A Collaborative Web-based Issue Based Information System (IBIS) Framework

Toktam Ebadi
Maryam A. Purvis
Martin K. Purvis

The Information Science Discussion Paper Series

Number 2009/06
August 2009
ISSN 1177-455X

University of Otago

Department of Information Science

The Department of Information Science is one of seven departments that make up the School of Business at the University of Otago. The department offers courses of study leading to a major in Information Science within the BCom, BA and BSc degrees. In addition to undergraduate teaching, the department is also strongly involved in postgraduate research programmes leading to MCom, MA, MSc and PhD degrees. Research projects in spatial information processing, connectionist-based information systems, software engineering and software development, information engineering and database, software metrics, distributed information systems, multimedia information systems and information systems security are particularly well supported.

The views expressed in this paper are not necessarily those of the department as a whole. The accuracy of the information presented in this paper is the sole responsibility of the authors.

Correspondence

This paper represents work to date and may not necessarily form the basis for the authors' final conclusions relating to this topic. It is likely, however, that the paper will appear in some form in a journal or in conference proceedings in the near future. The authors would be pleased to receive correspondence in connection with any of the issues raised in this paper, or for subsequent publication details. Please write directly to the authors at the address provided below. (Details of final journal/conference publication venues for these papers are also provided on the Department's publications web pages: http://www.otago.ac.nz/informationscience/pubs/). Any other correspondence concerning the Series should be sent to the DPS Coordinator.

Department of Information Science
University of Otago
P O Box 56
Dunedin
NEW ZEALAND

Fax: +64 3 479 8311
email: dps@infoscience.otago.ac.nz
www: http://www.otago.ac.nz/informationscience/

A collaborative web-based Issue Based Information System (IBIS) framework

Toktam Ebadi, Maryam Purvis and Martin Purvis

University of Otago, Dunedin, New Zealand
Toktam_ebadi@yahoo.com, {tehrany, mpurvis}@infoscience.otago.ac.nz

Abstract

This research focuses on the design and development of an IBIS-based tool called IBISMod, which facilitates a distributed and collaborative decision-making process. IBIS-based systems help analysts and designers in the process of formulating the requirements and design issues associated with complex problems that are difficult to specify. In particular, it captures the rationale behind group decision-making process. The group members are usually distributed over a network and may be working together concurrently. IBISMod is based on Rittel's Issue-Based Information System. This particular implementation is a web-based tool that makes it possible for the participants to work together on a specific problem while they may be physically present in different locations. In order to improve the interactivity, speed and usability of the framework, the AJAX approach has been adopted.

1. Introduction

Explicit illustration of a decision rationale enables people to benefit from potential advantages of discourses, particularly in the context of group decision-making. The knowledge that people share for solving a particular problem becomes available for others to examine. The demonstration of a decision-making procedure can serve as a documentary record of decision developments, that can be used as a basis for justification and better decisions (Lee; 1990; Yakemovic and Conklin; 1990). Therefore it is desirable to find a mechanism that can support the process of group decision-making and facilitate recording of the ideas behind these decisions (Veerman and Treasure-Jones; 1999). Such a mechanism should be enabled to keep track of decisions and allow participants to inspect the discussions that belong to decisions. The users can amend the decisions if needed. Amendments may be required as the result of new development associated with concurrent activities.

Over the past years many systems have been developed for decision support. gIBIS (Conklin and Begeman; 1988) is a hypertext group tool which aims to capture the rationale of design process. gIBIS system allows users to construct and browse decision networks. Furthermore, it has been designed to support the collaborative construction of these networks by any number of cooperating team members distributed across the local network.

QuestMap developed by Conklin (1996) was an attempt to capture the key issues and reasoning during a meeting. It is based on gIBIS hypertext groupware tool (Conklin and Begeman; 1988) and aims to produce a map based on an on-line conversation that would lead to key decisions and plans.

SYBIL(Lee; 1990) is another system that supports group decision making by representing and managing qualitative aspects of decision making procedures. It is a knowledge-based system that provides services for managing of dependencies, doubts, perspectives and decisions.

HERMES (Karacapilidis and Papadias; 1999) is a web-based system that augments formal decision making processes by supporting argumentative discussion among its users. It is implemented in Java (applets) and runs on the web browser. HERMES system has the capability to be used for distributed, asynchronous collaborative projects. The system also integrates a reasoning mechanism and organizes the existing data in a discussion map.

REMAP (Ramesh and Dhar; 1992) is a conceptual model used to capture deliberation during requirements analysis, which relates process knowledge to the objects that are created during the requirements engineering process. It is based on IBIS model of argumentation (Kunz and Rittel; 1970; Rittel and Webber; 1973) and is supported by a system based on Telos language (Mylopoulos *et al.*; 1990).

Compendium (Conklin; 2001) is a hypermedia tool. It is a visual environment for collaborative, semiformal modelling. It extends Ritter's Issue-Based Information System to support collective sense making. Compendium's hypertext functionality is a significant addition to the issue-based notation derived from IBIS. Its Java-based open architecture has enabled interoperability with other database and collaboration technologies. One part of Compendium is a facilitation approach to Synchronous (generally face-to-face) meetings that focuses the group on the collaborative structure of multi-dimensional models. Another part of Compendium is off-line modification of these models and semi-automatic production of customized reports that are essentially specialized "views" of the hyperbase (hypertext database).

This research intends to design a tool, IBISMod, to facilitate the process of synchronous collaborative decision-making and document the rationale behind these decisions. IBISMod is a multi user system implemented to run from a web browser which makes it platform independent. This feature enables the tool to be accessible from any place and any computer by authorized persons. It deploys AJAX to improve the usability of the system. In addition, it can be extended to run on small devices such as Personal Digital Assistance (PDAs) and cell phones.

2. The IBIS background

2.1 IBIS concept

IBIS was developed by Rittel as a method to "support coordination and planning of political decision processes" (Kunz and Rittel; 1970). The method provides clarification for specific design (Louridas and Loucopoulos; 2000). IBIS method is a way of finding solutions for complex or ill-defined problems that Rittel characterized as 'wicked' problems (Rittel and Webber; 1984). 'Wicked' problems refer to problems which are hard to specify and formulate due to complex social and human oriented nature of the problems (Purvis *et al.*; 1996). The understanding of participants from a 'wicked' problem evolves as more aspects of it revealed during the discourse. In addition, since these contexts involve social and human oriented activities, defining these sets of problems is also a 'wicked' problem. This results in different solutions for particular problem that come from variety of stakeholders with their own vocabulary and their hidden attitudes toward the problems (Mackenzie *et al.*; 2006).

As a model of structuring discourse, IBIS helps the process of communication in a problem-solving domain. It enables the system to capture different aspects of the problem from different viewpoints of interested participants. This approach improves the chance of selecting the best solution for a particular problem (Isenmann and Wolf 1997).

2.2 IBIS elements

There are three basic types of elements according to Rittel's IBIS system:

- *Issue*
- *Position*
- *Argument*

An *issue* is the main part of IBIS. It describes the problems that can be in the form of questions, which have been produced or taken into account during the discourse (Kunz and Rittel; 1970; Conklin; 1996). A *position* is an answer to an issue that helps to solve the issue. Each issue may have many positions. An *argument* is a statement that supports or opposes a particular position. Each position may have some arguments. IBIS represents various nodes (Issues, Positions, and Arguments) and their relationships with each other. Every argument points to at least one position and every position points to at least one issue.

The following is an example of a group decision-making process using the IBIS method in order to examine what would be an appropriate operating system in the context of developing a particular system. In this example 'I' stands for Issue, 'P' stands for Position, '+A' represents a supporting Argument, and '–A' represents a negative Argument.

I: Which operating system should be chosen to implement this project?

P: Microsoft Windows
+A: very popular
-A: not secure
-A: slow at times

P: UNIX
-A: command driven
+A: easy file sharing

P: Macintosh
+A: efficient memory management
+A: good UI

Relationships between nodes can be defined according to Figure 1. It shows all the authorized relationships between IBIS elements. Several Issues can be generalized into a single Issue or one particular Issue can be specialized into many Issues. A new Issue can be raised in association with a present Issue, Position, or Argument. The relationship of this type of Issue with the existing node is presented by *is-suggested-by*. Positions respond to Issues. An Argument can either support or object to a Position.

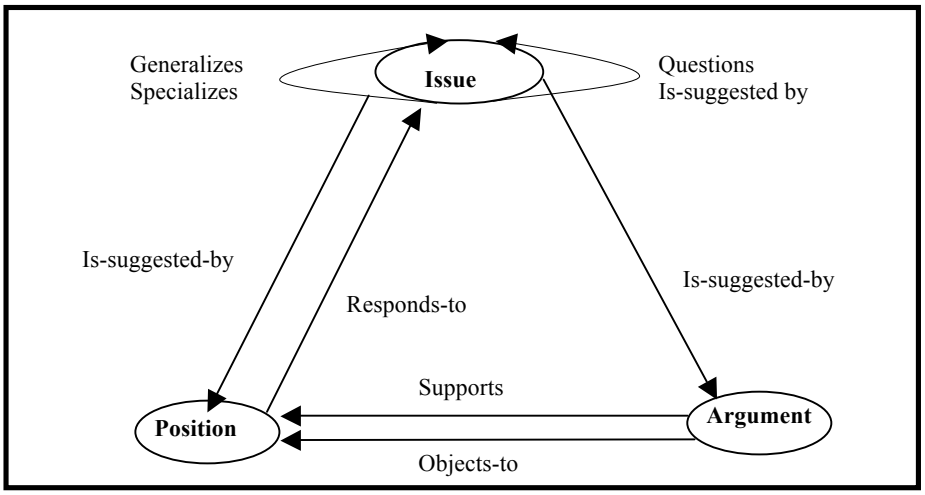

Figure 1. Authorized IBIS relationships.

3. An example

In this section an example of issue deliberation is explained. The example describes some of the design and implementation issues explored in the process of the making the IBISMod. Figure 2 shows a particular IBIS discussion structure of design Issues considered. Figure 3 provides a snapshot of the IBIS tool with more Issues and a bird's eye view. Bird's eye view facilitates scrolling to an area out of the local view.

I: What is an appropriate architecture for developing IBISMod?

P: Design a pure java-based tool (Ex. swing)

This Position has the following Arguments:

+A: There are plenty of open source applications that may be used
+A: High level of security can be achieved
-A: It is dependant on java compiler

The first two Arguments support the Position and the last Argument opposes the Position. The decision can be further amended by adding arguments that support or oppose various positions. Therefore, other criteria and evidence can be brought into the document. As a result some more aspects of the problem might be revealed.
The second Position is:

P: Design a web-based tool using JavaScript

The second position states that the tool can be web-based and written using JavaScript (client side application). The following Arguments propose supporting ideas for this Position:

+A: Platform independent and can be run from any machine that has a web browser.
+A: Ability to be run on small devices

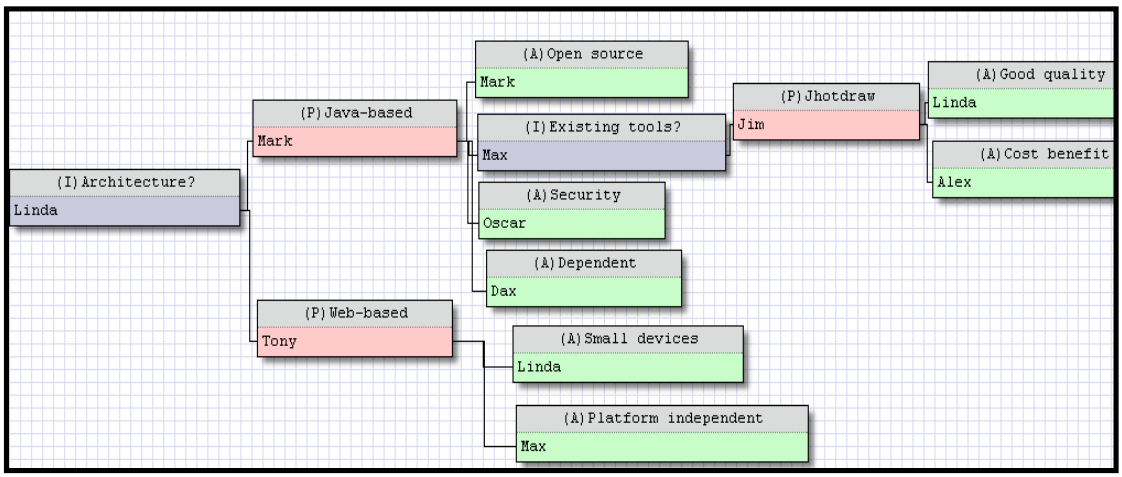

Figure 2. IBISMod design issues

These two positions are mutually exclusive. This is the most regular relationship between Positions. Moreover, an Issue may rise from a Position. In this example the following Issue rises from the "Java-based" Position:

I: Are there any existing drawing tools?

Instead of writing an application from scratch it is desirable to benefit from the existing java-based drawing tool for drawing the IBIS nodes. The relevant Position and Arguments are:

P: Jhotdraw (name of a drawing tool)

+A: cost benefit
+A: good quality

In our experience web-based tools written in JavaScript have been used in order to benefit from the feature of platform independency. Another aspect that motivated us to choose this path was the potential of extending our current framework to run on small devices such as PDAs, cell phones etc. This is because Mozilla mobile browser uses the same code base as the desktop Firefox browser. This feature enables participants to contribute in process of group decision-making from any place.

4. The IBISMod collaboration tool

IBISMod is a tool for knowledge sharing and capturing the design rationale. It allows members of the Design team engaged in deliberation of requirements to record their different attitudes towards the problem into a document. As the design deliberation proceeds, the participants define different Issues, Positions and Arguments. Moreover, it facilitates the concurrent working of multiple participants on the same model.

4.1 Functionalities

The IBISMod provides a visual presentation of the IBIS graph structure. Nodes and their links are displayed on a canvas of unlimited size. Different parts of IBISMod are shown in figure 3. There are three parts on top of the canvas: Design, Node and Import/Export. On Design part, there are some buttons: Create Node, Delete Node and Clear. Create Node adds a new node to

the model. Delete Node shows an alert message and if the user confirms it then the selected node would be removed. Clear provides a clean canvas for creating a new model. In addition, the title of the node should be completed in this part. On Node section, certain information should be completed. This includes Type (type of node, i.e. Issue, Position or Argument), Author (name of the author) and Note (node information in detail). Nodes with different types are shown with different colors. Date and time of node creation are automatically saved but are not shown here.

A node can be selected by clicking on it. The user may edit the node information on the Node section. It is also possible to change the position of nodes. The user may drag a node and drop it in a new position. After repositioning the nodes, the links are automatically redrawn.

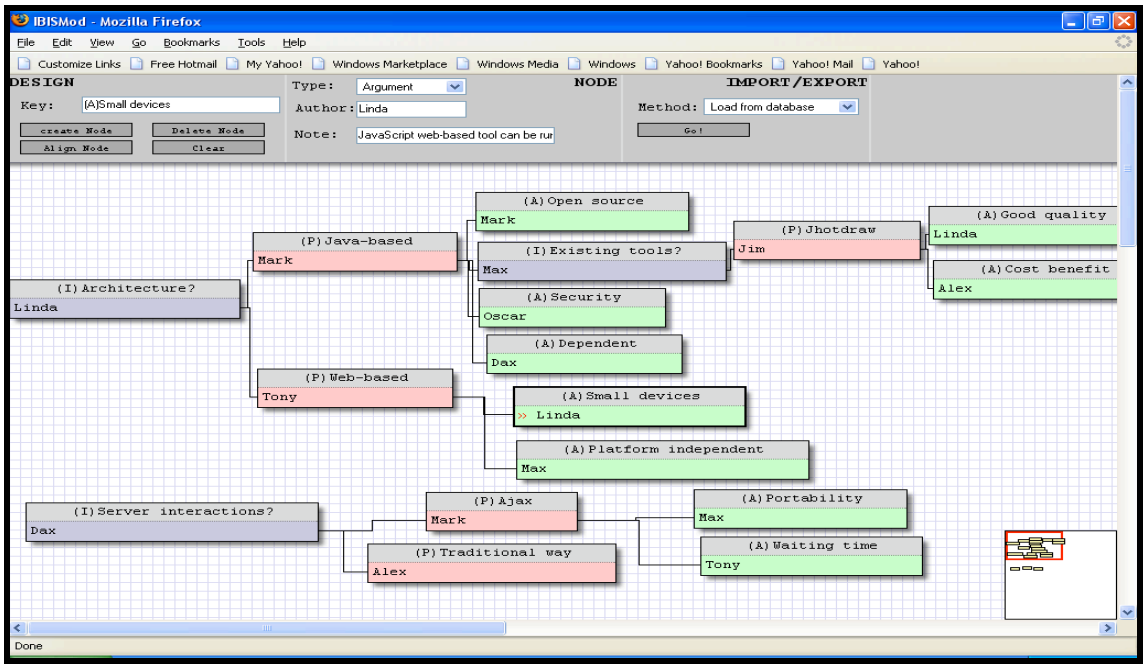

Figure 3. IBISMod tool

Import/Export section includes a popup menu. This menu contains some options including *Export to XML* (shows the XML format of the current model), *Import from XML* (creates a document from XML data), *Save to database* (submits the current model and saves it into database) and *Load from database* (opens a model).

In contrast with most web-based tools that have difficulties to run on variety of browsers this tool has been adapted to be run on most popular web browsers such as Internet explorer and Mozilla Firefox. In order to provide the same functionality to different browsers and browser versions, a technique called *code forking* has been used. For this purpose, first the browser of each participant is detected and then based on the browser information, code to handle the particular browser is written.

Another feature is that Scrolling through the document is very easy using the bird's eye view and also scrollbars. Even for very large graphs, it is easy to scroll to an area outside of the local view of the browser by dragging the small rectangle of bird's eye view to a specific area. This has been positioned on the right bottom of the canvas.

4.2 Architecture

The IBISMod tool under development employs a client-server architecture so several users may synchronously contribute to work on the same model by using the IBISMod tool. The client side application has been written in JavaScript. It uses AJAX methodology to produce an interactive environment. Server side application has been written in Java. Tomcat is used as the HTTP server and MYSQL server database has been used as the database server. Java Database Connectivity (Sun Developer Network; 2005) is used to connect to the MYSQL database.

4.2.1 AJAX methodology

Traditional synchronous (postback-based) communication requires a full page reload every time data has to be transformed to or from the server. A full page has to be reloaded even if a small part of the page is changed. So that is time consuming. Moreover, user interaction with the application is interrupted every time a server call is needed.

AJAX (Asynchronous JavaScript And XML) introduced by Garrett (2005) is a web development technique for creating interactive web applications. In AJAX applications a JavaScript-based program is created that runs on the browser (AJAX engine). The engine serves as a mediator that exchanges the data with the server in the background. The core idea behind AJAX is the asynchronous communication with the server. So data is transferred and processed behind the seen. This feature enables the user to continue to interact with the application like desktop applications while the request is being processed.

AJAX uses a combination of several technologies listed below:

- Standard-based presentation using XHTML (Extensible Hyper Text Markup Language) and CSS (Cascading Style Sheet). CSS is a declarative language for creating style sheets that specifies the rendering of HTML and other structured documents.
- Dynamic display and interaction using the DOM (Document Object Model). The Document Object Model is a platform and language-neutral interface that enables programs and scripts to dynamically access and update content, structure and style of documents.
- Data transition and manipulation using XML and XSLT (EXtensible Stylesheet Language Transformations). XSLT is an XML-based language that is used for the transformation of XML documents.
- Asynchronous data retrieval using XML HTTP Request.
- JavaScript that combines everything together.

4.2.2 Integration

Figure 4 shows the lifecycle of the IBISMod. Assume that a user has created an IBIS model. The IBIS model shown on the browser of each participant is stored in DOM. Then the JavaScript program converts the data to XML format and sends it by making a request to the AJAX engine (refer to appendix A). The AJAX engine then sends the request (that can be ‘Get’ or ‘Post’) to the server side program, which is a Java servlet.

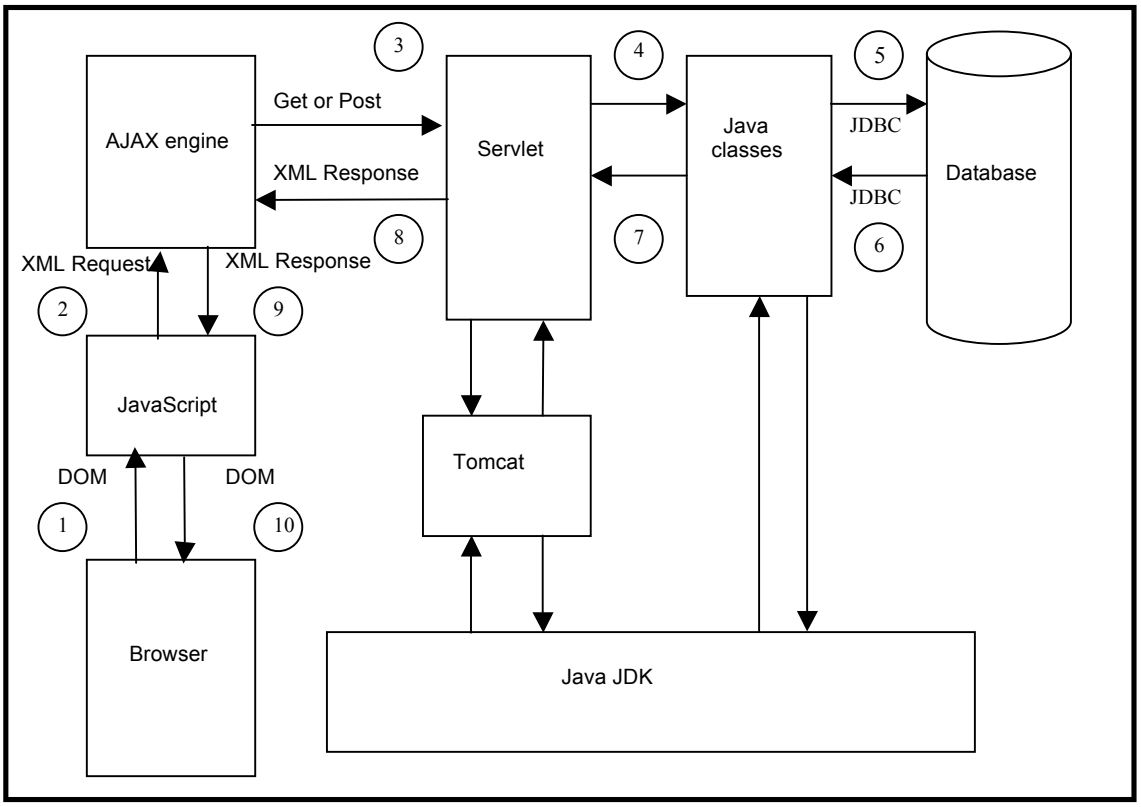

Figure 4. IBISMod lifecycle

A servlet upon the receipt of the request decides what to do. If the request is a 'Post' message, it saves the data into the MYSQL server database or updates the user's model. If the request is to save a document, the servlet assesses the request and checks if there is a document with the suggested name in the database. If the document is new it creates a new model in database. Otherwise it saves the new changes to the existing file into the database. If the request is to update the user's model then user's model is merged with database model. If the request is a 'Get' message, the information of the requested model will be retrieved from the database. Then, the servlet produces an XML response. The XML response is sent to the AJAX engine (on client side). AJAX engine submits the response to the JavaScript program. JavaScript program updates the DOM and the response is shown on the client's browser.

5. IBISMod Implementation

The main page of tool is a HTML page that imports some JavaScript files and a Cascading Style Sheet (CSS) file. The JavaScript files provide most of the functionalities of the tool and CSS file specifies the appearance of the HTML elements present in the main page. IBISMod is run on the web browser after the deployment of the appropriate JavaScript and HTML files in the Tomcat server.

The background of the canvas is an image. Nodes are s elements with borders. The shadow of the node is an image. The shadow is not shown in Internet Explorer. The elements of the bar on top of the canvas are also s elements that have been set on top of the browser using CSS. Bird's eye view is a set of elements each representing one node. To scroll to

a particular area, mouse movement from left and top is calculated. Then the view is updated based on the amount of mouse movement to match the particular area.

5.1 Synchronization

Participants may add, edit or delete nodes to any document. A node can be edited or removed by its creator. Synchronization is achieved by updating each participant's model if the same model in the database has been changed. When the database model changes, all users that are working on the same document concurrently receive a message that indicates the original model has changed. The system asks the users if they would like to update their model. If so, their models are updated. If not, user can keep working on her model.

If the user accepts to update her model then the user's model is merged with the database model, so that the user has the latest version of the document (including her changes). But if two users have been working on the same node concurrently a conflict might occur. A conflict occurs when several users are editing the same node of the document at the same time and when the changed nodes have a link to another node in updated model in the database. There is a greater chance of losing more data if the user does not update her model because she has not been aware of changes made by other users.

5.2 Merging method

For the merging purpose several conditions should be considered. The following is the pseudo-code for merging the user's model and the database model.

```
Merge(old_DB,current_DB,current_model){

        DB_deleted_nodes = Compare_oldDB_currentDB(old_DB,current_DB);
        Compare_oldDB_userDB(old_DB,current_model) {
                Add_node(new_nodes);
                Remove_node(removed_ nodes);
                Change_node(changed_nodes);

        }
}
```

The function *Compare_oldDB_currentDB(old_DB,current_DB)* compares old database model (the initial model that user had downloaded) with the current database model in order to list the removed nodes since user has downloaded the model. This list is called *DB_deleted_nodes*. The *Compare_oldDB_userDB(old_DB,current_model)* compares the old database model with the user's current model. It produces three lists including *new_nodes, removed_ nodes* and *change-nodes*. *new_nodes* contains all new nodes that user has added to the model. *delete_nodes* includes the nodes that user has removed. c*hanged-nodes* contains all nodes that user has changed. *Add_node* adds the new nodes *(new_nodes*) to the database model. If the parent node has been removed from the database model (if parent node exist in *DB_deleted_nodes*) then the new node and its parent are added to database model. In this case the current user would own the parent node. If the parent node has changed then associated nodes are ignored (are not merged with the database model). *Remove_nodes* checks the database model and if the user's removed nodes (*removed-nodes*) do not have a link to another nodes in the database model and the user who has removed those nodes is the creator of those nodes then

these nodes are removed from database model. *Change_node* checks the database model and if the changed nodes (*changed_nodes*) do not have a child and also the user who has changed these nodes is the owner of these nodes then these nodes are changed in database model.

6. Scenario-based evaluation of the IBISMod

Users run the IBISMod tool on their own machine and then may download the relevant model. Users work on their own copy of the document and they may add, delete or change some nodes. When the users finish working on their document they may save their copy. Here using an example we demonstrate and evaluate how we create and record a distributed decision making process using our collaborative decision making tool. For this purpose we use an example of choosing the appropriate operating system for project implementation.

Case 1.

Alex and John download a copy of the model that is shown in figure 5 from the database and start to work on it simultaneously.

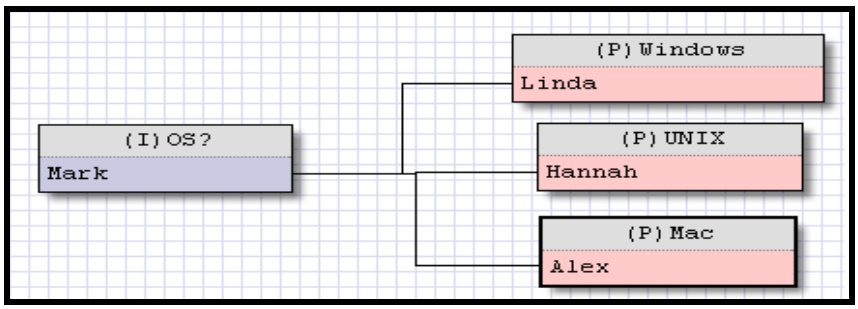

Figure 5. Initial model

Alex adds the "Not secure" Argument to "Windows" Position and also deletes the "Mac" Position (figure 6). John adds "Popular" Argument to "Windows" Position and "Good UI" Argument to "Mac" Position (figure 7).

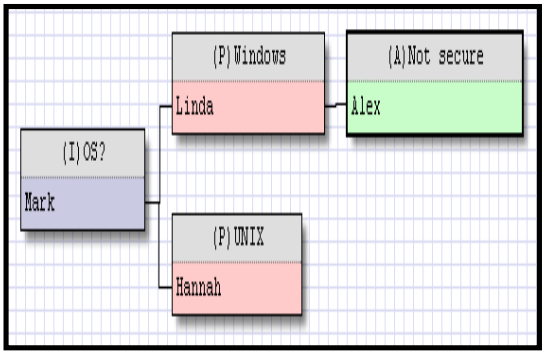

Figure 6. Alex's modified model

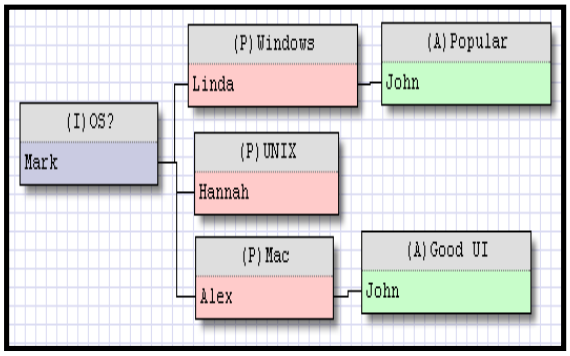

Figure 7. John's modified model

Alex saves his model while John has not saved his model yet. "Not secure" is a new node. So it is added to the database model. Also "Mac" is removed from the database model. The model in

database would change to the model shown in figure 8. Then John receives a message that suggests an update and John accepts to update his model. The new node is added to john's model and he owns the "Mac" Position. Updated model is sent back to the John (database model does not change). Figure 9 shows John's updated model.

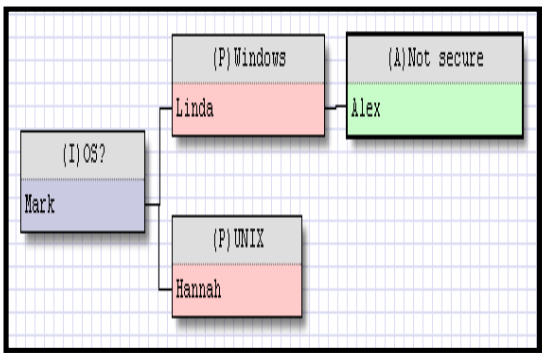

Figure 8. Modified database model

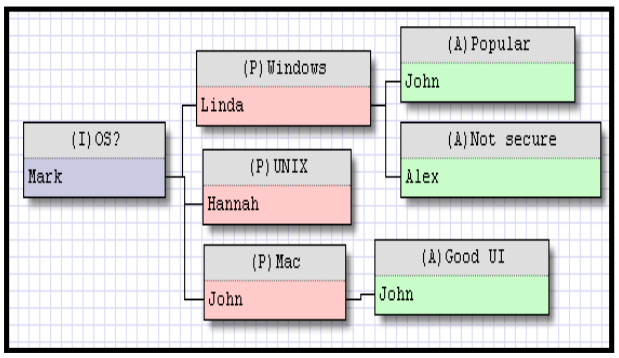

Figure 9. John's model

Then John adds the "Easy file sharing" Argument to "UNIX" Position (figure 10). Then he saves his model. At this time the new nodes are added to database model. Figure 10 shows latest version of the model in database.

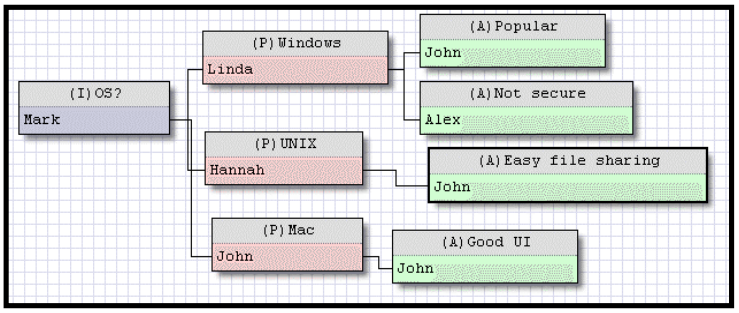

Figure 10. Latest database model

Case 2.

The following example shows how a conflicting situation may arise. Alec and Hannah download the model that was shown in figure 5 and start to work on the model concurrently. Alec adds "Command driven" Argument to "UNIX" Position and Hannah deletes the "UNIX" Position and adds the "Memory management" Argument to "Mac" Position. Figure 11 and 12 show Alec and Hannah's models respectively. In this case, Alec saves his model first. New node is added to database model. Figure 13 shows the updated database model. A message is sent to Hannah and suggests her to update the model. She accepts to update her model. The "UNIX" node that is deleted by Hannah has a link to another node in database. Therefore, this action is ignored and "UNIX" including its sub-nodes are merged with Hannah's model. Hannah's updated model is shown in figure 14.

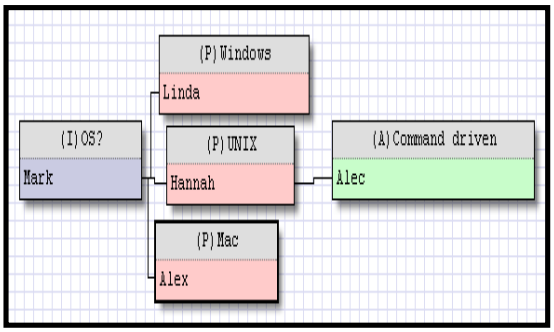

Figure 11. Alec's model

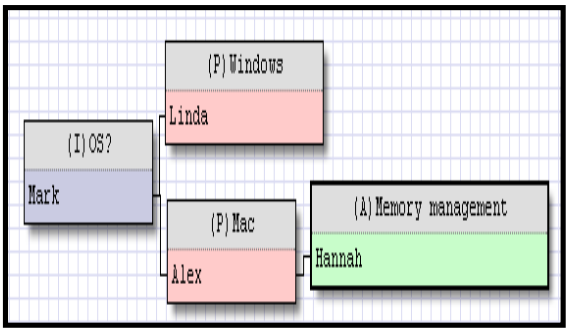

Figure 12. Hannah's model

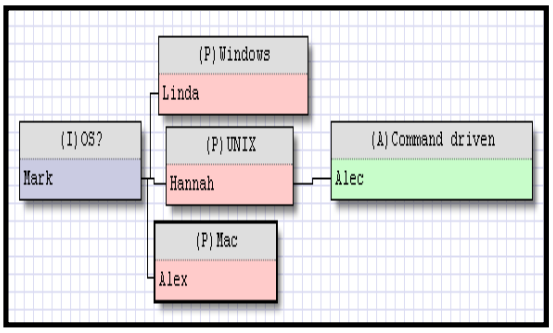

Figure 13. Database updated model

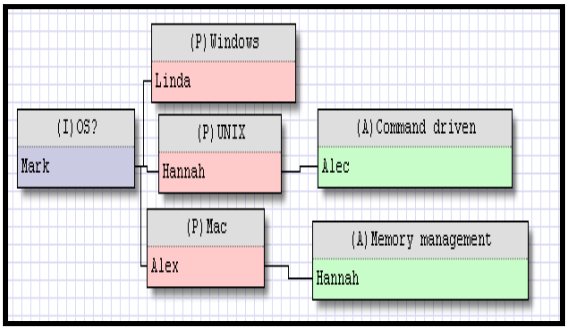

Figure 14. Hannah's model

Now Hannah saves her model. Figure 15 is the latest model stored in database.

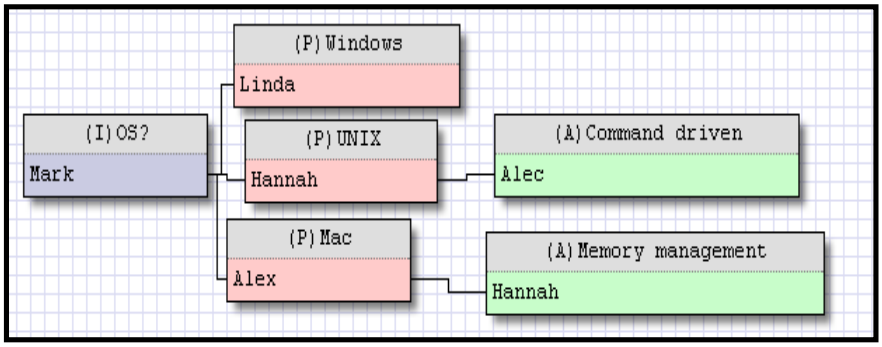

Figure 15. Latest version of database model

Conflict happens because "UNIX" Position has a sub node and if Hannah deletes it, the sub node would be deleted too. If it is being possible to delete nodes that have sub nodes, we may lose lots of useful nodes regardless of the effort that has been made to construct them. Therefore, the ordinary users are not authorized to delete a node that has a link to another node.

IBISMod allows privileged users like the administrator of the system to delete any kind of node for the purpose of maintaining the document. In this case, the system sends an alert to warn that this node has some sub nodes. The administrator can then decide to delete or retain the node.

7. Conclusion

It is generally accepted that IBIS-like systems are appropriate instruments for a group-oriented problem solving in situations like planning and design. The focus of this paper is on the design and development of a tool that can be accessed by clients from anywhere using any platform. Furthermore, the tool supports synchronous collaborative decision-making processes. In addition, this tool has attempted to improve the response time of the system by using AJAX approach. This is particularly beneficial while working with large IBIS graphs that take a long time to get the response page.

8. Future work

The IBISMod can be further expanded to improve scenarios that involve conflict resolution. This can be achieved by locking the changing node and its dependant nodes. The IBISMod tool can be enhanced to include a mechanism for calculating winning Positions. It is also desirable to extend and test the implementation of the framework domain to include the hand held devices like mobile phones. The tool can be further enhanced so that it is possible to locate nodes that meet certain criteria. Such as highlighting all issues that were generated after certain date.

9. Acknowledgements

We gratefully acknowledge Ondrej Zara, in OpenLinkSoftware Company for his open source tool and friendly advice. We also appreciate Tony Savarimuthu and Mark George, in the Department of Information Science at University of Otago for their helpful advice and comments.

References

Conklin, J.(1996). Designing Organizational Memory: Preserving Intellectual Assets in a Knowledge Economy [On line]. Touchstone Consulting Group, Inc. Available from: http://www.touchstone.com/tr/wp/DOM.html. [20 January, 2006].

Conklin, J.(1996). The IBIS Manual: A Short Course in IBIS Methodology. [On line]. Available from: http://cognexus.org/id26.htm.

Conklin, J.(2001). Sense-Making and Knowledge Collaboration Tools [On line]. CogNexus Institute. Available from: http://www.cognexus.org/sensemaking.doc. 2006].

Conklin, J. and Begeman, M. l. (1988). "gIBIS: a hypertext tool for exploratory policy discussion." ACM Transactions on Information Systems 6(4): 303-331.

Garrett, J. J.(2005). Ajax: A New Approach to Web Applications [On line]. adaptive path. Available from: http://www.adaptivepath.com/publications/essays/archives/000385.php. [20 February, 2006].

Isenmann, S. and Wolf , D. R. (1997). IBIS—a convincing concept…but a lousy instrument? the conference on Designing interactive systems: processes, practices, methods and techniques, Amsterdam, The Netherlands, ACM Press,

Karacapilidis, N. and Papadias, D. (1999). "Computer Supported Argumentation and Collaborative Decision Making: The HERMES system." Information Systems 26(4): 259-277.

Kunz, W. and Rittel, H. W. J. (1970). Issues as elements of information systems. Working paper. University of California at Berkeley**:** 1-9.

Lee, J. (1990). "SIBYL:A tool for managing t group Decision Rational." CSCW 90 Proc ACM: 79-92.

Louridas, P. and Loucopoulos, P. (2000). "A Generic Model for Reflective Design." ACM Transactions on Software Engineering and Methodology 9(2): 199-237.

Mackenzie, A., Pidd, M., Rooksby, J., Sommerville, I., Warren, I. and Westcombe, M. (2006). "Wisdom, decision support and paradigms of decision making." European Journal of Operational Research 170(1): 156-171.

Mylopoulos, J., Borgida, A., Jarke, M. and Koubarakis (1990). "Telos: Representing knowledge about information systems." ACM Transactions on Information Systems 8: 325-362.

Purvis, M. K., Purvis, M. A. and Jones, P. (1996). A Group Collaboration Tool for Software Engineering Projects. International Conference on Software Engineering:Education and Practice, IEEE, 362-369

Ramesh, B. and Dhar, V. (1992). "Supporting Systems Development by capturing Deliberations During Requirements Engineering." IEEE Transactions on Software Engineering 18(6): 498-510.

Rittel, H. J. and Webber, M. M. (1984). "Planning problems are wicked problems." development in design methodology: 135-144.

Rittel, H. W. J. and Webber, M. M. (1973). "Dilemmas in a General Theory of Planning." Policy Sciences: 155-169.

Sun Developer Network.(2005). JDBC Technology [On line]. Available from: http://java.sun.com/products/jdbc/index.jsp. [8 December, 2005].

Veerman, A. L. and Treasure-Jones (1999). Software for problem solving through collaborative. Foundations of argumentative text, Amsterdam, Amsterdam University Press., 203-230

Yakemovic, K. C. B. and Conklin, J. (1990). Observation on a Commercial Use of an Issue-Based Information System. Computer Supported Cooperative Work, Los Angeles, Association for Computing Machinery,

Appendix A

AJAX engine

AJAX engine has been written in JavaScript and runs on the client side. The JavaScript code for the AJAX engine is as follow:

```
function ajax_command(method, target, data_func, return_func) {
        var xmlhttp = new XMLHTTP();
                var callback_response = function() {
                if (xmlhttp.getReadyState() == 4) {
                if (xmlhttp.getStatus() == 200) {

                                return_func(xmlhttp.getResponseText());
                        } else {
                                xmlhttp.error("problem retrieving data");
                        }


                }
        }
        xmlhttp.setResponse(callback_response);
        data = data_func();
        switch (method) {
                case GET:
                        var newtarget =
                                (/\?/.test(target) ? target+"&"+data : target+"?"+data);
                        xmlhttp.open("GET",newtarget,true);
                break;
                case POST:
                        xmlhttp.open("POST",target,true);
                        xmlhttp.setRequestHeader("Content-Type",
                                                "application/x-www-form-urlencoded");
                break;
        }
        xmlhttp.send(data);
}
```

The AJAX engine is provided by the *ajax-command* function. It produces an *XML HTTP Request* object and opens a connection with the server. *Xmlhttp.getReadyState()==4* indicates that a response is received from the server and *xmlhttp.getStatus()== 200* represents that the request has succeeded. If both the conditions are satisfied then the response is complete. *xmlhttp.setResponse(callback_response)* checks the response and if it is complete triggers the callback function. Callback function is a function that is called when a response is received. Then the *xmlhttp.open* function sends an asynchronous request to the server. An asynchronous request does not wait for the server to respond. The request is sent and then the application continues with its next task. Users can still work with the tool, click other buttons or even leave the tool. The server quietly responds to the request and when it is finished, then it would allow the original requestor know that it has been done. The end result is an application that is responsive and interactive.