Intertac Software Architecture

Cat Kutay
School of Computer Science and Engineering
University of New South Wales
Sydney 2052 Australia
E-mail: ckutay@cse.unsw.edu.au
UNSW-CSE-TR-0318

June 13, 2003

Abstract

This paper describes the development of a groupware system from the requirements developed through researching the activities of software enginering students who were developing specification reports in groups. The specification is designed for an imaginary, but realistic, *client*. The groupware was developed to enable these groups to meet more often in non-collocated sessions. A list of requirements that were developed for the basic application software are presented here, together with the Architecture and Interface.

The groupware is designed as a Constructivist and Collaborative Learning Environment (CLE) so the first aim is to provide a flexible and unstructured learning environment in which students can construct their own meaning. On top of this can be placed agents to provide assistance and feedback to improve aspects of this learning. This paper looks at the first part of this process, developing the environment with a component-based architecture to which agents can readily be integrated. Also brief summary of the agent support is provided, with a plan for future verification of the final system when complete.

1 Introduction

The groupware, namely Intertac, is designed to draw together a group of human minds to enable learning through interaction, using a computer as the medium. The computer software provides the basic blending of students' contributions augmented by the effects generated for the specific learning domain by a system of agents to guide the process of that learning. The research is based on the approach that the computer is a medium, not an end point of the interaction.

While HCI issues are considered in developing the initial design of the basic software the development of agents in based on Human-Computer-Human interaction or HCH. HCH is about removing the role of the computer as the intelligent agent and reducing its role to that of a mixer, with the ability to insert adaptive electronic components (or agents) that add extra effects and depth to the product of the human-human interactions.

HCH is about ensuring that the feedback from the system assists the students to improve their input to the group, that their contribution is as harmonious or discordant as necessary, and their understanding of the process and their roles in that process improves over time. The software is used to support members in adapting roles in meetings, to enable the group members be aware of each other and their contributions and finally to try and focus the group on the whole problem that they are dealing with, not their individual tasks.

The users are not groups with years of skills, they are still practising their group skills and their learning skills, as much as they are practising the skills of subject domain in which their studying is situated. They are apprentices and their learning must be guided.[1]

This paper presents the process of design that was used in developing Intertac-I, the initial basic Intertac system. The next stage of development of Intertac-II involved augmenting the software with the design and implementation of agent support andis summarised here, but will be presented in later papers in more detail. The design of Intertac-I was done with the aim of presenting some standards for the development of groupware. We first describe some unique features required for Intertac-I, which did not exist in systems available at the time of Intertac's development and which necessitated the development of new software.

Then we describe the requirements which were developed using Activity Research (as presented in Fjeld et. al. [6] and Jonassen & Rohrer-Murphy [10]) to analyse students in groups during a software engineering workshop course.

[1]These ideas are developed in the learning theory Cognitive Apprenticeship as developed by Collins et. al. [4]

2 Design of Constructivist Learning Environments

Years of research of learning in these modes has suggested some guidelines for course development based on constructivist and collaborative principles. To develop a design for a learning environment based on these principles, we must in the first instance adhere to the established guidelines for the development of such a system, which were originally derived for the classroom. Some of these have been applied, or can be applied, to software based courses such as when courses are part of distance learning modules.

The key characteristics for a Constructivist learning environment have been developed in Grabinger et. al [7], Jonassen [9] and Suthers [19] and have been extracted to form a set of Learning Requirements as follows:

LR1 Allow student responsibility for learning and self-directed learning, identifying their own knowledge, manage their own learning (conation).

LR2 Allow dynamic, generative learning. Knowledge is actively constructed by the learner, rather than being handed down. By manipulation of the learning space with mindful activity, meaningful learning is achieved (constructive).

LR3 Provide complex contexts which include the physical, organisational and socio-cultural contexts of the problem. The context is to be described in the problem or project and may be represented in the software. This should include the context to which the solution must fit, as well as the expectations and customs of the other stake-holders (complex).

LR4 Provide authentic contexts by using realistic problems and examples that must be interesting, appealing and engaging (authentic).[2]

LR5 Allow ownership of problem by the student. To allow this the context should be changeable (motivation).

LR6 Provide collaborative opportunities that allow exposure to multiple perspectives. Students are required to reflect verbally in their group and to examine other understandings. Develops cooperative skills (conversation theory).

LR7 Reduce bias in the learning derived from the limited representational formats offered to students to enable them to manipulate and converse about their knowledge (representational bias).

LR8 Allow reflection through the articulation of what has been learnt and how. Understanding the individual learning process. Feedback to planning next phase of learning (action theory).

These learning requirements can be developed to some extent in a software based course. We are suggesting the following software

[2]These tasks can be developed using the analysis techniques of of Activity Theory.

learning requirements for Intertac-I to realise the above learning requirements:

SLR1 — User Interface The objects that represent the learning content, such as documents or diagrams should be able to be manipulated and extended or restricted to match their interests (LR5).[3]

SLR2 — Constraints Reduce the constraints on the student's representation of their knowledge as a resource for conversation. Provide a variety of primitive elements both textual and diagrammatic that the user can manipulate for different logical representations of their knowledge (LR7).

SLR3 — Salience Increase the salience or automatic perception of information contained in users' representations by providing different types of representations that visually express different knowledge relations (LR7).[4]

SLR4 — Planning Planning and reflection tools should be provided and integrated into the system. Include Action Plan templates for groups to manipulate (LR1). Enable selection of roles and other individual approaches with support for each role (LR1). Visual schema of progress can be made accessible to student for review (LR8).

SLR5 — Integrated System Primary elements of the software interface and the items of domain knowledge presented in the system should be interactive and able to be linked to each other (LR2). The tools should provide support for research through the web (LR2), or other students' solutions (LR6).

SLR6 — Enable Multi-user Contributions Enable changes from different computers to be presented to each user's interface for updating the screen in a *synchronous* manner (LR5). Enable users to edit and annotate others' work (LR6). Where necessary allow semi-synchronous updates where editing changes to the user interface are updated on each screen in bulk, rather than step by step (LR4).

SLR7 — Asynchronous Editing Also allow users to work alone in an asynchronous mode on Intertac tools (LR4).

SLR8 — Component-based Architecture Enable users to use their own tools in the system, possibly linking different editors through the server to provide suitable interfaces to different users' needs (LR5).

SLR9 — Chat Interface Provide a tool to enable a *chat channel* to be set up (LR6). Chat can be augmented with tokens to

[3]For example in the present learning domain being studied, the groups can change the requirements to restrict the problem to a reasonable size, or individuals can select which part of the report they write, or concentrate on.

[4]For example DFDs can be displayed as separate levels to show structure or as a single complex level to follow data flows.

describe each speech act, and topics to denote the context of the contribution (LR4).[5] Contributions should be linked with any previous contribution to which they refer (LR6).

SLR10 — Feedback Assessment should be integrated into the project with embedded learning appraisal activities, such as checking rules and regulations of the domain (LR8). The software should include functionality for the instructor to develop guidelines that are able to be adapted by the users' software or tools (LR8).

SLR11 — Agent Support Sufficient depth to the problem must be offered, then provide sufficient scaffolding to allow the student to work at their own level (LR3).[6] Ensure discussion is situated in the current learning context and follows optimal learning approaches (LR8).

2.1 Review of Basic Groupware Requirements

Given the Software Learning Requirements above which are based on learning requirements, there are additional requirements for working in remote groups. These requirements were developed from the initial observations of collocated meetings during the *workshop course*, but address the features of the context that are either not available when these groups presently work remotely, using the phone and existing editors with computerised file transmission; or are now feasible given the computer medium on which the communication is taking place.

The Groupware Requirements are divided into Functional and Non-Functional Requirements. Functional Requirements are those which capture the intended behaviour of the system, or what the system does to fulfil the needs of the user, and include all the Software Requirements listed above. This behaviour may be expressed as services, tasks or functions the system is required to perform. Non-Functional Requirements are system qualities that characterise when and how the system fulfils user needs.

Functional Requirements :

FR1 Support multi-user actions over a visual workspace and enable easy natural interaction for participants. This includes the ability to use collaborative drawing and marking tool similar to those used in face to face meetings with pen and paper. On top of this can be laid extra features that a computer can provide, such as removing and moving structured objects around the screen.

FR2 Support the handling of textual and graphical primitive objects that provide concurrency[7] of different users' views

[5] Users can be provided with menus of topics an tokens to frame their contributions to the discussion.

[6] In this course the context to which the solution must fit is the *needs of the client* and communication of the group's proposed design to the client.

[7] All users see the changes to the primitives that appear on the interface more or less as they occur and in the same order.

and WYSIWIS[8] view sharing.

FR3 Use repositories for files that are local to each user. Provide *versioning* support for each user.

FR4 Enable easy analysis of the interaction. This would preferably be enabled through logging facility that would accommodate analysis that steps through the history of the interaction.

FR5 Provide natural interaction with agents that play the part of a mentor or tutor. This can be in the form of separate *advice* screens or separate tools that presents a similar interface to that provided when working with peers.

FR6 Support structuring of the group processes during a meeting. The format of this structure will vary with the needs of the group.

FR7 Integrate with the present, conventional ways of doing this work, such as the phone and other programs.

FR8 Provide possible compatibility with other applications, through text based output files for all tools.

FR9 Provide awareness of other users logged into the session.

FR10 Support rough editing comments overlaid on the formal diagrammatic and textual tool interfaces.

Non-Functional Requirements :

NFR1 Avoid high bandwidth use by avoiding audio links. While having an audio link would overcome many of the difficulties involved in communications over a distance, it would slow down the rate of visual communication to an unacceptable level and create problems when synchronising screens.

NFR2 Simplify communication speed and detail between users by transmitting only text based updates to document contents.[9]

NFR3 Provide a robust communications structure that supports persistent sessions (especially if FR3 not provided), or at least refreshing of users' screens with latest views of the tools when required.

NFR4 Provide rapid response from server by minimising logging and agent support at server, relaying more on client software for this.

NFR5 Provide tools for software engineers that resemble the tools with which they are familiar, and hence they find it easier to work with.

NFR6 Analyse interactions in this domain requires dealing with design problems that are ill-defined and unstructured.

NFR7 Develop advice agents that will have to work from data collected in the above analysis and so face the same difficulty of the unstructured nature of the input to the analysis.

[8] What You See is What I See.

[9] Diagrammatic objects can be described as a string for transmission.

NFR8 Provide technical support for multiple and distributed processes on separate systems. Support for the software must deal with problems with rapid turn around. In a learning environment the users do not have time to wait for a system to be redesigned during their course.

When first reviewing these requirements to select from the existing groupware products, the solution was to access the screen display of an editor and broadcast it to all group users. In the most ideal application, the individuals in the group would be able to choose their preferred editor, and the shared document would be transferred and then translated to each separate user interface. Similarly the input to each editor would have to be caught and coded for the separate editors.

It has occurred to the author in this vain quest for such software that it would be dangerous to system security to have such a system available. Assuming these features could be attached at run time (since the editor functions already without them, the software must use existing input and output to do this *grab*), and worked over the network (since different computers would need to communicate) any user would be able to grab any process on any computer they like and redirect the input and output to their own computer.

2.2 Selection of Software

The initial step of this thesis was to study the existing groupware systems for compatibility with the Software Learning Requirements of this project. There are many existing systems and they have various distinct features however the software availabel at the start ofthis reserach, in particular Habanero [8], Matchmaker [11] and Teamwave [20], were not suitable for the users' needs due to lack of at least one of the following requirements:

1. Ability to plug in agents to the applications or tools to provide scaffolding to the learning process (SLR11). This applied to all commercial systems, unless we could get access to modify their source code.
2. Ability to link tools through the server that have equivalent content but may differ in their interface, and open and close tools locally to a single user as that user required (SLR1 and SLR5). This applied to all existing groupware as the separate user tools are too closely linked.
3. Ability to edit text files in a synchronous or semi-synchronous mode (SLR6). Many groupware systems do not supply a text editor due to the difficulty in providing synchronous communication of rapid edits across a long network.
4. Ability to plug in new applications as they become available, that resemble the editor and other tools commonly used by students in stand-alone mode (SLR8 and FR5). Again commercial groupware does not allow access to the code to enable this, although Group

Kit and Habanero do enable this and provide detailed information on how to do it.

5. Use event based messages to synchronise simple content changes through the server with local replication (NFR2). Event based messaging is used in most recent systems, however the link between applications is usually too strong, as in Habanero which transmits the entire document framework.
6. Ability to augment editor and diagrammatic tools to include different representational primitives to reduce representational bias (SLR2 and SLR3). Editors and diagrammatic tools were limited, and hence new tools had to be added whatever software was selected.
7. Server designed as a synchronous broadcast process and no more, with all interface storage and error logging done at the client end if required (NFR4). The closest to providing this was the Matchmaker server but this system runs extensive logging of events at the server.
8. Save files in individual local repositories for each user (SLR6 and NFR3). Again many recent systems use local repository but maintain logging and agents support at the server.
9. Provide reliable user support for the software system (NFR8). This was not possible for Habanero or Matchmaker.

We could not find a system to satisfy these requirements, so a new system named Intertac-I was developed. The process of developing this software provides a basis for the development of extension tools and agents for future groupware, so is summarised here. Also the basic process matches the type of process that we are trying to teach in the Software Engineering Workshops so should be implemented here also.

We did however re-use some software, both the Matchmaker [11] server[10] and the Java Swing classes, and developed Intertac-I from these.

3 Process for the Design of Intertac-I

In order to provide complete requirements for Intertac-I the future users of the system had to be studied in their present learning context to analyse what they needed from the system. Software design for human users must be studied in the social matrix in which the software will be used [21]. This is difficult to achieve, as the present collocated context is quite different in many respects to the final context of learning through groupware. Hence part of the analysis is to establish changes in the learning that will result from the change to distance mode.

Software design requires an analytical framework that includes group aspects and a conceptual underpinning. In this thesis Activity Theory

[10] Synchronous systems is the most difficult aspect of the programming, so it was decided to re-use software for this.

formed the theoretical basis. Guidelines for the design of the software in this thesis were developed as a Four Step Model based on the work by Soloway et. al. [17] where features of the design are listed as: context (for the software); tasks (that the software will perform); tools (required for each task); and interface (to tools).

Also the work by Markus and Keil [12] covers issues of process change within the organisation to enable the introduction of a computer system (task change). Suchman [18] looks at issues of representation in design and how these both become part of the final work practices and are often different from the viewpoint of different users. Nardi [15] gives an example whereby looking at activities and actions as distinct entities in design, the designers approach to understanding user needs was improved. This has a parallel in learning theory with the manner in which surface learning skills can be combined to form a deeper approach. The individual actions or skills can be automatically selected by the deep learner for their appropriateness to each software design situation [3, p.256]. Thus as they become an expert, they may appear to use templates or rule-based approaches, rather than deep approaches [2]. Yet this is denying the expert's use of intuitive decisions and use of plans [3, p.253].

Besides the case studies mentioned above, there has been little work in developing a design model for implementing Activity Theory in software design. Therefore a model was developed for this research for designing software for a learning context which involved four steps. Recently an Eight-Step-Model was hypothesised and tested for designing software using the Activity Theory model was developed in a PhD thesis by Mwanza [14]. The Four Step Model developed from work by Soloway et. al. [17] and used in this thesis is here divided into subsections that correlated to and expanded on the alternative model by Mwanza, to establish to links with her work.

Context :

Interface of System Describe the visual aspects as a whole.

Activity of Interest Describe the activity as whole.

Objective Describe the activity as part of a plan.

Subjects Describe the human agents of the activity.

Community Describe the larger learning plan.

Tasks :

Steps of performing Describe the tasks performed by the subjects including alternative ordering of tasks.

Division of Labour Are there separate roles involved?

Co-ordination of Labour How are the separate roles linked?

Tools :

Means of Performing Describe the way the subjects perform this activity including alternative tools.

Rules and Regulations Describe any rule or regulations for the performance of this activity.

Object :

Interface Focus of task in terms of input, the visual and manual link to system.

Outcome Focus of the task in terms of output.

Activity Change Effect of the software task on the original activity.

This model is used as an approach to augmenting the generic Software Learning and Groupware Requirements listed in earlier sections with aspects relating to the domain of Intertac-I. The more detailed requirements that were developed and implemented in the software were developed by using the above model to analyse each activity of interest that would be carried out on Intertac-I and thus extract the variations in tasks or operations for that activity. For instance if the activity is *decision making*, then we found:

Context will be the type of phenomena on which there is disagreement, the significance in terms of the overall design, whether more than two people are involved, etc;

Tasks will be such steps as adding and deleting parts of a diagram or report, repeated comments referring to the concept or phenomena being debated, roles taken in the debate, etc.

Tools will be pen and paper used in group edit, individual editing work then taken to the group, any other learning that is referred to in order to reach a decision such as diagrammatic rules, etc.

Outcome is an agreed diagram or file that is not edited for some period. The outcome will be changed little when using the software mainly in terms of enduring logs of discussion over the phenomena, but the activity itself is changed greatly by the use of Chat for discussion instead of speech.

These details are not dealt with here. However the specific results of the activity research in relation to the Architecture and Interface of Intertac-I as they are now developed, are presented.

4 Architecture

The architecture of Intertac-I was designed to handle the following aspects:

1. Front-end:

   (SLR5) Event-based messaging for maintenance of the synchronous interface.

   (SLR6) Updating in blocks to reduce the rate of updates.

   (SLR8) Component-based design for easy upgrading to incorporate the tools of the learning domain, such as concept maps, HTML editors etc.

   (SLR8) Common format of tools for easy analysis of activities across tools.

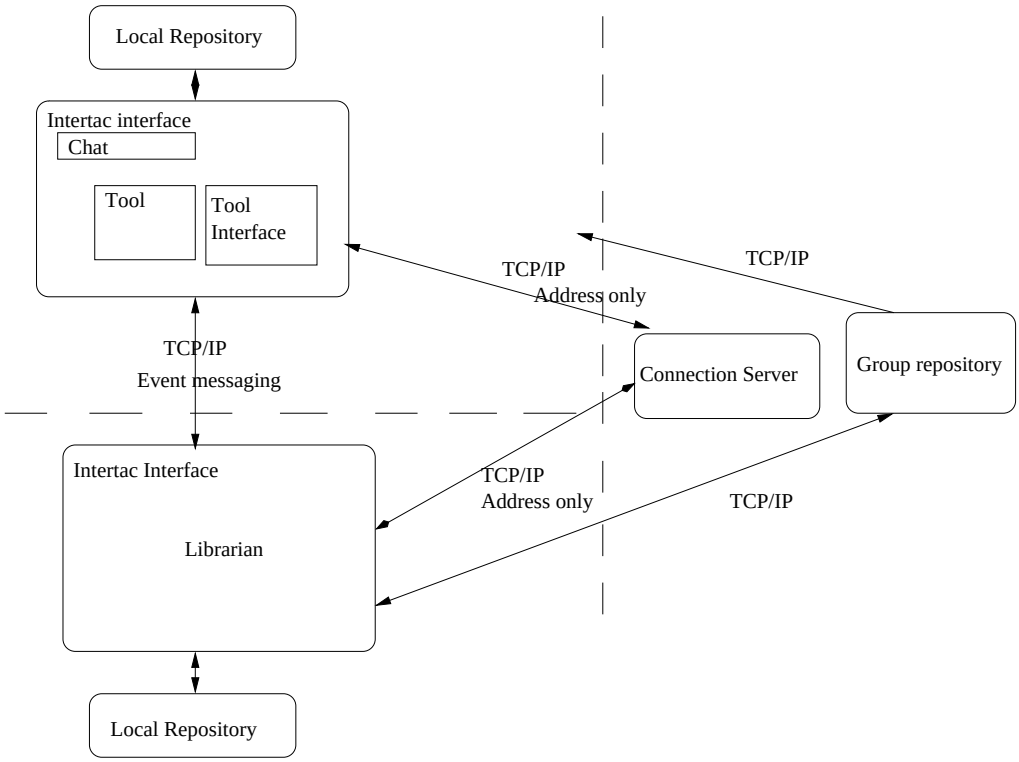

Figure 1: Architecture for Intertac-I System

2. Back-end:

   (SLR7) Compatibility allows for future compatibility with other external applications, such as individual editors, by using XML output from the Intertac-I tools.

   (FR3) Local repository for individual reflection and review.

The original design of Intertac-I was to have the tool as the main component to be manipulated. However with the need to adapt open-source applications to the system, Intertac-I was redesigned to make the Internal Frame the main component, as this was an easy adaptation of the Frame format of many tools. The tool is then added to, and manipulated through, the Internal Frame. Changes to the tools are shared as changes to the Internal Frame contents.

Intertac-I provides both synchronous and asynchronous sharing of files. One of the most crucial aspects of synchronous systems is the efficiency of exchange of change events, so that the individual user will maintain an almost continuous editing mode, while changes are added when possible from other users. This provides users with an artefact that they can discuss as they are being edited.

Intertac-I used a centralised server which maintained a tree structure of serialised representations of all the internal frame objects and the text or graphical objects in the frame. At any time a user could join

a *session* on the server and receive a broadcast of the current screen display. Changes can be tracked between different users and different times. Also the centralised log included details about which windows are selected by each user.[11]

The server originally used by Intertac-I was developed at the University of Duisburg [13] and maintains a tree structure of serialised representations of all the internal frame objects and the text or graphical objects in the frame.[12]

5 Interface

There are many existing groupware systems, with various features to suit their context of use. The main feature of the design of Intertac-I is its flexibility. The Intertac-I software design is based around a single window, with a panel overlaid for scribble drawing and notes. Within the main frame various internal frames should be able to be opened by the user on request. The internal frames should contain the tools for viewing text or diagrammatic files. The position of these frames should be selectable by the user.

The Intertac-I interface is based on a Java windows class, rather than rooms,[13] as this provides an analogy with the computer as a window onto the files being viewed and edited. This matches the user's concept or model of design tools used in the software engineering school. Also the use of Java as the programming language provides cross-platform functionality for the system. While the interface may not appear exactly the same on all computer systems, the software can recognise the software objects on any computer systems.

The main frame interface requires the following shared features:

1. *Awareness* Text Line to know which users are on-line (FR9).
2. Overlay window for rough notes and sketches that can be selected as visible or not (FR10).
3. Visible representation of the state of the Overlay On/off button (FR10).
4. Chat single line frame that shows the last line of chat to provide visual access to the interchange when the chat window is obscured (SLR5).
5. Overlapping interfaces to the Tools that have been opened for group view (FR1).[14]
6. Each user's contributions to the session to be distinguished e.g. by colour (FR9).

[11] This information can be stored on the client side as selecting a window (other than Chat) to edit, locks it to other users

[12] This server created many processes on the host machine, and also kept a log of changes transmitted. The resulting memory and file space used by the server caused problems. As a result a new server was written.

[13] Compare to Teamwave.

[14] Tools can also be opened in local mode only.

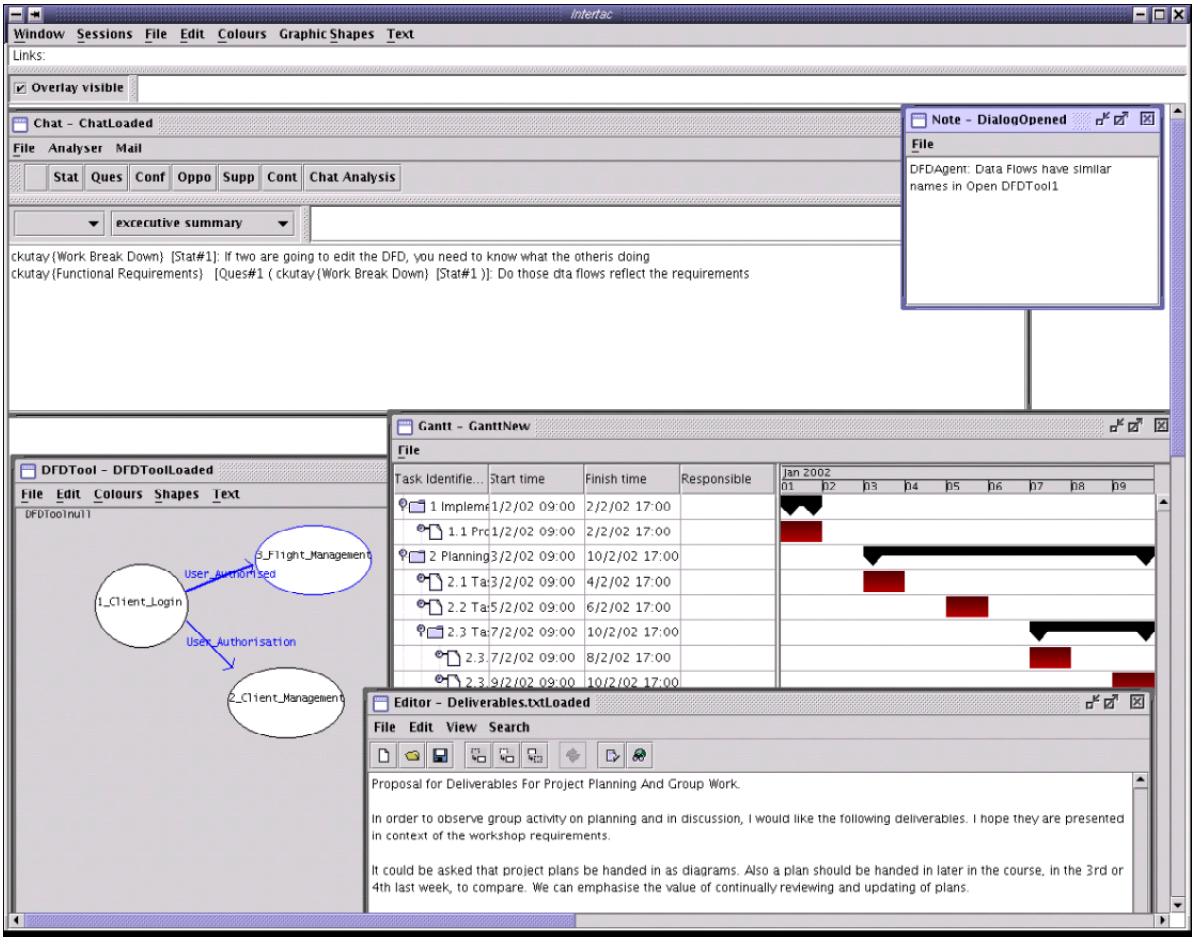

Figure 2: Intertac Interface showing Chat window, DFD diagram, Gantt chart, Editor and Agent feedback in Dialog Note.

7. Select on menu to start or join a group session (SLR9).
8. Select on menu for a role (FR6).
9. Separate tools available to users who selects a role (Facilitator, Record Keeper and Librarian) to assist these different roles (FR6).

Many aspects of the interface are set as variables in user files and can be altered at run time. The data repository is local so users may have different versions and work alone on their material. Even in group mode, the users can close or minimise windows independently of each other. Dialogs or feedback can be presented in single view mode, or group mode on all windows.

Aspects of the interface which can be altered by the user are:

1. Role selected by user e.g. extra tools are available for a facilitator, record keeper and librarian
2. Applications or files that are open e.g. files can be selected to open at start up, if these are the ones being worked on
3. Format of diagrammatical objects e.g. The number of text lines to be included as descriptors in planning or DFDs and their description can be altered

The requirements for the applications themselves were based on generic models rather than being designed for a specific domain. They

can however be extended with features adaptable to the domain of learning.[15]

6 Supporting Agent Development

The interaction and learning processes that were selected for study during experimentation on Intertac-I were:

1. increasing depth of learning;
2. group processes;
3. efficiency in document production; and
4. efficiency of learning.

Given the broad scope of interactions that fall under these headings, it was necessary to restrict the review to particular aspects that concerned the lecturers in these courses. The results of the experiements are published in later papers, however we describe here the softwrae included in Intertac-I to provide the data needed for analysis.

6.1 Translating Actions

In order for the work of students using Intertac-I to be analysed for suitable interventions by agents, a separate form of *Outcome* is created for the software. This attribute is reduced after the experiments, but it provided the text data needed to analyse the state of the students as they learnt.

From the initial design of Intertac-I it can be seen that the tools can provide data to the agents that enables the following:

1. Analysis of common threads running between the tools.
2. Checking the timing of document production or content of interchanges according to the stages as specified in the planner.
3. Comparing student documents and diagrams to a template where appropriate.[16]
4. Analysing documents for adherence to rules of the domain, such as graphical rules or limits on length and layout of text documents.
5. Analysing interactions in detail from the Chat tool and also across all tools.
6. Feedback comments to a common Notepad window, as a contribution to Chat, or as a Dialog Window that locks the interface until answered.

[15] For instance the original concept diagram application was extend to draw Data Flow Diagrams.

[16] A course planner can be maintained in unedited form as a guideline for fixed dates in the project. Similarly a document template can be retained to provide guidance on document layout and type of content.

Three aspects of learning as described by Dillenbourg and Self [5] can be supported by the agents using the above data in the manner described here:

Verbalising strategic decisions Observing conflict between action requiring explanation and feedback through dialog windows when additions and deletion patterns suggests conflicting actions.

Acquiring reflective skills Creating conflict between individual knowledge and experience requiring resolution by proposing new approaches to concepts through the common note window when groups or users reach a certain level.

Acquiring a better model of learning Observing conflict between different individuals' knowledge requiring resolution and feedback through dialog windows when conversation patterns suggest conflict.

While the students make various keypad and mouse actions during their interactions with the computer (or with their peers who are also on-line), the log of these actions can be quite meaningless unless the actions can be described in context.

For instance, an arrow linking two objects (a and b) is not just an arrow from point a to point b. It is a link between these two objects and whatever information they contain. In other cases the interpretation of the actions will be dependent upon the context. Some text in a circle may just be a way of highlighting text, or in the case of DFDs, it is a process and its name, by convention.

6.2 Translator

For all the tools there had to be a common output of data required for analysis. This data should include enough detail for the computer to verify various rules and patterns. This process was originally carried out by a translator in Intertac-I which provides data to a human agent to review. In Intertac-II this process was implemented in the basic agent structure to enable agents to verify their own rules. This division is made as the translator necessarily will be dependent in some aspects on the domain of learning that the software is being used for.

There are various ways that actions can be combined to form coherent acts of learning or acts of constructing meaning.

Construction Combining a sequence of actions (whether coincidental in time or not) to form a more conceptual description of an act.

Completion Updating acts to include previous related actions that now can be combined and understood as one act.

Alteration Updating previous acts as changes are made.

In fact it is not entirely appropriate to use the term *completion* for general updates in this context, as there is no completion until the final reports are handed in. The work of relating acts to each other (for instance a student may construct a process in a DFD then link this to another two processes, and thus form a level in the diagram)

is the part of translation or domain-dependent analysis that is left as part of future work.

6.3 The functionality of the translator

This work is based on the formalism developed by Akhras & Self [1] but takes a different approach. The design is for a specific domain, but since the domain-dependent components are separated from the Intertac-I design, it is assumed the system can be augmented to other domains.

The translator keeps a record of every action of the group members, including deletions. At any time an action is taken, this must be compared to other actions in terms of:

Proximity For example, start and end of an arrow, or position of inserted text.

Duration Length of time for each act from this group of actions.

Conceptual Re-occurrence of important concepts in the course.

Opposition As in deleting and replacing with different structures.

Completion Actions that are complete in themselves such as chat comments with keyword descriptors.

Absence Absence of an expected part of an action.

It is important that the translator does not 'over translate' and remove too much information that may be domain-dependent, or which may vary over time as the students' conflicts develop and resolve. Otherwise the job of analysing the data, while shorter, will be weakened. Also some actions may be considered as part of more than one act. They are assigned to acts in the order listed above and thus while no conflict occurs, some information may be lost.

7 Intertac-II

The modular structure of the Intertac-I architecture allows the developers to plug in intelligent agent components. The design of the next stage of the software involved the development of rule-based agents that are initiated by each tool and the interface of Intertac-I. These agents were developed from experimental use of Intertac-I where students carried out their project work, with the translator functionality being used to summarise their activities.

The patterns that were extracted from the group interactions and learning, were used to develop an Implementation Pattern Language, which is discussed in later papers. Agents providing diagnosis and feedback can be developed for different learning domains from the pattern method developed.

The architecture was augmented to the following:

The next step in developing the software involves providing coordination of the agents to prevent conflict and enable combined feedback; and a more thorough analysis of the history of student activity

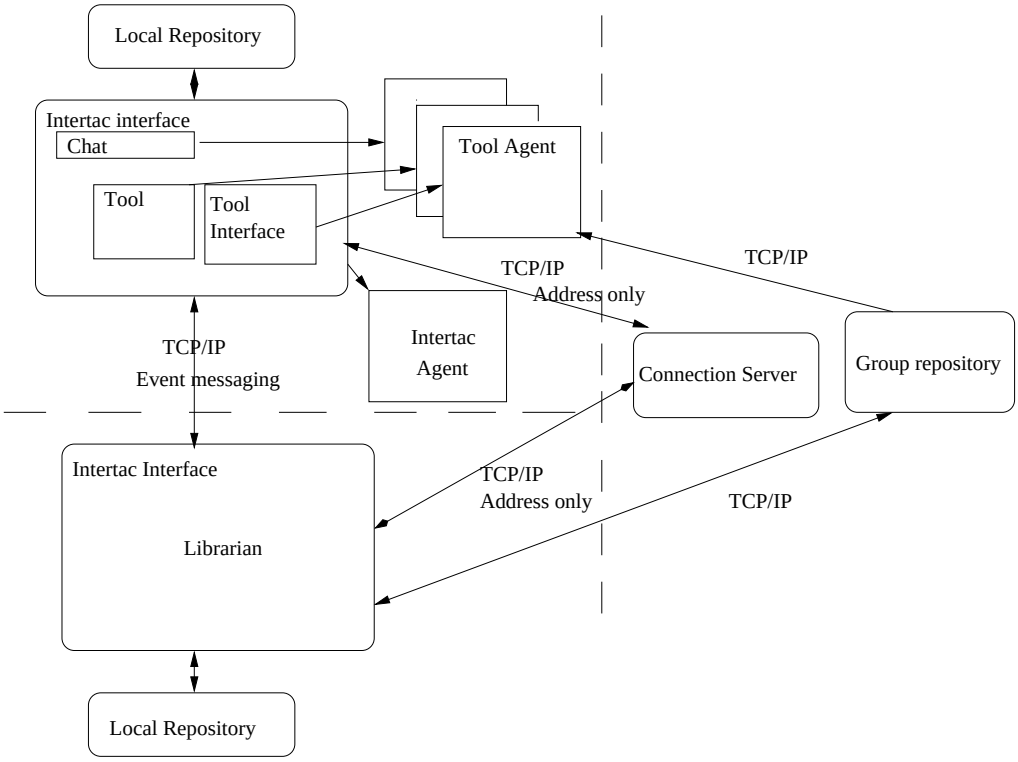

Figure 3: Architecture for Intertac-II System with no User Model

to provide the basis on which agents can analyse students' depth of learning approach. This work invovled the development of a User or Group Model.

8 Verification Testing

As in the development of any software for general use, it is important that it be used by other people than just the developers, who are not typical users. Thus much of the fieldwork described in the next chapter involved analysing how students used the software, and how this could be improved.

Once the computer supported co-operative work (CSCW) system is developed, there are different approaches that can be taken to verify the effectiveness of the software. A controlled experiment can be conducted to analyse the effectiveness of meetings held through the software interface, as compared to collocated groups. In fact some of this work had been done by Rein and Ellis [16], with the main findings being:

1. The option of group or individual messages enabled the quiet members to influence the direction of the meeting by posting their comments and questions while the general discussion was going on.
2. The BlackBoard system tended to increase group focus and attention on completing the task. The increased group focus resulted in some improved capture and organisation of ideas and more effective and efficient meetings

When Intertac was developed, it was envisaged that some form of controlled test could be conducted; however various factors mitigated against this. Firstly the assistance being developed was heavily inter-related between types of advice. Rather than select some aspect of the advice and switch other aspects off, and compare this across all types of advice, it was felt the system should be verified as a whole.

Secondly the tools developed were originally quite primitive, as the assistance was the focus of the work. In the meantime these have been improved, partially through the use of open-source software,[17] yet there is still resistance to the use of new applications amongst software engineers. It would be hard to filter out this resistance to achieve a valid picture of the assistance developed within these applications.

The verification of Intertac-II including its agent support, should consider the aspects of Constructive Learning Environments that were adopted as the goals of this work:

Scaffolding By developing User (or Group) Models that enable the tracking of students overt response to advice and the actions that follow any advice [**?**] some analysis of the scaffolding effect can be examined. While students will have responses to the immediate use of such agents, it is important that the patterns used by the agents to provide feedback be re-analysed to verify the effect of the feedback, if any, on the patterns observed.

Alternatives Another important aspect to verify is the search agents. This will involve running the agents on documents produced by students to verify that the Design Patterns extracted in searches are valid comparisons or alternatives.

Feedback A study should be made of the feedback that is received during the course of a workshop and how these relate to the resultant document and design produced by the group. This will be to verify if design problems are missed in the feedback or feedback is made that is not helpful.

Reflection During the workshops the students can be interviewed about their approach to learning software design, their approach to working in groups remotely and their conceptions of the key aspects of the course. These can be related back to the agents that are designed to deal with these learning patterns and verify that the agents have either identified or responded in some way to these approaches.

9 Conclusion

The project has made some headway in developing the groundwork, providing some data to assist developing models of students' learning depth based on their learning processes. However, the main functionality of the system, that of the scaffolding, has still to be implemented

[17] The ability to plug in any Java Frame based application is the major advantage of the design of Intertac.

based on data from students' learning actions. Then there remains the non-trivial task of analysing the educational acts as part of a particular learning model.

10 Glossary

Agent — a program which is running in the background and communicating with the system software. The agent uses the input from the keypad and mouse to receive information and enters responses into the windows on the screen (compare human agent).

Action — the conscious steps of an activity.

Activity — from activity theory, is a name for the collection of steps taken or content of the activity, plus the context and consciousness of the human agent.

Application — a stand-alone program, that may be added to groupware to be used in a single activity of the group process, such as a Chat application, an Editor application for text files, or a Graphical Editor. Such programs are based on stand-alone applications for single mode work (see also Tool).

Awareness — in groupware awareness is reserved as a word to describe any tool used to enable users to be aware of who else is linked to the same session or common, shared interface.

Chat Channel — a tool either in a groupware system, or standalone, that enables rapid transmission of text messages that are displayed as separate lines in the form of a sequence of contributions to the 'chat', which may be out of order in terms of providing side-by-side display of question/answer pairs.

Client — person who has requested the production or design of a piece of software. Not used in the client/server sense.

Collocated Group — group that is working face to face with access to visual cues, etc.

Completion — The end of an Activity.

Constructivism — theory of learning that states that the learning takes place inside a student, through their own construction of their knowledge, with or without outside influence.

Correctness — conforming to requirements, used as property of course or software.

CVS — Concurrent Versioning System.

Decision Making — User in various groupware to describe processes to support groups coming to a conclusion. It may just invovle voting, or in the case of this work, may include the various steps of proposal and counter-proposal.

Depth — the approach to learning taken by a learner towards a course or project, or the level of learning approach assumed in a learner when forming advice to the learner.

Framework Constituents — background concepts of a course that are important to the understanding what it means and what it takes to learn the course material, but are not thematised, or expressly explained, in the instruction (see Technical Constituents).

Grab — term used to describe the action by software to intercept input or output form a program to redirect to another format or device.

Human Agent — as distinct from a software agent, a person who communicates with other humans through the software. The human agent uses the mouse and keypad to enter information into the tools, and the screen to receive (see also Agent).

Intertac-I — Software developed to provide an unstructured flexible medium for distance group communication, file sharing and synchronous editing.

Intertac-II — Augmentatin of Intertac-I with agents to suport various aspcts of learning in the software engineering domain.

Level — can refer to the level or depth of learning in a student's approach to a course, or the level of a diagram within the tree structure of a Data Flow Diagram.

Local Files — refers to the file system on the computer the user is logged into (see also server)

Needs of the Client — The initial expression of the requirements for software by the people or organisation who will be buying it.

Outcome — An activity has an outcome that is the result of the activity. In developing software that is designed to support this activity, the outcome is also an aspect of the activity that the software may change.

Server — software running independantly of groupware tools, usually on a different computer, which handles communication isses to maintain synchronous views by all users (see also Synchronous).

Session — used in reference to groupware to describe the communication between users during the period in which they are linked by the network.

Synchronous — refers to the functionality of software that takes a continual stream of changes from many users' keypads and mouse inputs and provides a list of changes to each user's interface that maintain concurrency, or match the order in which the changes were received by the server. Changes may arrive faster than the server can re-transmit them, but it must retain the information from all back-logged changes.

System — a computer program that stands alone but involves many parts. For example the separate tools or applications of groupware together form a groupware system.

Technical Constituents — Technical aspects of the course that are assumed knowledge, or explained in the course (see Framework Constitutents).

Tool — an application that is not stand-alone. For example agents in most systems, and the applications in Intertac-I (see also Application).

Versioning — as files are edited and changed, the computer system can save a record of the changes form one version to the next. These changes can be in the form of commands to edit from one version to the next. If the user wishes to access previous changes to undo them, or repeat them on a different version of the file that lacks these changes, the computer can support this process automatically.

Workshop or Course — refers in this thesis to the software engineering design workshops for which Intertac-I was developed and in which it was trialed.

WYSWIS — Pronounced 'wizzywis'. What You See is What I See. Concurrent views of a window between multiply users.

References

[1] F.N. Akhras and J. Self, System Intelligence in Constructivist Learning. *International Journal of Artificial Intelligence in Education*, **11** 4, 2000, 344–376.

[2] J.R. Anderson, P. Pirolli and R. Farrell, Learning to program recursive functions, in M.T.H. Chi, R. Glaser and M.J. Farr, editors, *The nature of expertise*, Hillsdale, NJ: Lawrence Erlbaum.

[3] Booth, S. (1992). *Learning to Program: A phenomenographical perspective*. Gőteborg Studies in Educational Sciences,**89**, Acta Universitatis Gothoburgensis.

[4] Collins A. Brown J.S. and Newman S.E., Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics, in L.B. Resnick, editor, *Knowing, learning, and instruction: Essays in honour of Robert Glaser*, Hillsdale, NJ: Lawrence Erlbaum Associates, 453–494.

[5] P. Dillenbourg and J. Self, Designing human-computer collaborative learning, *AAI/AI-ED Technical Report No. 91*, in C. O'Malley, editor, *Computer-Supported Collaborative Learning*, 1994, Berlin, Springer-Verlag, 245–264.

[6] Fjeld M., Lauche K., Bichsel M., Voorhorst F., Krueger H. and Rauterberg M. (in press): Physical and virtual tools: activity theory applied to the design of groupware. B. A. Nardi and D. F. Redmiles, editors, A Special Issue of Computer Supported Collaborative Work (CSCW): Activity Theory and the Practice of Design.

[7] Grabinger S., Dunlap J. and Duffield J. (1997). Rich environments for active learning in action: problem-based learning. *Association for Learning Technology Journal*, **5**, 2.

[8] Habanero. Retrieved Janurary 10, 2001 from http://havefun.ncsa.uiuc.edu/habanero.

[9] Jonassen D.(1998). Designing Constructivist Learning Environments. In C.M. Reigluth, editor, *Instructional theories and models*, 2nd Ed. Mahweh, NJ, Lawrence.

[10] Jonassen D. and Rohrer-Murphy L (1999). Activity Theory as a framework for designing Constructivist learning environments, *Educational Technology, Research and Development*, textbf47, 1, 61–79.

[11] Matchmaker University of Duisburg. Retrieved August 10, 2000 from http://collide.informatik.uni-duisburg.de/Software/Docs/JavaMatchMaker.

[12] M.L. Markus, M. Lynne, and M. Keil, If we build it, they will come: Designing information systems people want to use, *Sloan Management Review*, **35**, 4, 1994, 11–25.

[13] M. Műlenbrock, F. Tewissen and U. Hoppe, A framework system for intelligent support in open distributed learning environments.

In B. du Boulay and R. Mizoguchi, editors, *Artificial intelligence in education: Knowledge and media in learning systems*, 191–198. 1997, Amsterdam, The Netherlands: IOS Press.

[14] D. Mwanza, Towards an Activity-Orientated Design Method fro HCI reserach and Practise, PhD Thesis, 2002, The Open University, UK. Retrieved November 8, 2002 http://kmi.open.ac.uk/people/mwanza/phd-thesis.

[15] Nardi, B. A. (1996) Some reflectionson the application of Activity Theory in Bonnie A. Nardi, editor, *Contexts and Consciousness: Activity Theory and Human-Computer Interaction*, MIT Press, Cambridge, Mass. 69–102.

[16] G.L. Rein and C.A. Ellis, The Nick Experiment reinterpreted: Implications for developers and evaluators of groupware, in *Office: Technology and People*, **5**, 1, 1989, 47–75.

[17] Soloway, E., Jackson S.L., Klein J., Quintana C., Reed J., Spitulnik J., Stratford S.J., Studer S., Jul S., Eng J. and Scala N., Learning Theory in Practise: Case Studies of Learner-Centred Design. Proceedings of CHI 96, Vancouver, BC Canada April 13–18, 1996, 189–196.

[18] L. Suchman, Making work visible, *Communications of the ACM*, **38**, 9, 1995, 56–64.

[19] D. Suthers, Analyzing learner discourse effecs of representational bias. *Workshop at AI-Ed '99 9th International Conference on Artificial Intelligence in Education — Analysing Educational Dialogue Interaction: Towards Models that Support Learning*, Le Mans, France 18th-19th July, 1999.

[20] Teamwave. Retrieved June 29, 2001 from http://teamwave.com.

[21] T. Winograd and F. Flores, *Understanding Computers and Cognition: A new foundation for design*, 1987,NJ: Ablex.