

Assisting Students by Analysing their Interaction and Learning

Cat KUTAY and Peter HO*
Computer Science and Engineering
University of New South Wales
Sydney, Australia
{ckutay, peterh}@cse.unsw.edu.au

Abstract. To provide support for students, an unstructured learning software system was developed. This system was extended by a series of rule-based agents which aims to scaffold the particular learning needs of the domain. The software was developed for a software engineering workshop in which students are asked to produce a detailed design of a system from a given set of requirements. This paper looks at the development of a Blackboard-style User Model incorporating historical data to support the initiation of the rule-based agents, plus the use of weighted decisions for co-ordination.

1 Introduction

Computers have been used to assist learning in many domains. This work is an attempt to develop a process of codifying student learning needs, in a group-based project course, into rules for agent support and learner modelling. Our subjects are students involved in developing software engineering specifications in groups. Without appropriate supervision it is common for groups to develop systems that show lack of understanding of the course requirements.

We initially developed a tool called Intertac-I [1] to facilitate storage, management and group communication between group members. The tool enabled concurrent planning, editing, drawing and chatting. This in itself proved to be insufficient as no feedback was provided. The goal then was to further enhance Intertac-I with agents which enable students to develop a desired depth of understanding and a range of skills in the area.¹

Experiments were conducted with students using Intertac-I. They are working in an open domain, on an ill-formed problem. In work by Constantino-Gonzalez et. al. [3] students develop public as well as private designs that can be compared to develop advice. In the present work students deal with separate aspects of the work so there is little access to their thoughts on contributions by others. Instead, the group interactions and their learning approach are analysed on the basis of repeated processes.

The results of the experiments were used to develop rule-based agents that are a feature of Intertac-II. This paper looks at the type of agents developed and the

* The authors would like to thank: David Abdelmassih, Manoj Chandra, Sherman Lo, Zhicong Leo Liang, Zi Qi Lu and Naveed Hussain for contributing ideas and their programming skills.

¹ See Booth [2] for similar research into learning needs for students learning programming.

User or Group Models which are to be implemented to support these agents. The Models are to interpret both the historical development of the depth of learning and guide students in the processes they are to develop.

2 Implementation Patterns

The interaction processes and learning approaches that were studied are described in a pattern language² that is implemented in Intertac-II by agents. It must be noted that these are different to the interface patterns developed in HCI (such as [4]), as we are looking at the Human-Computer-Human (or HCH) interactions. Moreover, these patterns are different to the interaction patterns studied in other CSCW systems (such as [5]) in that the next step of implementation is included in the pattern structure.

The difficulty at present is that these patterns are manually filtered out and then implemented as agents, rather than by automated analysis of patterns from the interaction. This is the limitation of any rule-based system. However, the agents have been designed around the concept of an interface program. The interface program is designed to enable instructors to code the rules for future agents in possibly other domains.

The patterns provide a link between the desired learning as developed by the instructor and the agents. Agents are designed to implement support that either encourages the development of the procedural knowledge that a particular learning approach requires or provides the conceptual knowledge that may be required by a user to enable them to reach a deeper level.

3 Agent Development from Patterns

The patterns developed for analysing learning content of the students' online files provide learning objectives in a format that can be easily translated into agent support. The pattern structure developed is described in other work by the authors [6].

The agents developed in Intertac-II from these patterns are designed to implement constructivist principles of learning as enumerated by Savery and Duffy [7]³ and the principal steps of problem solving as described by Greeno [8].⁴

As with any multi-agent development, it is desirable to analyse the connections between agents. This connection is made through the pattern language and is encoded in agents to enable two areas of co-ordination. These are:

User Model of Level Agents that need to determine the level or learning approach or process development in the group, require historical data to support the assessment of this process; and

Co-ordinating Multi-Implemented Agents Only one type of agent should be providing feedback at one time. Agents of the same role and type may be combined in feedback format but where two patterns are not complementary, decisions must be made on which to implement.

² Concept of patterns for describing common features in systems was developed by Alexander [11]

³ Constructivist environments include: Scaffolding, Alternatives, Feedback and Reflection.

⁴ Problem solving steps are: Arrangement, Transformation, Deduction and Induction.

This paper looks particularly at the major agent types that require this support. Firstly, we will examine agents which support students developing procedural knowledge about their processes extended over time, and possibly over different sessions using the software, such as the Concept Agent and the Decision Agent. Secondly, we will review the Search Agents which are used to match the a section of the design of one group to that of historical documents to support students developing conceptual knowledge of the domain.

4 Generic Agent Design

The Intertac-II agents were developed using a rule language and structure that enables the coding or rules from many domains. The patterns provide an outline for translating learning needs in a course into an agent process. The types of agents developed to achieve these implementations are tabulated below:

Name	Content	Individual/ Group	Model Support
Intertac:			
Concept Extensions	Extend use of concepts	Group	Prior actions of agent
Linked to:			
Chat Feedback	Analyse discourse interactions	Group	Concept use in other tools
Planning:			
Planning Schedule	Follow Plan Template	Group	Stage reached
Topic Course Planning	Topics for each stage	Group	Stage reached
Sched.			
Editor:			
Explain Change	Change in file compared to last version	Group	Change patterns which gained response before
Explain Change Requirements	Change in User requirements	User	Change pattern which user responded to before
Justify Change	Late change in design	Group	Previous pattern of design change
Re-Design	Users requested to contribute alternatives	Group	Previous response to agent
Document Rule Checker	Checks Rules relating to document	User	Rules not ignored by user before
Diagram Editor:			
Diagram Rule Checker	Check rules relating to diagram	User	Rules not ignored by user before
Require User Model:			
Search	Look for patterns		Which search to make
Decision	Follow decision process		Stage of decision reached
Learning	Follow Learning process		Level of learning reached

4.1 Learning Agent

The basic learning agents are those designed at providing increasingly complex examples of the application of a concept. For an agent to do this, it is important that the system be able to track the use and development of domain concepts in the learning. A summary lexicon can be provided for a domain to use to search students contributions to Chat. Students are encouraged to focus their conversation by grounding it in the course.

The students can also receive support for how much and what type of conversation has been linked to each lexical word or phrase. For instance has there been much explanation, disagreement, argument, or decisions? This is done through

agents that support user interaction, which are not dealt with in this paper due to space restrictions.

The combined agent analysis can then generate their response actions relative to the expected level of engagement with that concept in the course, at each stage of the workshop, through the a course plan which also can be provided with Intertac for the learning domain. For instance, when a topic is introduced into seminars the software would expect more engagement.

4.2 Decision Agents

It was found that teaching the decision making concept was not sufficient, and in fact the actual skills needed to be practised for students to be successful. This may have been exacerbated by the group work being conducted online, which made the handling of decisions more difficult.

The steps of the group decision making process as described by [9] were observable from the users files as consisting of the following actions:

Exchange of information Either chat contributions with similar words, or files open with similar content or structure.

Aggregation of these exchanges One person draws or writes ideas. Often it is in shortened form, and so loses many of the ideas in the presentation.

Setting constraining conditions Usually another person will provide the constraints on the design or idea as presented. Setting enabling conditions The drawer/editor tends to be the positive voice, since they are already controlling the presentation.

Results Sometimes another person will take over the drawing and the process will start again. Other times, the students reported that they just let the person work with that part of the design. The level of discussion is very dependent on motivation and whether the students care about the details of the design.

The analysis of decision making processes is more complex than the other patterns which involved consecutive stages of action which respond to the same conditional. In this case each stage is analysed by different conditionals. In fact Decision Patterns are like Learning Patterns in that they require an analysis of the depth achieved or the progress made by the group to advise the next step. These agents are supported by the User Model which is used to maintain a history of the group process. This is explained in Section 6.

The Decision Agent is user initiated. The group or a member of the group must realise that they need guidance with their decision. This may be initiated by a positive response to another agent suggesting they come to a decision with agent support. Or it may be selected from the menu. Initially such an agent can be generated in the form of a wizard, which describes each step in the process, with a button for the user to select when they have completed a stage.

Each time a stage is selected, the agent can initiate analysis of the group interactions that may relate to that stage. For instance in the constraining stage, the Decision Agent can generate chat agents to analyse the interactions in the Chat window for conflict.

The agents generated at each stage can provide detailed analysis of the interactions at each stage. If the group repeats such processes a few times, it is hypothesised that the data so collected can be used as a comparative model to determine the stage of the group in their decision. This is a form of learning by the

system that takes the User Model data during each stage and presents this as the data for analysing the reoccurrence of that stage. The User Model has been designed with such applications in mind.

4.3 Search Agents

Search Patterns are used to examine files from previous years for specific features to select examples of alternative designs for users. The research extracted the Search Patterns which could be used. This diagrammatic analysis relies on selecting the basic aspects of each drawing primitive and looking for patterns of similar designs.

The Design Pattern lists similar features of designs. This can be used both to check for similar design aspects in previous years and also for differences between the groups older version of a design and their recent version.

Visual issues such as joining data too close to each other was a problem. Alternative better spaced designs can be displayed.

Keyword searches for keywords missing from the processes and data flows can be done. Unfortunately the keywords can change between projects in each workshop, so it will be hard to analyse across years, except when dealing with common system processes, such as login processes or the designing of time into the system.

Design Steps Some steps in the process of diagrammatic designs are handled badly by students, such as combining sections of a highly detailed level and then moving the detail to a lower level of the design. The two diagrams in this process, before and after, can be displayed from the context of another project, and hence abstracted from the details of the particular processes of the students design.

Similarly, documents from previous years can be searched for changes in design pattern. The patterns observed between distinct projects which would be worth noting for search categories are:

Length of section Where a users sections appear too short on some aspect, display a longer example.

Use of sections If important sections such as Non-functional Requirements are missed, then the students could be shown an example for them to see what this section should cover.

Expert Sections Where sections are generally handled badly in the course, such as sections on integration of aspects of the design document, then an example of a skilled report could be displayed.

This agent does searches generated at the request of other agents to match the design pattern of a section of groups design to historical documents, using a unification algorithm between graphical representations of the two designs. The features that are searched to find similarities in patterns are in the table below. The process is to find objects with feature that are similar to attribute one, and are alternatives in attribute two.

Tool Searched	Object Found	Attribute 1 (search similar)	Attribute 2 (search dissimilar)
Document	Section	Name	Length
Document	View	Keyword	(Search all)
Diagram	Process	Name	Input data and Output data
Diagram	Process	Name	(Search all)

Diagram	Data	Name	Splt data vs. unsplit
Diagram	Process	Name	Number of input or output
Diagram	Process	(Search all)	Spacing of input or output

User feedback on these agents can be recorded. This can be either requesting users to respond whether the comparative or alternative design was useful, or analysing changes that follow the suggestions from Search agents. This can be used to change the weight on each search pattern.

However it is clear that some searches will be more useful towards the start of the design (such as cluttered designs) and others later in the design (such as number of processes per level). Again the information as to the stage the group has reached will come from the Group User Model, either in terms of their progress through the course plan, or in terms of the design complexity or how near completion it is.

5 Agent implementation

The modular structure of the architecture allows the developers to plug in extensions to the existing applications and intelligent agents components. User modelling agents provide a framework for analysing students actions. Where the group is analysed, the agent can select whether they treat the group as the addition of the individuals contributions, or the most significant individual according to the pattern of interest to that agent. Agents use this data to initiate actions deciding on the level of feedback needed, or whether an agent that has satisfied its conditional, from analysing students actions, is appropriate for initiation at this stage. When more than one agent is initiated, they will be selected on the basis of their present weight.

Agents in Collaborative Learning Systems are generally Knowledge-based agents with some rule learning methods. The agents being developed for Intertac are rule-based, derived from the pattern language discussed. Some of these rules must be learnt from analysing many examples, such as diagrammatic analysis, however at present these are done manually. Agents then use the knowledge to analyse data on an individual as part of the group, or the group as a whole.

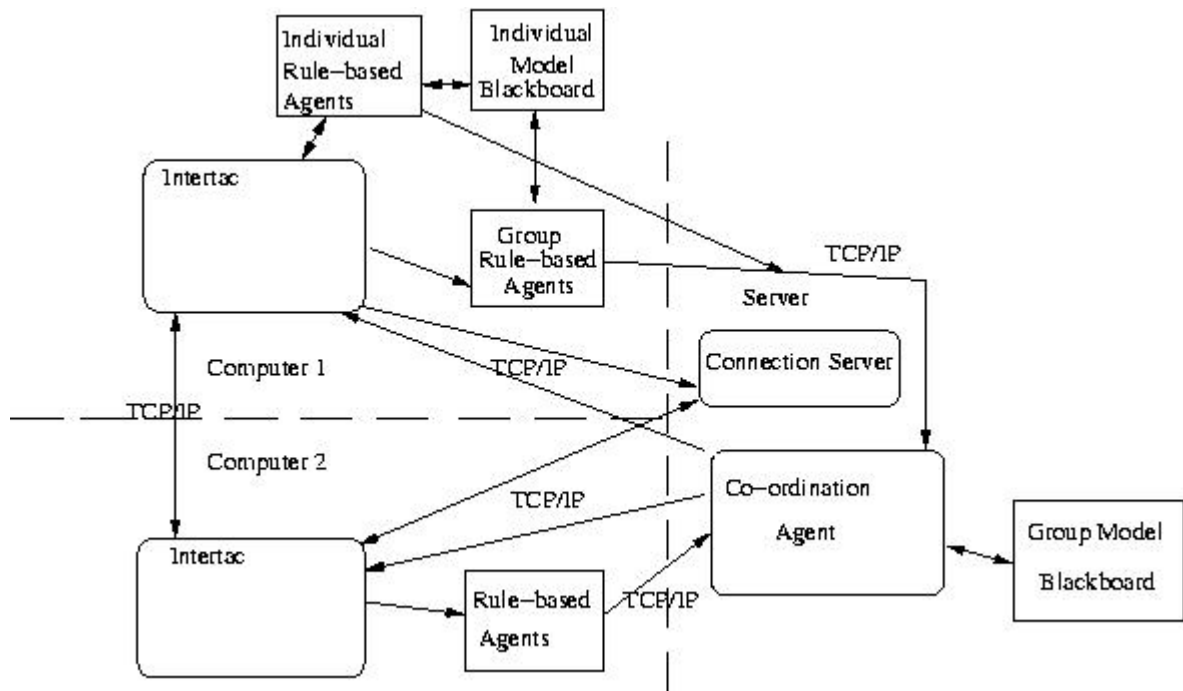


Figure 1: Architecture of Intertac-II including Co-ordinator Agent and User Models

The agents are run by the client, rather than the server, as this gives access to the users data files and means that the agents function even when the user is offline. How the agents interact with each other varies according to the role of the agent.

The feedback from agents can be varied. For example a simplified pattern of interaction can be analysed from chat tokens or topics by agents, then appropriate comments are entered in Chat or a common Notepad window. It is also feasible to automate this type of feedback for diagrams or documents, with agents based on analysis of the graphical patterns.

Some formal languages have been developed to express these rules, such as those by Akhras and Self [10] and Muhlenbrook et. al. [12]. In this work an approach based on an agent implementation pattern language have been used. These agents have been written for Intertac-II using rule files written in a Perl-style language saved in XML format.

5.1 Agent Architecture

The agents are linked to each tool, or to the Intertac-II main window. Where they analyse a specific tool, they are initiated by that tool. When an agent is designed to analyse a combination of tools, they are initiated by Intertac-II itself. However, many agents may analyse a single tool, but still require information from others tools. This information is to be maintained in a blackboard architecture for providing data to the multi-agent system [13] and [14]. The rule-based agents metaphorically gather around the blackboard, look at information written on it, analyse them and add their conclusions. In this case the conclusions are added via the Co-ordination agent.

Agents support both individual and group work. User Models have been designed for both single users and groups. They differ in terms of the data they record, as the features of each type of user vary. However they have much in common. The Group Model is to be maintained by a single Co-ordination agent. The Individual User models are maintained locally for local agent use in single mode.

Where relevant the Group Agents access the local Individual Model for data relating to their activity.

The resulting communication between agents can be summarised as:

1. The Co-ordination Agent maintains blackboard data information collected from historical analysis of learning process for Group User Models.
2. Group Agents checks blackboard for historical record of Group Process, or Individual User Blackboard.
3. Group Agents verify enactment with Co-ordination agent.
4. The Co-ordination Agent stores data on feedback from Group Agents, including data relating to any individual model that is used by Group Agents.
5. Group Agents vary their weighting according to usage.
6. The Co-ordination Agent stores all data at end of session and reads this again at the start of a new session.

6 User Model

As patterns of interaction are observed and linked to the students learning approach, or common errors in design are extracted, then the aim is to use the learner or group model to present information at a greater depth of learning to stimulate reflection or to extend understanding.

The User Model designed for Intertac-II is a history of:

Concept list A list of concepts provided by course for analysing students learning.

Concept complexity Information on how often a concept has been used, in which tool. This is used by agents to suggest increasing complex applications of that concept to the group.

Concept experience A record of how much help has been received on a concept to data.

Decision stage The stage reached as selected by user.⁵

Decision information A record of number of chat contributions during this stage by each user.

Decision aggregation A record of time taken to do re-design during this stage.

Decision constraints A record of length of each users contribution to chat during this stage, that contains links to keywords used in the design or documents.

Decision enabling Number of edits during decision constraints stage by the re-designer. Also a record of change of the editor at this stage.

Decision Result If decision is reached, save data, if not, start again.

Design complexity Information relating to number of levels in the diagram, number of processes in the higher levels, etc., that give an idea of how near completion the design may be.

Design completion Related to the complexity, but contains information on how closely linked the requirements and DFD are, hence how close to completion.

Design experience A record of how much feedback on designs received to date.

Course Plan Stage reached in course plan, including concepts covered in workshops or lectures.⁶

⁵ This may be extended to agent selection of the decision stage.

⁶ Selected by the user or group updating the time-line.

Agent stage Upon completion of each session, the weighting achieved by each agent is stored.

Agent links Agent pairs that have been initiated together and the selected agent each time.

File versions Links to the latest version of each file on the main server.⁷

Windows open The windows that the group has open at the end of one session can be saved for the next session.

The Individual Models include all the modelling data so that the user can work in single user mode. The analysis is performed on the individuals initially, but where required the group model can be produced as the aggregate of the individuals models (such as decision information); as the maximum of the individuals (such as concept complexity); or as any one of the individuals (design complexity is analysed from all files open to all the group).

These aspects were chosen to provide scaffolding for which it is necessary to know what stage the group has reached. The context is the stage in the course plan, and the level of support received so far. The consciousness of the group changes depending on its focus. This analysis may be complex; for instance users may not be contributing to chat, as they are busy drawing. Then finally any information that can be collected on user actions can be analysed according to the agent rules for the content of the learning.

The agent rules implemented at present in Intertac-II can analyse basic processes for determining the depth of learning approach exhibited at that time in the group interaction. However the initiation of agents and the depth of response the agents provide can use different semantics based on the attributes from the student model. This has been shown to be effective [15] in a closed domain where students spend more time working collaboratively.

7 Maintenance of the User Models

The User Models are maintained by the Co-ordinating agent which receives information from the Intertac and tool agents. Also information from the user can be received. Agent advice can be generated in the form of a window message that students can cancel or accept. This can provide some feedback on the perceived usefulness of the agent input, and provides an indirect form of user editing of the User Model. A complete interface to the model has not been developed (compare work by Kay [16]), as the data as saved is not yet interpretable in a usable form by a single view or format.

Another approach is to use the agents to monitor the user response. Intertac-II is designed to monitor user activity and agents could be expanded to follow the activity after they carry out an action, to verify if there is any positive response to their advice, that is, verify if the subsequent actions fit the desired process of learning or interaction pattern.

8 Conclusion

⁷ These links are used to analyse difference with new version opened during session. The Librarian saves updated version when group accepts new version.

Constructivist learning environments encourage flexibility and discourage attempts to prescribe interactions between students. However by a judicious choice of formats, students can be questioned about their understanding, or have timely alternatives suggested from interventions by simple intelligent agents. This has been developed in software and is being extended to include User Modelling. The modelling provides co-ordination of possibly conflicting advice and some aid in assessing the level of learning already achieved by the group.

References

1. Kutay C., Intertac Software Architecture, *Technical Report No. 0318*, Computer Science and Engineering, University of New South Wales, <http://www.cse.unsw.edu.au/~ckutay/Intertac/IntertacArchitecutre.pdf>
2. Booth S., Learning to Program: *A phenomenographical perspective*. Goteborg Studies in Educational Sciences,89, Acta Universitatis Gothoburgensis, 1992.
3. Constantino-Gonzalez M. and Suthers D.D., Coaching collaboration by comparing solutions and tracking participation, in P. Dillenbourg, A. Eurlings, K Hakkarainen, editors, *European Perspectives on Computer- Supported Collaborative Learning*, Proceedings of the First European Conference on Computer- Supported Collaborative Learning, Universiteit Maastricht, Maastricht, the Netherlands, March 2224, 2001, 173-180.
4. Schummer T., Retrived on January 10, 2002 from: <http://www.groupware-patterns.org>.
5. McManus M. M., and Aiken R. M, Petri nets: using a formal model to describe synchronous rules in concurrent processes, *Technical Report TUCIS-TR-1999-001*, Temple University, 1999.
6. Kutay C. and Ho P., Using intelligent agents for the analysis of students interaction and learning, submitted to *AIED03 Workshop on Technologies for Electronic Documents for Supporting Learning*, 2003.
7. Savery J.R. and Duffy T.M., Problem based learning: an instructional model and its constructivist framework, *Educational Technology* **35** 5, 1995, 3-17.
8. Greeno J., Trends in the Theory of Knowledge for Problem Solving in D.T. Tuma and F. Reif, editors, *Problem Solving and Education: Issues in Teaching and Research*, 1987, John Wiley and Sons, New Jersey, 1980.
9. McLeod P.L., New Communication Technologies for Group Decision Making. R. Hirokawa and S. Poole, editors, *Communication and Group Decision-Making*, 426-461.Sage Publication, Beverly Hills. 1986.
10. Akhras F.N. and Self J.A., System Intelligence in Constructivist Learning in *IJAIED*, **11**, 2000, 344-376.
11. Alexander C., Ishikawa S., and Silverstein M., *A Pattern Language*, Oxford University Press, Oxford, 1977.
12. Muhlenbrock M., Tewissen F., Hoppe H.U., A Framework System for Intelligent Support in Open Distributed Environments, in *International Journal of Artificial Intelligence in Education*, **9**, 1998, 256-274.
13. Lander S., Staley S. and Corkill D., Designing Integrated Engineering Environments: Blackboard-based Integration of design and analysis tools, *Concurrent Engineering: Research and Applications*, Special Issue on The Application of Multi-Agent Systems to Concurrent Engineering, 1996.
14. Soler J., Julian V., Rebollo M., Carrascosa C. and Botti V., Towards a real-time multi-agent system architecture. *Proceedings of the Workshop on Challenges in Open Agent Systems*, at Autonomous Agents and Multi-Agent Systems, AAMAS02, Bologna, Italy, 2002.
15. Constantino-Gonzalez M., Suthers D.D. and Santos J., CoachingWeb-based Collaborative Learning based on Problem Solution Differences and Participation, *International Journal of Artificial Intelligence in Education*, **13**, 2002.
16. Kay J., Halin Z., Ottomann T. and Razak Z., Learner Know Thyself: Student Models To Give Learner Control And Responsibility, *International Conference on Computers in Education (ICCE97)*, Malaysia, AACE, 1997, 17-24.