoperability as a means of articulation work, *WACC '99: Proceedings of the international joint conference on Work activities coordination and collaboration*, pp. 39-48.

[Venable, 05] Venable, J. (2005): Using Coloured Cognitive Mapping to Support IS Development, in Fisher, Julie and Kautz, Karl-Heinz and Linger, Henry (ed), Workshop on IS Research: A North-South Dialogue, Nov 09 2005. Melbourne, Australia: Monash University.

[Walsh, 97] Walsh, N. (1997): A technical Introduction to XML, Journal, http://nwalsh.dom/docs/articles/xml/.

[W3C, 11] W3C. Xml Schema (2011), http://backpack.blackboard.com/