Implementation Patterns for Supporting Learning and Group Interactions.

Submitted for the completion of the degree of PhD in Computer Science and Engineering at the University of New South Wales

Cat Kutay

August 1, 2005

Abstract

This thesis covers research on group learning by using a computer as the medium. The computer software provides the basic blending of the students' contributions augmented by the effects generated for the specific learning domain by a system of agents to guide the process of the students' learning.

The research is based on the approach that the computer as a medium is not an end point of the interaction. The development of agents in based on Human-Computer-Human interaction or HCH. HCH is about removing the idea that the role of the computer is that of an intelligent agent and reducing its role to that of a mixer, with the ability to insert adaptive electronic (software) components that add extra effects and depth to the product of the human-human interactions.

For the computer to achieve this support, it must be able to analyse the input from the individuals and the group as a whole. Experiments have been conducted on groups working face to face, and then on groups using software developed for the research. Patterns of interaction and learning have been extracted from the logs and files of these group sessions. Also a pattern language has been developed by which to describe these patterns, so that the agent support needed to analyse and respond appropriately to each pattern can be developed.

The research has led to the derivation of a structure that encompasses all the types of support required, and provides the format for implementing each type of support. The main difficulty in this work is the limited ability of computers to analyse human thoughts through their actions. However progress is made in analysing the level of approach by students to a range of learning concepts.

The research identified the separate patterns that contribute to learning agents development and form a language of learning processes, and the agents derived from these patterns could in future be linked into a multi-agent system to support learning.

Acknowledgements

The author would like to thank her supervisors Geoff Whale and Peter Ho and Albert Nymeyer for providing much support both in the research and the experimentation. Also she would like to thank them for being very pleasant people to work with.

The author would like to thank the academics at The University of New South Wales and Sydney University with whom she worked as a tutor and lecturer. They enabled her to discuss education theory in practice and implement some of the ideas of this thesis in her teaching of campus based students.

The author would like to thank the other researchers who responded so fully to her email requests for clarification of theory and approach. It enabled her to feel less isolated from the distance education research field. Also she thanks the academics in Computer Science and Engineering who provided instruction in programming, formal methods and the process of software design that formed the basis of this thesis.

Also the author would like to thank the support staff at the university who coped with her glaring ignorance of computing matters and terminology through her first years on campus, and also the School of Computing Science for providing the computing resources that enabled her to set up and maintain her own systems that supported the development of Intertac.

The author wishes to thanks all the students who were involved in the experiments, especially those who worked on the initial version of Intertac and had to deal with a server that was poorly integrated and not designed for such an open interface as Intertac provided. Also she would like to thank the Faculty of Engineering for supporting and funding these projects.

In particular the author 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 to the development of Intertac III and the agent support for Intertac-II which was developed at the completion of the work reported here.

The author acknowledges the help of Andrew Westcombe in editing the initial draft of the thesis and providing invaluable coaching in expressing herself in written English.

Finally the author sincerely thanks Dianne Wiley for all the time she put into a thesis that was outside her domain, but benefitted greatly from her strictures. The author only trusts this thesis is a credit to her efforts.

Contents

List of Figures

List of Tables

Chapter 1

Patterns of Learning Interactions

1.1 Introduction

This research is the study of the way that quite different tunes from different players can blend to make a sound that is harmonious and fuller than the original pieces. In music these sounds are achieved through a mixer, an electrical panel that takes different musical sounds as inputs and blends or mixes them at different volumes, tones and delays.

When we listen to live music such as jazz, we listen to the different instruments, played by talented musicians with their particular preferred modes of playing, particular interpretations of the score, and we marvel at how they blend together. That is, how the individual musician's instruments, talents and sections of music played manage to fit together to make a whole that is greater than the sum of its parts. Yet to play in public, a musician requires amplification, and the mixer can ruin their[1]

[1]The plural forms *they* and *their* is used in this thesis rather than *he or she* singular even when the latter may seem more appropriate. This is supported by the recommendations of the guide by Miller and Swift (1980) as cited by Kay [196]. The singular form of *they* and *their* may also be used either in case of unspecified gender or indefiniteness as to the person, as described at http://www.crossmyt.com/hc/linghebr/austheir.html in reference to the use of *their* in Jane Austin's work.

music. They still rely on the mixer to produce volume balance, tone and *body* and also control the foldback or feedback and effects such as echo.

When we work in groups to produce something other than music, we still must blend and interconnect. Most people lack three of the ingredients common to groups of musicians. The appreciation of the type of instrument, or part we play, the ability to hear others and blend, and the desire to appreciate the whole over its parts.

In this work we are working on drawing together a group of minds for learning by using a computer as the medium which interprets and augments the activity of the **students**.[2] These effects are generated for the specific learning domain by a system of **agents** that are designed to guide the process of that learning.

This research project is based on the approach that the computer is a medium rather than an end point of the interaction. While HCI issues are considered in developing the initial design of the basic software, the development of agents is based on Human-Computer-Human interaction or HCH. HCH is about designing the role of the computer not as the intelligent agent but as that of a mixer, with the ability to insert adaptive electronic components (or software agents) that add extra effects and **depth of outcome** to the product of the human-human interactions.

HCH is about ensuring the software is used to support members in adapting roles in meetings (by providing suitable interfaces and tools), to enable the group members to be aware of each other and their contributions (by notifying the group of changes and the interaction patterns) and finally to try and focus the group on the whole learning domain that they are dealing with (by introducing conceptual questions and ideas), not just their individual tasks.

[2]This thesis uses many terms from computer software and learning design. To provide some help with the language of each domain some terms are written in bold type and defined in the Glossary Section 11. We hope this will assist the reader.

1.2 Thesis Goals

The thesis presented in this work is that learning in the domain studied can be formalised and presented in terms of learning activities. Each type of learning activity has a pattern or structure and can be inter-related in the form of a Pattern Language (see Alexander [13]).[3] The development of Pattern Languages was initiated by Alexander [13] in the domain of architecture design. Later work has been conducted on patterns for Software Design by Gamma et al. [129], HCI Design [328], Requirements Engineering by Maiden et al. [234] and Analysis of Qualitative Interactions by Martin et al. [241]. The present Implementation Patterns extend the concept of Interaction Patterns to look at their implementation in agents that analyse and provide feedback on the basis of these interactions.

The work involved five steps based on the conversational learning approach of Laurillard [217] and to be implemented through software for Computer Supported Learning (CSL):

Goals Identification of the conceptions of group work and software engineering held by students designing software specification documents and the classification of **depth of approach** in their understanding of the subject involved in each conception. This will provide the basis for the learning goals in the software design.

Design Design the basic software structure for the learning environment that will enable the learning process. This software is called *Intertac-I.*

Actions Examination of learning strategies used by students working in groups to

[3]A pattern structure is developed for learning activities and these are related to each other in terms of relative **depth of activity**, constituent concepts of over-riding patterns, etc. This provides a language to analyse the learning activities and decide on advice to be given.

identify how the various depths of approach are manifest as **depth of activities**. Data is collected from students working in groups on the software system, and analysed for patterns in space, time and interaction between students. The data collected is used to develop a method for translating student text and graphical interactions into educational actions (cf. Ritter & Koedinger [309]). This provides a means of analysing the particular concept or skill that students manifest at any time in a flexible environment.

Feedback Develop a pattern format[4] for the interpreting the interactions that represent the concepts and skills of the learning domain studied and provide suitable feedback. A language structure[5] is developed for the patterns by relating them to the students' depth of activity.

Future Work Implement the patterns in the software, in the form of agents for Intertac-III. There are two types of agents, those that analyse students' interactions and those that analyse the students' learning. The final phase, is to provide inter-agent communication and an agent structure through **Learner Modelling**. This will enable support to be given that is appropriate to the depth of activity attained by the group and the milestone reached in the design document, hence remove the **scaffolding** as the students' comprehension of the domain develops.

1.3 Scope

Given a thesis that deals with such a broad range of issues, it is important to make clear at the outset the scope of this research. This present research presupposes that

[4]To develop patterns a consistent format is devised in which every pattern can be described fully.

[5]patterns can then be connected to each other with a semantic structure to form a language of patterns.

the courses where students study in flexible mode, using the system developed as part of this work, will be a group based project or *problem based* course. However at present there are few University courses developed within this framework and even the courses used in this research lack some of the factors that are prominent in the creation of such courses.

This work is part of an ambitious approach to support problem based learning. However, the ideal conditions for learning are not always present in the course structure mainly due to lack of resources, although the software developed for this research attempts to implement the tools required for such a course.

Also the research looks at groups that are functioning and are learning, although this may be at different levels or depths of activity. Since the work looks at the interactions, either group communication or learning interactions, which can be analysed by software agents, issues such as group psychology, dysfunctional groups, adult learning, course design and access to information are not within the scope of this thesis. There is some discussion of learning in groups, and learning group processes, and this is sufficient for the aims of this work.

This is a study of the groups who volunteered for the experiments, and limited to the narrow domain of them writing a specification report and the learning involved in that process. The thesis is an analysis of their learning and interaction although informed by the theory of how this may be affected by various differences in their approach to learning.

In designing feedback based on students' activities on the computer, it would be useful to have a thorough understanding of the possible causes of the depth of their learning activities. However if the different causes cannot be isolated on-line this does not assist in developing feedback.

Finally the work is qualitative, and designed to build theories and build pattern structures. The validity of the patterns is not asserted, but left to future work. The process of the derivation of the patterns is explained to enable others to repeat this procedure and verify if more patterns or a different structure is evident. However this difference would not we believe affect the pattern structure that is derived, and which forms the final outcome of the research.

1.4 Contributions

The contributions of this thesis are as follows:

1. Concepts of and approaches to learning software design. A Phenomenographical study is made of this area.
2. Computer Supported Collaborative Learning(CSCL) software is designed and a process for development of software for a Project Based Learning course is described.
3. A design for learning agents is developed around a pattern structure to enable the making of decisions based on the group or individual student's learning activity level.

As part of this research, software is developed for the use of students of the Software Engineering Workshops at the University of New South Wales. In particular the agent support is designed to augment the learning environment due to the following requirements for learning that are not being met in the existing course, and could not be met by providing web based resources:

1. Immediate feedback on performance in the area of rules and procedures of the domain
2. Providing feedback that is context sensitive both in terms of timing and the specific approach being taken by a group of students to software design

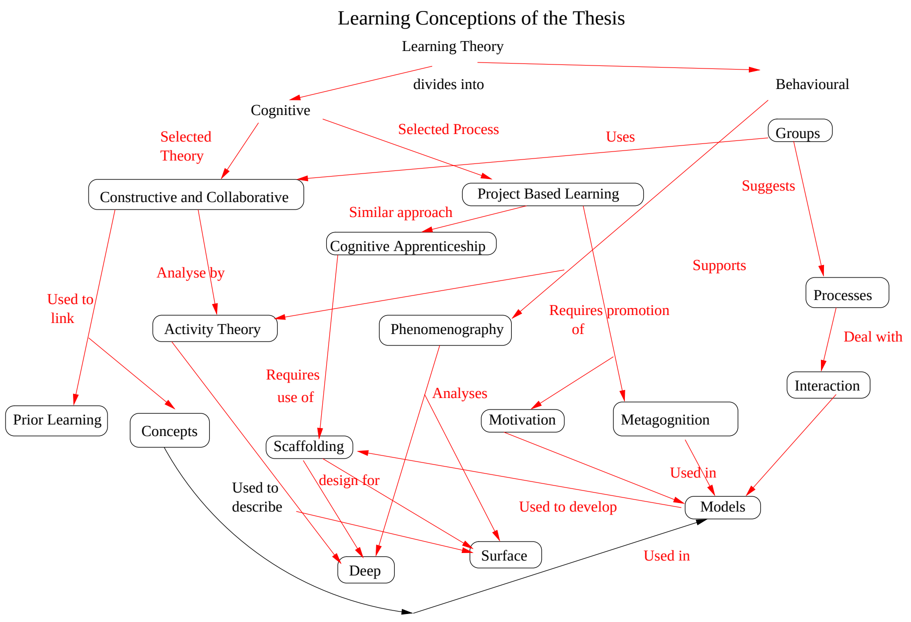


Figure 1.1: Concept Diagram linking the Learning Theories used in this Thesis.

3. Tracking and analysing the patterns of interaction during the learning.

1.5 Structure of the research

Concept diagrams of topics covered in the thesis are shown in Figure 1.1 and Figure 1.2. The diagrams show some of the links between theories, without attempting to provide a complete analysis of how theories relate. The topics on dark outline are the concepts covered in this thesis, and the rest are related ones that are not dealt with due to space constraints.

In Figure 1.2 the concepts not covered in the thesis are also included, except with dotted outline. Those topics that are introduced in Chapter 5 on the software environment are outlined with a dashed and dotted rectangle.

The important factor in this research revolves around the provision of a learning context for students' projects that affords the sort of activities that will change

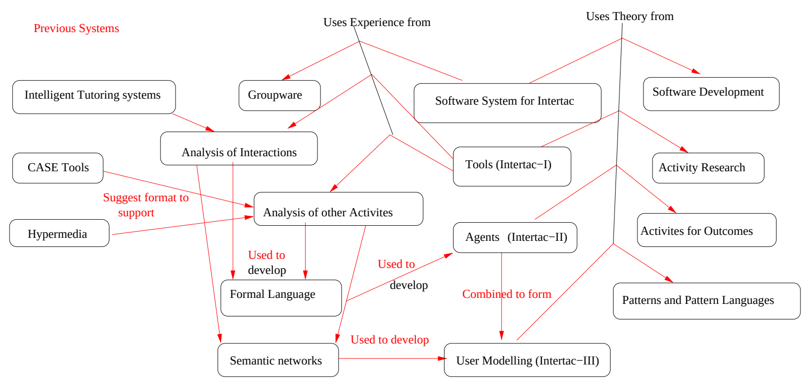


Figure 1.2: Concept Map of Software Concepts used in this Thesis.

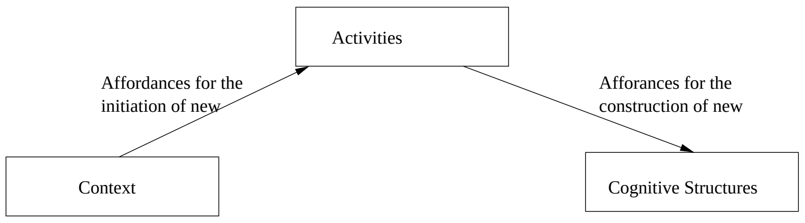

Activities
Affordances for the initiation of new
Afforances for the construction of new
Context
Cognitive Structures

Figure 1.3: Model of Constructivist Student Analysis.

and provide contradictions in their present cognitive structures. The distinguishing properties of Constructivist learning (see Akhras and Self [10]) are firstly the activity, that is learning by doing, and secondly the experience, that is doing more through learning. This activity links the context of the learning with the cognitive structures or consciousness developed by the students (see Figure 1.3).

In this work we trace the development of the software in the following stages showing the contribution to research:

1. Develop a picture of both the content of the course being studied by the students and their approach to learning in that course through Phenomenographical research.

2. Interface design using a Human-Computer-Human or HCH approach to enable the basic group interaction with editing and chat tools. This is developed

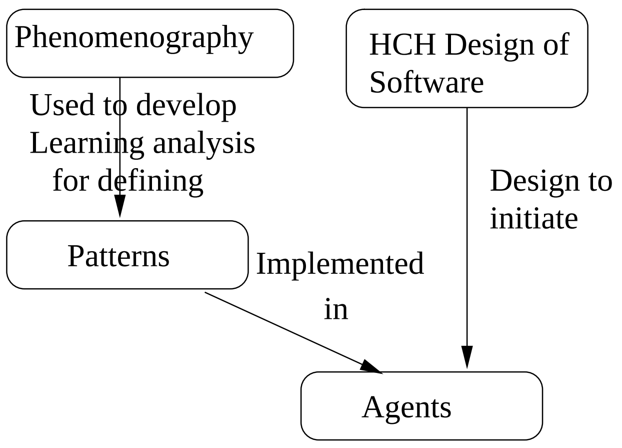


Figure 1.4: Contributions and how they relate

through research using Activity Theory.

3. Enhance the interface through Activity Theory experiment with students using the basic interface, to produce Intertac-II.
4. Analyse patterns of interaction and learning used on the basic tool by various groups of students to develop a framework to include Agents support. This work is done using Action Research.
5. Design the patterns for agent implementation in Intertac-III to provide the learning support for the learning approach taken by groups of students within the domain.
6. Consider the future stages such as implementation of the agents, which are needed to provide more flexible support and feedback.

This thesis is divided into the following chapters:

Chapter 2 Foundations of Learning Theory and Computer Supported Learning. Covers the areas of learning theory and development of learning software tools which provide the background to the research of this thesis.

Chapter 3 Methodology of the Research.

The methodology used in this research and the plan of how the research developed.

Chapter 4 Phenomenography of learning.

Extracts the goals for learning at different depths. Results of the Phenomenographical study of students' learning which forms the initial research of this thesis.

Chapter 5 Description of Intertac.

Describes the design of a learning tool called Intertac and explains the decisions on what data is collected for analysing the use of Intertac.

Chapter 6 Patterns Description.

Looks at the student's actions that relate to each goal. The results of students' use of the basic Intertac interface are analysed for patterns of learning, and for pattern structure or processes used by students.

Chapter 7 Pattern Structure.

Looks at the feedback to be provided as scaffolding for the students' actions, relating to the goals of the learning. To implement scaffolding for the learning of the course, a section of the domain knowledge is presented in pattern format to be translated into agents.

Chapter 8 Implementation

Provides a comparison with previous results of research into what sort of work on-line requires scaffolding. Previous work looked at chat interactions or simplified learning situations.

Chapter 9 Conclusion.

The aim of this work is summed up with the description of a process of analysing and implementing patterns from observations of the processes of learning in a domain.

Chapter 10 Future Work.

The research is not complete as the Intertac-II software must now be expanded with agent implementation of the patterns and verified against the original aims for its development.

1.6 Prior Publications

The following chapters have had material previously published in:

1. Chapter 5: Technical report by C.Kutay: Intertac Software Architecture, Technical Report No. 0318, Computer Science and Engineering, University of New South Wales,
ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/0318.pdf, 2003.

2. Chapter 6: C. Kutay, P. Ho and G. Whale, Internet Based Groups in Computer Science: Helping Groups Work. Proceedings of ENABLE '99, Helsinki:HIT, 1999, 56–68.

3. Chapter 7: C. Kutay and P. Ho, Using intelligent agents for the analysis of students' interaction and learning, *Proceeding of AIED'03, Workshop on technologies for Electronic Documents for Supporting Learning*, 2003.

4. Chapter 7: C. Kutay and P. Ho, Assisting students by analysing their interactions and learning, submitted to *Proceeding of AIED'03, Workshop on Learner Modelling for Reflection*, 2003.

1.7 EndNote

Since this thesis refers to existing commercial software systems, it is acknowledged here that in all cases all trademarks used in this work are the property of their respective owners.

Chapter 2

Foundations of Learning Theory and Computer Supported Collaborative Learning (CSCL)

2.1 Introduction

The thesis covers a wide range of research, and so must necessarily focus on some specific areas, while only commenting on the relevance of other areas in passing. This research is put in the context of *Constructivist* educational theory for the development of course scaffolding. For analysing how students learn in a given context to develop the scaffolding the thesis draws on other methodologies.

The review of the methodology used in this work is provided in the Chapter 3, and also a description of the experimental context is provided in Chapter 5.

Another aspect reviewed here is the existing body of knowledge relating to groupware development for flexible learning and agents support. The advantages of software in collecting data on student actions provides a basis for looking at developing a formalism for analysing students' learning. By developing a formal language of learning, we can look at patterns of learning, an extension of the pattern languages which are reviewed here.

2.2 Background

Learning theory has been split for years between those who see it as a cognitive process (including Biggs [34] and Vygotsky [380]) and those who see it as a modification of behaviour (including Gagné [128]). There is also a division within the Cognitive strand between those who see all learners ability to accumulate knowledge as dependent on their inherent intelligence and the Constructivists (see Jonassen and Rohrer-Murphy [186] and Savery and Duffy [319])who consider the prior knowledge of the learner to be a significant factor in cognitive learning.

In this work we use the ideas of the Cognitivist Biggs who considered that learning outcomes can be related more or less to the Hypothetical Cognitive Structure of the student (see Biggs [34]), and the work of Behaviourists such as Marton and Saljő [243] and [244] who see the learning outcome as a number of qualitative different views on a specific content under study.

For Gagné [128] who is a Behaviourist, learning is progressive and, in that it occurs progressively, it does so by way of a **learning hierarchy**. In learning new material, the learner will draw on previously learned capabilities to acquire it. To learn *higher-order* material, the learner will rely on more remedial, previously learned material, to aid him/her in processing and acquiring the new material. This work is based on Behaviourism yet it is used often by Constructivists as a guideline for developing learning materials.

Instructors who work according to Cognitivist Theory provide the students with challenging tasks and expect them to make intellectual decisions about how to solve the problems. They can provide the intellectual **scaffolding** to help students learn and progress through the different stages of development, and this is also an idea that Constructivists embraced into their learning system.

When these ideas were first translated into learning materials on computer, this has often been done in a fairly unstructured manner, providing little guidance. This has arisen out of the view that when all the steps of the learning behaviour are provided, students may follow whichever step they wish at whatever time. While the learning hierarchy provides some necessary structure or linear sequence of skills to be learnt, students must be given some motivation through control of their own learning progression.

In an unstructured environment such as linked web pages, this can lead to confusion and students becoming *lost in hyperspace*. Work has been done on preventing this type of learning situation. In the work by Gygi [158] the system keeps track of where the user has been and provide tools to allow the user to backtrack, others such as Passardiere and Dufresne [282] and Eklund [112] advocate the use of visual maps representing the nodes of information and the links between them. Alternatively Learner Modelling can be used to provide feedback to students on their progress and their contribution, as in Kay [194] and Kutay [207].

In a learning domain involving **ill-structured** problems (see Spiro et al. [349]) the instructional systems based around Constructivist principles are more suitable. These are problems that tend to include large amounts of information. Rather than a single right answer they may have several acceptable solutions (see Vos and Post [378]). A learning methodology that incorporates interaction with an environment, cognitive conflict and negotiation of shared understanding, is Problem Based Learning(PBL) (see Savery and Duffy [319]).

The present research develops a Constructivist learning environment to support Project Based Learning in software engineering courses. Project Based Learning is a variant of PBL that provides a less unstructured learning environment, focused on the end product or project. Due to the complexity of this type of learning environment, it is necessary to use a detailed methodology for thoroughly researching different learning styles and patterns.

Social Constructivism, along with an offshoot of Behaviourism called Constitutionalism, have moved towards similar approaches to analysing learners. Social Constructivists have developed methods from Activity Theory (see Jonassen and Rohrer-Murphy [186]) and the Constitutionalists have developed the Phenomenological method. Both approaches promote the use of observation and reflection to improve teaching. Accordingly, these methodologies are used in this thesis for this same purpose.

2.3 Social Constructivist Learning

Constructivist Learning is based on the following principles(from Savery and Duffy [319])

1. Understanding is in our interactions with the environment, and cognition is not within the individual but part of the whole learning context.

2. Cognitive conflict or puzzlement is the stimulus for learning and determines the organisation and nature of what is learnt.

3. Knowledge evolves through social negotiation and through the evaluation of the viability of individual understandings.

A branch of Constructivist learning, with a focus on group learning, is the Social Constructivist view of learning as developed from Piaget and Vygotsky [380], in particular using the social emphasis of the latter. Social Constructivism is similar to the Radical Constructivism developed by von Glaserfeld [143], and it is interesting to look at this view first.

2.4 Learning in Groups

The present work falls within the tradition in education that promotes working in groups as being more effective for student learning. There is much debate about the validity of such an approach. While it is acknowledged that some individual efficiency may be lost, in that the members of the group that have greater understanding will have to spend time bringing the other members up to speed, there is a large body of research relating to University learning (including Deutch [103] and [105], Laurillard [213] and Davis [106]) that supports the move to group based learning.

Some notable studies of the success of this learning mode have been:

1. Joint effort during complex problem solving tasks improves both the process of achievement and the quality of the solution (Deutsch [105] and Laurillard [213])

2. If the emphasis is put on achieving a correct or good answer quickly, then a group has a higher probability of achieving this aim than an individual, all else being equal (Davis [106])

3. Group problem solving can often yield more alternative solutions than can individual work (Davis [106])

4. Groups perform better than individuals in tasks which afford a wide possible range of answers (Thorndike [359])

5. Pairs are better than individuals when working on problems requiring some originality of thought or insight, but not on routine mathematical problems

(Husband [176])

If we look at the Conversation Theory of learning, developed by Pask [281], we see the concept of learning described as a conversation. This conversation may be with the instructor or with peers:

1. In order to learn something, an individual must be able to report what he or she knows (i.e. engage in conversation).
2. The individual must come to know the subject, operate on it and receive feedback on the results of these operations.
3. To organise investigation of a subject, operations must be generated from a global framework and must receive feedback on the results of the operations.

These are the process the software developed in this thesis is designed to support.

Many educators including Laurillard [217] have highlighted the success of University learning in groups where this **conversation** is carried out between peers. In the area of cognitive apprenticeship, Brown, Collins and Duguid [58] have noted that learning in groups is favoured as it provides for:

1. Collective problem solving which not only accumulates individual knowledge, but also provide synergistic processes to insights and solutions. In this work we show how the software analyses different in individual and group input.
2. Displaying multiple roles. Students are exposed to the many different roles needed for any cognitive task. Different roles are displayed by different members, and reflective narratives and discussions are encouraged. In this work we show the sort of tools that can be provided to support the different roles.
3. Confronting ineffective strategies and misconceptions. Teachers have little time and exposure to each individual, but groups provide opportunity to draw out,

confront and discuss these. In this work agents provide questions and feedback on concepts

4. Acquisition of collaborative work skills. In this research the agents are designed to provide feedback on individual interactions.

Furthermore, Dillenbourg et al. [101] have studied the problem of grounding, or *reaching a common understanding* on a topic under discussion. When two partners in a discussion misunderstand, they have to build explanations, justify themselves, often make explicit some knowledge that would have remained assumed, monitor each other and thus reflect on their own knowledge, processes and so on. These mechanisms are crucial in collaborative learning and may explain why this is sometimes more effective than learning alone.

When courses and projects are provided in distributed mode, this becomes a challenge to provide group communication resources that match those of collocated learning as closely as possible. Assuming the learning is taking place within groups, with co-operative task structures, the aim is to develop effective groups. Much of the research around analysing or developing effective groups relates to enumerating the main processes or functions involved in effective group communication.

The main processes required for effective group processes for carrying out a task as well as learning a task assumed here are (Soller et al. [341], Barge and Hirokawa [26], Barnes and Todd [27], and Shaw et al. [334]):

1. Participation

2. Conflict and resolution

3. Collaborative or supportive skills

4. Facilitator and reminder roles and rotation of these

5. Allocation of and responsibility for duties
6. Division of work into separate units or integrated work
7. Immersion in group processes or promotive interaction
8. Social grounding
9. Performance review and reflection

If we look at studies of the psychology of groups, there are patterns of group development such as the simple model described as Forming, Storming, Norming then Performing developed by Tuckman (1965) and patterns of group processes such as planning and leadership. Also there is the Gibb Model [137] which describes the following concerns for social interactions:

1. Acceptance and the development of trust, place in group, increase in confidence and reduced anxiety
2. Flow of the group in terms of perceptions, feelings and ideas through the group
3. Goal formation including goal setting, problem solving and decision making in the group
4. Control including regulation, co-ordination and implementation of activities

Finally there is work on dysfunctional groups (and dysfunctional learners such as work by Shelley and Shelley [335]), which is beyond the scope of this thesis. It is not assumed the computer can deal with groups that fall outside the domain of learning and interacting groups. These groups can be dealt with by the tutors who receive logs reports based on the level of interactions and learning diagnosed by the system.

When designing tools for distributed groups the aspects of group interactions and planning constitute the conceptions of group work in which we would like to train the students. As well as teaching the domain, the groupware can be used to teach about groups, group dynamics and group process.

2.5 Problem Based Learning

The students being studied are working in a Project Based Course. This is a form of Problem Based Learning (PBL) which was developed in Australian Universities through the work of people such as Boud, Feletti [48] and Foley [126]. Boud assumed experience as the base for learning and developed a framework to provide courses that are user-centred and rely on learning from experience. Problem Based Learning (PBL) involves learning from real-life problems, related to the profession under study, was used first in Australia in teaching Medicine. PBL has now extended to many other professions, including Computer Science as it is a methodology particularly appropriate to teaching practical skills. The theory of learning and course structuring under PBL has been well developed and exponents of this method regularly refer to examples of courses developed using this method to explain their approach.

PBL exemplifies the Constructivist principles of learning as presented by Slavery and Duffy [319]. The points that relate to the software developed in this work to support the PBL course are:

1. Design the task and learning environment to reflect the complexity of the environment that students should be able to function in at the end of learning

2. Design the learning environment to support and challenge the learner's thinking

3. Encourage testing ideas against alternative views and alternative contexts

4. Provide opportunity for and support reflection on both the content learnt and the learning process

PBL begins with a problem for students to solve or learn more about. These are often framed in a scenario or case study format. They are **ill formed** and imitate the complexities of real life situations. The approach uses an inquiry model: students begin by organising any previous knowledge on the subject, posing additional questions, and identifying areas where they need more information. Students devise a plan for gathering more information, then do the necessary research and reconvene to share and summarise their new knowledge. Students may present their conclusions and there may be an end product, but there should be adequate time for reflection and evaluation.

The problem is the driving force, although the solution may be focused on to some extent. Some problem based approaches intend that students should clearly define the problem and arrive at a clearly stated solution. (e.g. [14]). Others concentrate on the learning and information gathering and may have no solution (e.g. [383]). The practitioners in this field of education find the most difficult thing is to explain what a *problem* is. A problem in this context has been defined as *a question put forward for consideration, discussion, or solution.* Others find this too vague and turned to the more concrete *project based* learning. The model of Project Based Learning has provided clearer guidelines for the learners to enable them to cope when the change from lecture style to research style learning is encountered.

Another area of contention in developing courses for Problem Based Learning has come from tutors and lecturers. These people are used to being the authority who transmits knowledge to the students. In their new role they are encouraged to *not answer questions* but instead to propose ways of coming to an answer. When they take this too literally, students find their questions are all being answered with an evasion or an expression of lack of knowledge by the tutor. Students then come to perceive the learning process as useless and overly demanding and the staff as ignorant.

Experience from tutors in PBL courses has provided some guidelines for producing support agents in software (e.g. Ambury [15]). In developing software to support learning, designers are trying to replicate the tutor's work online (see also Koedinger [200] and Lepper [221]). Software resources provide an ideal method of ensuring information is available on-line or through menu help links to provide immediate feedback to students' queries and concerns. In PBL the tutors need to learn how to facilitate learning. In software this converts to the need to develop facilitative agents and provide advice for the human tutors who receive filtered data from the groups.

2.6 Context, Consciousness and Content

While it is beyond the scope of the thesis to study the implementation of PBL courses, is it worth noting how such courses are implemented, as this guided the creation of the software to support such courses.

Firstly, the actual learning that occurs relates to the content of a domain and how this is manipulated and integrated into the students knowledge. In general for constructive learning, ill-structured problems are favoured. It is in fact the flexibility of the teaching in PBL and the ill-formed nature of the problems that makes the software support hard to develop.

Secondly, students who do participate in such courses, have different priorities in accessing their lecturers and tutors, or a different consciousness of what constitutes a good teacher. For instance students particularly in later years, tend to identify factors in teachers such as (from Mullen [260]):

1. keeping students on track

2. asks lots of thought provoking questions

3. involves class in discussion
4. advises on how to study
5. able to provide options for possible solutions and
6. able to connect different pieces of information

Finally, it is worth noting that as part of the implementation of such a course, and hence of such software to support learning, various aspects are required. The ones that relate to software support include (Lahteenmaki [212] and Oriogun et al. [276]):

1. Students must be sufficiently informed about the new way of learning at the start of the course which could be provided by web pages and hyperlinks to further references, and regular feedback during the course.
2. Students need to be informed they are learning enough, or motivated to work if they are not, with regular feedback in tutorials, or online.
3. Students need to encounter different practical experiences during the different phases of their education, such as roles and different problems, which can be added to their experience of the software during the progression of the course. The software has to develop and change as they progress.
4. Co-operation between students, instructors and practitioners is crucial to the success of the course, and the development of the software in this work relies on the support of teaching staff assisting with the development of the agent analysis algorithms and feedback.
5. As in developing any flexible delivery course, many people of a wide range of skills are required to work together to implement the new course format, and this up front commitment is much greater than for conventional courses.

6. Courses must be planned more in unison, so that the learning from one course to subsequent courses is a smooth transition.

7. Teaching such a course requires a commitment to work side by side with the students, as a crafts person would, and provide an open and supportive interaction with students which values their approach and contribution to the problem and its solution.

There are other factors also relating to teacher acceptance of the course approach [254] that are perhaps the hardest to achieve in a school unless fully supported by the administration. Only then will students be sufficiently supported through their uncertainty in the new learning mode.

Another aspect that is again not within the scope of this work is the cultural factors that are important in any learning and teaching development at The University of New South Wales. The University consists of a largely international or Asian Australian population and hence the language and cultural differences between students, and between student and the majority of Anglo-Saxon lecturers can be extreme and cause problems.

Furthermore the cultural difference in the way people both learn and interact in groups will affect the use of the tool and the required interface. This in turn will effect the validity of the analysis by learning support agents. The ability to analyse student interactions and learning processes is limited by the mix of cultures in Australian universities. Work done on GSS [385] suggest that there is a need to consider how group practices vary and change the effectiveness of different software to support groupwork for different cultures.

As noted by Dixon [102] in a presentation at the Third Asian Pacific Conference on PBL in 2001, when students in Hong Kong are studied while working in small groups, they focused on discussion with the the tutor and avoided student initiated discussion. Also if the tutorial work, or in this case the group communication, is in a language that is not their native tongue, students may be even less engaged.

2.7 Depth in Learning

Phenomenographical studies of learning have developed an approach of categorisation of students' conceptions of learning as well as their approaches to learning (see Marton and Säljö [243] and [244]). The approaches to learning are divided into deep or shallow.

Deep approaches have been shown to correlate to effective long term learning whereas shallow approaches have not. The aim then becomes to develop environments for learning that encourage and support deep approaches. Säljö [318] classified approaches to learning text into five categories:

Surface Subjective Learning as a quantitative increase in knowledge. Learning is acquiring information or knowing a lot.

Surface Restrictive Learning as memorising. Learning is storing information that can be reproduced.

Surface Elaborative Learning as acquiring facts, skills and methods that can be retained and used as necessary.

Deep Integrative Learning as making sense or abstracting meaning. Learning involves relating parts of the subject matter to each other and to the real world.

Deep Reflective Learning as interpreting and understanding reality in a different way. Learning involves comprehending the world by re-interpreting knowledge.

Presenting an environment that encourages abstract conceptions, or meta-learning, requires a context with certain characteristics [242]:

1. a well structured knowledge base,
2. interaction with the instructor and other students,
3. learner activity and
4. motivational content.

Biggs [41] used his study of writing processes to develop the SOLO Structure of Observed Learning Outcomes, which provides a different approach to a similar analysis of the deep/surface approach to learning. The SOLO structure hinges on the different level of abstraction of the content of students' written answers as compared to the structure required by the question. However this structure of the question as perceived by the student is dependant on their approach to the learning domain, or the categories developed by by Marton and Sắljő [243].

Biggs classified abstraction levels as follows [37], and included a link to the deep/surface division in learning as listed above:

1. Pre-structural: question not addressed and the discourse structure is inadequate. Approach is surface-subjective.
2. Uni-structural: question addressed in only one relevant line of argument. Approach is surface-restrictive.
3. Multi-structural: question addressed with more complete listing of knowledge, but the genre is still inadequate. Approach is surface-elaborative.
4. Relational: requirements of question are satisfied. Approach is deep-integrative.
5. Extended abstract: question answered at a higher level of generality and abstraction than the case needs. Approach is deep-reflective.

Biggs [34, p.597] concluded that in relation to their separate work on categorising the outcome of learning:

> In short, despite the theoretical difference, Marton's work is (on this point) directly comparable to our own.

The relationship between the approaches to, and outcomes of, learning is exceptionally strong. Studies in history, computer studies, and text processing show that the surface approach, almost without exception, leads to a quantitative outcome of unstructured detail and a deep approach to an appropriately structured outcome (see Biggs [36] and [37], Marton and Sǻljő [243], van Rossum and Schenck [372], and Watkins [384]).

Later Phenomenographical research by Booth [45] into students' perceptions of programming found a rather different approach to the deep/surface divisions. She found the divisions between a deep and surface approach to be more a matter of the number of skills developed in a programmer, or the alternative approaches at their disposal to handle different situations.

Booth developed the following categories to describe students' approaches to programming:

Expedient which focuses on producing a complete program from the outset by using existing programs or adapting a known program

Constructural which focuses on recognising details of the problem in terms of the features of the programming language, the constructs, functions and keywords.[1]

Operational which focuses on writing a program based on an interpretation of the problem within the domain of programming; the problem if considered from the point of view of what operations the program has to do.

[1]The term constructural is used by Booth [45] to define an approach that is structural but not organised in a coherent manner e.g. bottom up designs in software.

Structural which focuses on writing a program based on an interpretation of the problem within its own domain; the structure of the program is first considered and then a program is devised.

There is no comparable research to the work by Booth, into other aspects of software engineering such as Requirements Engineering, Design and Specification. The research into learning experiences in software engineering conducted as part of this thesis is an extension of this previous Phenomenographical research and shows similar categories of depth of approach to learning are developed in the new area researched.

As a corollary to this section, it should also be noted that any study of the **depth of approach** to learning is culturally sensitive. In the work by Marton et al. [246] with Asian students in Hong Kong it was shown that the notion of learning in depth can include learning by rote. Asian students are analysed through interviews as learning by rote with the intentions of thus understanding what they are studying, a concept that is not familiar amongst the European students studied previously by Phenomenography. However, this aspect is not as significant in the present research as little opportunity for supporting learning by rote occurs during group activities, so this fell outside the present domain of study.

2.8 Designing Software for Learning

The thesis is in the area of the design of Computer Supported Collaborative Learning (CSCL). In particular it derives part of its work from the theories developed through Problem Based Learning (PBL) research and provides the initial steps in developing tools to implement this learning style in distance mode. This differs from other work on Computer Aided Learning (CAL) which focuses on Intelligent Tutors, such as Brown and Burton [57] [64], Van Lehn [371], Collins et al. [80], Koedinger [200]and

Anderson [19]. The domain is more unstructured, the problems ill-formed, and the analysis hence much more difficult. As observed by Reusser:

> machine-tutoring based on cognitive simulations of the student is not possible across a full range of open-ended tasks and domains, where fuzzy language and qualitative world-knowledge based reasoning are required. This is especially true with regard to error modelling. [305, p.45]

Groupware has been developed in other projects. Most have not been implemented outside controlled classroom settings except GroupKit and Habanero [159]. These systems have been developed to study some aspect of learning interaction and frequently lack the tools necessary for full text and graphical editing, and the development of a report from these. The present research was therefore an ambitious project in terms of developing groupware for the experiments. However the tools required for a full analysis of learning and group interaction, with augmentation through agents support, was not available.

Therefore a new system is developed, based most closely to the Habanero [159] design. This section explains why this approach was taken.

2.8.1 Supporting Group interactions

There have been other systems attempting to support group learning. COLER (Constantino and Suthers [85]) is a web based collaborative coach that supports the combined activities of students, rather than those of an individual. This is similar to work done using the Matchmaker system (Műlenbrock and Hoppe [259], where a task analysis approach is used to monitor and analyse the moves of multiple users in a shared workspace. In COLER, advice is generated primarily from comparing students' individual solutions that they produce first, and the subsequent group solutions. Also individual participation in group chat is tracked. This approach does away with the restrictive use of sentence openers, instead using expert domain knowledge on significant differences in the diagram structures and interactions that promote group learning.

COLER is evaluated for the feasibility of the method of advice generation by a comparison of responses from COLER with those of a human expert to the same group of diagrams. Constantino and Suthers [86] show that the experts find most of the advice reasonable, although only about a third of the advice is what they would have given in that context. In consideration of this result, it is noted that the COLER advice is done in real time as a reaction to the learning process, while the expert works on a delayed analysis of finished products.

Collaborative Case study systems such as LeCS (Learning from Case Studies developed by Rosatelli and Self [315]) provide agent support for the learning within well formed domain. In LeCS, a pair of subjects work at a distance to solve a case study. The software is set up with the resources of the case study and the process that students are led through during the tutorial. The system used script based representation of *standardised general episodes* to support interactions and initiate interventions in the case of misunderstandings. The LeCS system used three types of agents: recording agents, storage agents, and advisory agents. The recording agents tracked events such as the users' actions, the steps they are on, and timing and duration of user-initiated events. The storage agents stored didactic material and knowledge bases relating to the domain of the case-specific utterances and the case study solutions as they are developed. The advisory agents recognised misunderstandings and the pair's progress through the steps.

Swigger et al [357] have produced tools to train programming students in the specifics of co-operative work. The effective skills selected are used as behavioural models to be encoded into individual tools for the system. In particular, the tools enabled them to organise and systematise their interactions. In this research a new groupware system is developed as it is felt that each existing system:

> seems effective in supporting collaborative work, but lack the coaching capabilities that enable participants to *improve* their collaborative skills [358, p. 4].

Soller et. al [341] presented strategies for helping students attain these learning skills, such as:

1. Encouraging participation with introduced brainstorming sessions or animated agents to partner a student and build their confidence to speak to the full group.
2. Software agents can encourage students to elaborate upon and justify their reasoning by playing the devil's advocate.
3. Assigning roles in a rotating basis and using agents to simulate missing roles.
4. Providing students with collaborative learning skills usage statistics raises awareness about the type of contributions they are making.
5. Providing feedback on students' performance may provoke group processing of this information.
6. Assigning human mentors to students and assigning roles to students to encourage answering and elaboration.

2.9 Analysis of Interactions

The thesis focuses on two main areas of learning support, the first is supporting the interaction within the group. The area of language analysis is useful for developing agents that can interpret and support communication in Chat tools. Language analysis can cover the computer's interpretation of student's contributions and the generation of natural language for the response by the computer to student actions.

Some software systems have classified the variety of speech acts [2] of teachers and attempts are being made to look at more complex patterns, such as the combination of speech acts to form more complex types, and the existence of a range of speech acts which can be used by a teacher to fulfil similar communication functions in identical educational contexts [292].

At the same time the agent or software needs to understand students' contributions, particularly on chat channels, in enough detail to generate suitable responses. Yet to break down even basic **Chat** text on computer into these categories, some form of natural language analysis of the program is required. The work by Burton on semantic grammars (cited in [390]) provides the best approach towards formalising interpretation of natural language input. This grammar is first implemented with procedural rules, and then later replaced with *augmented transition networks* [66]. Yet even with such a sophisticated approach, the noise introduced by the existing student's lack of familiarity with English would be a major drawback, especially at the University of New South Wales where the non-English-Speaking-Background (NESB) student population is high. Some natural language analyses use graphical unification algorithms to distinguish between sentences that are either *orders, questions* or *telling.* [178] and this simpler format may be useful in future augmentation of the software developed in this work.

The technique used in this work is to provide students with a list of descriptors for each speech act so that this analytical work is already performed on the text by the students themselves. The language attributes of the Collaborative Learning Conversation Skills Taxonomy developed by McManus and Aiken [231] are used as chat prefixes. Also secondary descriptors in the form of topics of conversation can

[2]Speech Acts can be described as activities in speech format

be selected. Finally the line to which a user's response is directed is included in the response line.

As an initial outline of the sort of analysis that can be achieved with this approach, we used the categories of O'Malley [272]:

1. Turn taking with smooth alternation for periods of understanding and withdrawal or disagreements for periods where pairs are not engaged with each other
2. Socially distributed production where one person begins a sentence, and the other completes it (if, then) forming a product or conclusion
3. Repairs such as justification, counter-suggestion, assertion and elaboration
4. Narration that describes action when a single person has control of the action
5. Language in action which can present new ideas by actions and can accept an idea by doing it.

The aim is to provide increased **grounding** of communication, that is the process of adding to the common ground of mutual understanding or belief about what is said and meant, between agents (in this case human). Grounding acts also have a cognitive effect in enforcing a re-phrasing of the subject matter. Traum and Dillenbourg [364] recommend providing resources to enable users to obtain some benefit from communication repairs. For instance structured communication interfaces that support negotiation (see Baker and Lund [23]) have been developed.

2.10 Analysis of Activities

As well as studying interactions, we look at supporting the various learning activities. In supporting and analysing the activities of students in the learning domain of software engineering, there are many aspects to consider. The student's work online involves, amongst other things, writing a report, working in a group and developing a software design.

The cognitive approach to writing has been well researched, particularly from the point of view of writing as a problem solving task (Flower and Hayes [125] and Humes [173]). Biggs [37] presented an approach to writing analysis with these activities:

1. Intentional, including affective and aspirational aspects prior to writing, or motivation.

2. Para-writing, including planning and knowledge updating.

3. Writing, including composing, transcribing, reviewing and revising.

The present learning situation involves writing a report and so includes many of the same aspects.

Webb [387] discusses students' help to each other in groups and the same principles apply when considering the design of computer based assistance in tutoring systems. For instance students who are learning must then try to use that learning themselves. Also learning may be difficult due to the problem of incorporating new concepts with little prior learning to construct them from [393]. If the work is beyond the cognitive ability, or the zone of proximal development is exceeded, the student will not be comfortable (and hence not motivated), or have a basis from which to develop new concepts or plans (or construct new learning).

2.11 Design of Agents

The agent design is developed as a pattern language which is in term derived from the learning ontology and its formal structure. From Akhras and Self in [11] and [7] we have derived the main components of learning process as:

1. The context of learning including content, dynamics, objects and tools
2. Process entities for the interactions between the learner and the environment which lead to the development of learning
3. Analysis Units of the process entities
4. Analysis of learning interactions through patterns, relation to context and learner's knowledge and patterns between learning situations, evaluated in the Analysis Units.
5. Analysis of the interactions or time-extended processes which are either cumulative, constructive, self-regulatory or reflective.

This suggests a framework for the agents that will perform the analysis of learning and interaction based on assumptions about the context. These are stored in a learner modelling framework. The agents then will take action (such as give advice) based on planned interactions with the context (both learning and group interaction). The enactment of the plans will be controlled by the interaction between agents, which specifies the process of agent action over time.

2.11.1 Patterns and Pattern Languages

Describing scaffolding of learning activities in terms of patterns aims to develop a similar format for specific types of scaffolding to enable their description in a more universally understood manner. The concepts of patterns and pattern language were developed by Alexander [13] in the field of Building Architecture and Town Planning. This had been extended to the area of software engineering and even groupware design [328]. The present work is an attempt to derive *patterns for learning*, to present teaching objectives in terms of a pattern of what is to be learnt.

Alexander's original focus was on the interactions between physical form of the environment and how this inhibits or facilitates individual and social behaviour with the environment, or how the product relates to the actions of the user. Patterns were to provide a facility to share knowledge about design solutions and the setting in which the solution is applied, so they:

> ... must be formulated in the form of a rule which establishes a relationship between a context, a system of forces which arise in the context, and a configuration which allows these forces to resolve themselves in that context (Alexander et al [12]

Bayle et al [29] see patterns as emphasising the characteristics of the environment that might facilitate or inhibit action. A design pattern describes a recurrent problem or theme and gives an abstract solution to this problem in such a way that the solution can be applied to different instances of the same problem. Patterns should also describe the rationale and trade-offs involved in the solution.

If these pattern characteristics are applied to socio-technical systems such as software design, a good pattern must capture the essential elements of the software system, and how the form of this system facilitates and inhibits desirable individual or social behaviour.

Maiden [234] developed patterns for **requirements engineering**, in an attempt to inform the validation of system requirements. He mapped key elements of Alexander's patterns as follows:

Form of the Environment maps to the system's functional and non-functional requirements as presented in a requirements document,

Desirable Behaviour maps to system scenarios which are a sequence of events and actions which describe desirable future system use in the environment, and

Good Design maps to patterns which capture the requirements which facilitate or inhibit the scenarios.

Another approach to pattern generation in the domain of co-operative interaction, such as used in the design of CSCW systems, is presented by Martin et al [241]. In this research they look for generally recurrent phenomena to develop the basis of a descriptive (rather than design) pattern language.

This present pattern language is developed to enable teaching objectives to be implemented as component based agents that augment the existing groupware system. As the ability to analyse student's activities increases, the agent support can also be augmented. However the pattern structure will provide a static framework in which to develop such new support.

One of the fundamental aspects of patterns is that they are used to describe the link between the artefact, in this case the software agents, and the user. In software agent systems, patterns rely on the form of analysis of the user that is available. This may be through modelling of the history of the users' actions or through direct analysis of actions. The aspects of the action that are chosen to describe a pattern should reflect the formal language used to implement the pattern. This is based on the ontology of learning developed in Akhras and Self [7].

When the pattern structure is developed the next process is to define a language for the interaction of individual patterns. This should also relate to the language of implementation. Hence the way teaching or learning objectives interact will determine the pattern structure. This led to a need to understand the learning process that is being supported by the agents.

The pattern language defines how various embodiments of the pattern relate to each other as well as the rules of this relation. For a pattern language, this work uses the structure of learning and the analysis of **depth of the activity** of learning. The links between patterns relate to the inclusion or generalisation of ideas within more general ideas, the broadening of concepts by links to related concepts, and so on, as developed in this thesis.

2.11.2 Semantic Networks

The patterns of a learning domain and their inter-relatedness can be represented graphically as a net or semantic network as developed by Quillian [295]. Semantic networks are:

1. A graphical notation for representing declarative knowledge in patterns of interconnected nodes and arcs.
2. Easily designed as a computer implementation
3. Knowledge representations or support for automated systems for reasoning about knowledge.

There are many forms of semantic nets that have been used in groupware systems.These show knowledge relationships between different concepts or states and the knowledge can have different strengths, varying from assumptions to message passing. The various forms are:

Definitional networks that emphasises the link between different information or concepts, such as generalisations, subtypes, etc. They include graphics and knowledge tree structures, and have been used in the development of Knowledge Based Agent Languages such as KL-ONE [53].

Assertional Networks that are designed to assert propositions and the information contained in the network is assumed to be contingently true. Such networks include conceptual graphs used to present the first order logic in language, and include concept diagrams. A concept diagram tool provided by the groupware Belvedere described by Suthers et al. [355] to assist students to construct their own knowledge representation.

Implicational Networks connect nodes by implication rather than assertion. Depending on the interpretation of the arcs in the network, these networks are called belief networks, causal networks, Bayesian Networks or truth-maintenance Networks. For example such Bayesian Classifiers have been used to estimate the probability that a document is liked during web searches by Pazzani and Billsus [284].

Executable Networks include mechanism such as marker passing or attached procedure passing which cause some change to the network itself and perform inferences, pass messages, or search for patterns and associations. Data Flow Diagrams are executable networks, as are Petri Nets, which have been used to formally describe collaborative learning processes such as creative conflict, through analysis of the pattern of interchange in sentence openers used in groupware by McManus and Aiken [232].

Learning Networks build or extend their representation of knowledge by acquiring knowledge from examples. The new knowledge may add or delete nodes or arcs or may modify the value or weight associated with the nodes and arcs. These include neural networks, and have been used in COLER to analyse participation and problem solution differences by Constantino-González and Suthers [87].

The approach in this work has used Executable Networks, since the aim to to produce implementable patterns. Also the networks are a series of events the group may pass through in their learning. By developing a model of this process we both retain a record of the history, as in a learner model, and provide the series of steps that terminate in knowledge or lack of it.

2.12 Conclusion

This chapter provided an overview of research prior to the work of this thesis on matters relating to education theory, as relevant to the design and development of a computer based learning system for a project based learning course.

This thesis uses constructive and collaborative learning theory as the basis of the learning support design, or the assumed method of learning to be developed through the software approach. The specific process of learning that we are to develop in the students is based around Project Based Learning (PBL). PBL draws on aspects of cognitive apprenticeship and requires developing motivation and metacognition in the students by the instructor, or the instruction materials.

To analyse the learning by students, this thesis uses behavioural techniques such as Phenomenography (to analyse learning) and Activity Theory (to analyse students on the software), since the computer can only provide information on student's actions. While interviews are conducted to back up this research, the final outcome of the research is the agents that rely solely on the actions of the students using the software.

As an adjunct to the learning theory is the area of group processes, that must be understood to study how the groups interact in their learning and analyse for successful group work. In particular, teaching through project learning requires that group work becomes part of the learning domain.

The models developed for providing the scaffolding in this learning domain include models from Concept development, Motivation, Metacognition and Group Interaction. The scaffolding is designed to vary according to the learning approach of students, which is analysed by the software as either showing the activity of deep or surface learning.

The scaffolding is implemented using software agents which are designed using a Pattern Structure which is in turn developed from the structure of ontology of learning. Hence the research starts with developing an understanding of how students take different approaches to learning and how these translate into depths of activity, in the context under study.

Chapter 3

Methodology of Research

3.1 Introduction

The experimental research undertaken for this thesis covers three related aspects. These form a sequential link for the development of proper mediation in a computer based group work system.

The first experiment analyses the approaches to learning used and what constitutes deep learning experiences by students working in collocated groups in the software engineering workshops studied. The second experiment includes the design of software that provides a framework to encourage students to be involved in flexible group learning. Finally the third experiment is to analyse students' activities in groups during the workshop to ascertain the actions that reflected different experiences of learning and support these for all groups through learner modelling and advice agents.

These studies are carried out through case studies of many groups of students, initially working in collocated groups. When the software is developed we studied students working on the software as remotely located groups. The research is based on broad case studies of a variety of group formations and problem situations that may exist during learning in the domain of the software engineering workshops.

The purpose of this chapter is to explain the theoretical foundations of the research methods used to analyse students' approaches to learning (Phenomenographical research), to develop software based on the User requirements (Activity Theory), to analyse their patterns of learning (Action Research) and to develop a formal agent language to provide support and finally to derive Learner Models to enhance learning.

3.2 Context of the research

The course in which the experiments are run in a Project Based Course where students are given a problem domain for the development of software. They must collect the requirements for the software, design a solution for the context (using Data Flow Diagrams) and write a specification report for the design. The specification is written formally in the B language, using a toolkit to support the writing of the specification.

The software developed and used in this thesis is external to the toolkit and is used for the communication within the group to develop the requirements, design and report, but not the specification section.

The work is performed in groups of three to four students, and students divide the work up and co-ordinate as a team. However the aim of the course is that they also learn from each other as to how the different parts of the final document are developed and for them to all be involved in integrating the parts.

It is a form of learning where, due to time constraints, groups do already work remotely using files sharing and Chat channels. However the level of exchange is low and learning still tends to be individually. Also the domain of learning is substantial, encompassing requirements elicitation from the users, aspects of design and specification, as well aspects of as group or team work.

The aims for the software system for online group work which is developed in this thesis are:

1. to make the students and their synchronous work more accessible to each other, at least in their group; and

2. to provide some super-structure or guidance which is synchronous with the learning taking place in each group.

It is acknowledges that the skills required to produce software design effectively in groups, are varied and often hard to specify:

> Unfortunately, predicting which [human] factors will be significant [in software design] is very difficult. This is especially true for engineers who have little experience of social or cultural studies. (from Sommerville [348, p.40])

A significant portion, if not all, of the learning is gained through solving the problem assigned to the group for development. The resources available to students are: prior learning; the lectures; weekly tutorials; text books; web references to existing systems of the type being generated; and their fellow students.

Some of the concepts used in the diagrams and the programming differ to the students' prior experience, so require some practice by the students and support by the staff. For instance specification models can use parallel operations but not sequential ones. This requires students to rethink processes that they may have designed before in another language, or are more used to thinking of in sequence. Many students miss this aspect and continue to program as for a procedural language.

3.3 Case Studies

The experimental methodologies used in this thesis are based in the Grounded Theory of research (see Glauss and [141]. Grounded Theory is used as a methodology for research that is: *inductive* in that it aims to propose a new theory, rather than to verify it; *contextual* in that the complexities of the context are incorporated into the understanding of the phenomena studied; and *process related* in that it facilitates the generation of theories of process, sequence and change pertaining to systems [277].

The general goal of grounded theory research is to construct theories in order to understand phenomena, in this case the experience of learning and how to provide learning support. A good grounded theory is one that is:

1. inductively derived from data,

2. subjected to theoretical elaboration, and

3. judged adequate to its domain with respect to a number of evaluative criteria.

This work is carried out using a case study approach as described by Yin [397] rather than statistical analysis. During the session course, a number of students groups agreed to have their work monitored and to do interviews. The output of this work is analysed.

While the work is limited to volunteer groups and to a small number of groups, there is some flexibility in that some groups are formed from volunteers by the researcher, and the groups varied greatly in composition as show in the Table 6.2 in Chapter 6.

The aim of the research is to extract the generic approaches that are common to different students who are learning at different levels. A statistical analysis does not allow the breadth of approach that is taken in this research for modelling the students intentions and actions in each situation that is studied. The grounded approach is used as the aim of this thesis is to develop theories of learning support through software mediation rather than proving the accuracy of these theories as is done through controlled experimentation.

The case study approach is described as an empirical enquiry which investigates a phenomenon which is contemporary and does this within the real-life context. The approach is especially appropriate when the boundaries between phenomena (in this case the differing depths of activities in learning) are not clear (see Yin [397]) as in this case. The methodology gains its validity by considering multiple sources of evidence. Data is required to converge in a **triangulated** fashion.

3.3.1 Process of Case Studies

The case study approach used is the structured-case study method of Carroll and Swatman [70], which uses a formal structure for its conceptual framework and for the research cycle. The framework used in the present research involves an explicit description of the pre-defined conceptual structure used in the research, rather than assuming total objectivity on the part of the researcher. The structure is derived from the research themes, the literature research, personal and professional experience and the world view of the research which forms the theoretical foundations.

The research is designed around an iterative research cycle of Planning, Data Collecting, Analysing and Reflecting. The last stage of the cycle in each of the following chapters produces the new conceptual framework for the next iteration in the subsequent chapter.

The experiments and the subsequent analysis are planned according to a process that related to the type of analysis to be done. In order to carry out a case study of groups, ten of collocated groups and twenty online groups of volunteers are gathered from a software engineering workshop course.

3.3.2 Variations within Cases selected

In the case studies different types of groups are selected to reflect the range of groups expected in the workshops in which the software is used. The groups of student are chosen to include groups that contain students with learning experience in terms of years of study or course studies that are either all similar or with some students being different to the rest of the group. These cases are used to allow the research to encompass the effect of the zone of proximal development on the actions of peer learning. Similarly the inter-group difference in terms of years (or domain) of common experience are studied to enable the analyse to encompass the effect of prior learning on group actions.

Within this broad range of groups the effect of the depth of the learning approach of the group is considered. This is extracted from interviews with the groups before, during and after the analysis of their interactions and learning are carried out.

The approach of this work is not to compare and contrast the different group formations with respect to range of learning experience, but to examine the range of possible patterns of work and support needed. When this support is developed, it is linked to the type of activities that signify the need for this mediation, such as the depth of prior use of a concept effects the concept description given, but otherwise the specific features of the groups are not further distinguished in this work.

3.3.3 Limitation to Case Study Research

The research uses a structured-case study approach which deals with the threats to different types of validity as follows (see Carroll and Swatman [70]):

Description Validity The conceptual frameworks developed clarify the research objectives and the iterations of the process ensure that initial data analysis guides further data collection and that there is collection of complete and accurate data.

Interpretive Validity The researcher's theoretical foundations are explicit as are the reflection stage which involves examination of the process, identifying biases, challenging current interpretations and highlighting inadequacies.

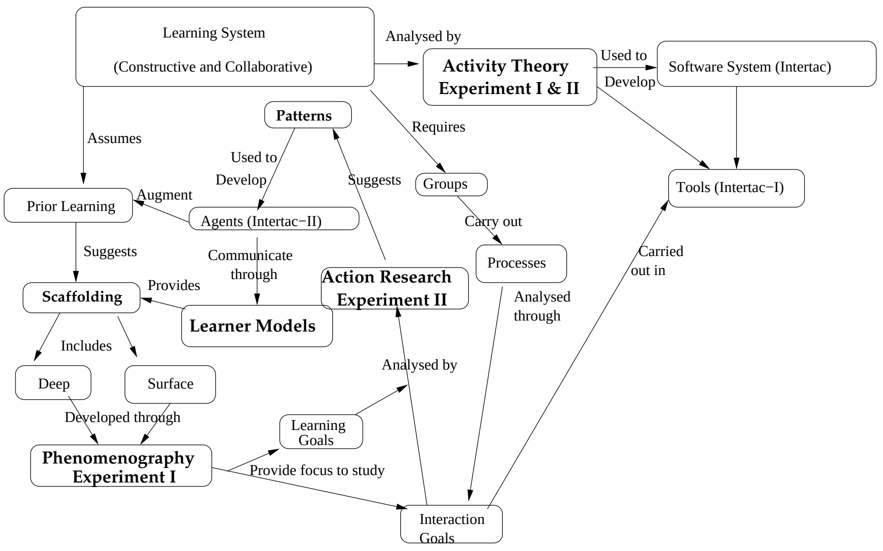


Figure 3.1: Concept map of processes used in Thesis

Theoretical Validity The reflect stage requires researchers to seek and consider counter evidence and alternative explanations. Also external sources are used when revising the conceptual framework at each cycle.

3.4 Methodologies Used

A summary of the research methodologies used in the experimental work is shown in Figure 3.1.

The following sections describe the methodologies in relation to the present work which covered two areas of experimentation: collocated and online. The collocated work involved two sections: the collocated learning approach and the design of software to replicate that mode of learning online. The online experimental work involved two sections: improvement of the software and development of agent support. The methodologies used are:

1. Phenomenographical study of collocated learning

2. Activity Theory study of online learning to develop and improve software

3. Action Research in software usage to develop agents

3.5 Phenomenographical Study of Collocated Learning

The analysis of students' depth of approach to learning in the domain studied uses a Phenomenographical approach. The Phenomenographical study of learning provides a methodology for analysing what and how students are learning(Booth [45]). This involves looking at the acquisition of understanding, use of higher order thinking skills and the depth of approach to learning.

Similar to Constructivist theory, Phenomenography tends to look more at how people learn, at their experience, rather than what they learn, or the structure of that knowledge. This is partly because it is assumed that the *how* or the act of learning is intrinsically related to the *what* or the content of learning. The various conceptions by a student of a subject are studied and categorised for their depth or shallowness and this is seen as an outcome of their learning which is related back to the context in which they learnt.

It is these similarities of how to study the social context of learning and students' experience of it that motivates the use of the Phenomenographical methodology to analyse students' learning in order to design the learning contexts that support appropriate approaches to learning. Also like Constructivism, Phenomenography seeks to accommodate the idea that it is logically impossible to produce deterministic predictions about the effect of educational situations on people, as people react to the requirements they see, not always the ones the instructor defines.

Phenomenographical studies of learning and a Phenomenographical approach by a lesson designer afford a basis for curriculum design and teaching that closely resembles that of Problem Based Learning.

> In a Phenomenographical teaching approach the teacher does not single out an element of curriculum and then choose a method for presenting it, but rather looks upon it as a part of the whole, considers the interrelatedness of content and approach for the outcome of learning, sees the learning event from the learner's perspective, and integrates content and form to facilitate and at the same time evaluate the outcome. (from Booth [45, p.49])

The Phenomenographical approach defines learning as a qualitative change in a person's way of seeing, experiencing and conceptualising something in the real world.

Individuals faced with the same learning content will in fact be experiencing different processes simply from their different focuses. A students who focuses on class discussion may see the variety of ideas presented by other students and gain insight into different approaches to a concept. Another student who focuses on the teacher's notes and writing may see only one approach to the concept.

3.5.1 Phenomenography Methodology

Research in education and programming of intelligent learning systems generally use the following methodologies:

1. Trial tests and controlled tests to verify use of various learning devices.
2. Thinking protocols or interviews to develop a tutoring system (e.g. the **buggy** model and intelligent tutors [18]).
3. Interviews with students to develop an overview of the various strands or approaches in the learning experience (Booth [45]).

The present work is based on this third methodology for the following three reasons:

1. The system is very complex and interconnected so it is not feasible to test the teaching or learning effects of each part or even each agent separately (see Chapter 5)
2. The area of study is too broad and ill-defined to develop tutoring protocols such as a definitive list of correct answers or incorrect approaches
3. The group's projects each year are generally unique in some way, so computer analysis will require some classification and knowledge of the range of conceptions, rather than optimum conceptions.

The aim of the Phenomenographical section of this research is to distinguish between the deep and surface approaches to learning, within a specific domain and to use this to develop **scaffolding** to encourage students to move from a surface to a deep approach. Deep/surface experiments move away from the view that individuals bring specific competencies to learning tasks and towards the idea that the learning environment, the curriculum and the assessment inform the approach which individuals will adopt.

Phenomenographical research into the ways that students experience learning has given university teachers some insight into ways in which learning and teaching can be improved. Early studies by Biggs [35], Marton and Sǻljő [243], Entwistle [117] Laurillard [214] and Ramsden [297] plus many since, have revealed relations between the ways in which students approach their study, the quality of the learning outcome, their previous experiences of learning and the context in which the learning occurs.

The choice of conception in this thesis as the central phenomenon and **concept** used in describing knowledge produced a more subjective and relative view than is traditional in educational research. This assumes that knowledge is fundamentally a question of meaning in a social and cultural context. The knowledge that students express is described in terms of conceptions, that is the meanings and understandings of phenomena.

One characteristic feature of Phenomenographical research results is of the use of categories of description. This form of results tends to favour abstraction, reduction and condensation of the richness of the data. This is based on the assumption that the conceptions are well delimited entities, have a certain complexity, and yet can be characterised by their important similarities and differences with other conceptions. [356]

In this thesis, the categories produced in Chapter 4 are linked to the activities analysed in 6 through cross-tabulation. The patterns are then shown in a table in Chapter 7.

3.5.2 Phenomenographical Process

In the first experiments, students who are working in collocated groups, during the workshop course being studied, are observed in meetings and interviewed afterwards as to their experience and understanding of that learning. In particular the researcher is interested in how they experience learning, or their actions, and what is their experience or learning or their intentions. The interviews are very unstructured, as the researchers is seeking to understand the approach each student is taking to their learning, or how they experience the group sessions.

However the range of interest of this research is restricted by selecting the areas of learning that are of interest to the lecturers and mentors in that this is the learning we wished to support. In some cases further aspects of learning arise as students express concerns with these, and so are added to the list. However in a new learning domain the areas researched for depth of approach is naturally different.

3.5.3 Limitation of Study

Once the area of research has been decided and the data collected, it has to be encoded and sorted to develop the qualitative analysis of the content of the material. While some researchers claim that such a method of content analysis is an objective technique (e.g. [31]), Mower [256] provides some frank discussion of reliability. What we claim here is that this is a quantitative study of the range of conceptions that occur amongst students and we make no conclusion about the applicability of any particular conception to other students in other courses of study.

Also we do not claim to have isolated a unique set of categories from the interview data. However we do claim that given the same categories and quotes from the data, other researchers tend to place each quote in the same categories as we did.

The difficulty arises in this research in claiming that the categories of learning approach that are developed after analysing the interviews are in some way objective [388]. While some amount of subjectivity may be unavoidable in coding transcripts for analysing patterns, a qualitative study does not assume that objectivity and reliability have not been achieved. Instead, Rourke et al. [317] claim that the discovery of an excessive degree of subjectivity, or disagreement between coders classifying students' responses to interview, signals to the research team that further refinement is needed in category definition or encoding protocol. In this work this is achieved by ensuring that the categories are not great in number and clear distinctions between comments falling into each category can be extracted from students' interviews.

Assessment can also be made of the *correctness* or *depth of approach* of each statement of conceptualisation of learning collected from students. Such an assessment is made by someone experienced in the field, so is in effect an expert's view, but does not negate the fact that there are paradigms, and students may follow an alternative paradigm to the interviewer/researcher, a point discussed in [388]. In this work, the issue of the level of understanding, or depth of approach, of different responses is discussed with other mentors and tutors in the domain to provide evaluation.

The assessment of depth of approach is an aspect that is assumed when designing a course and developing course goals. Hence this research is restricted in its immediate application to the present domain of study. However the approach is repeatable across other domains, and the tabulation against common learning activities will still be appropriate in other domains.

3.6 Activity Theory Study of Online Learning

Activity Theory has been developed with the aim of providing a methodology for analysing the components in a system and extracting the interrelationships between these components. Also Activity Theory has been proposed as an effective lens for analysing tasks and settings (**context**) and is a framework for designing constructive software learning environments (see Jonassen and Rohrer-Murphy [186]) so is the method or process used here in this thesis to develop the software learning context.

The production of any activity involves a subject, the object, the tools used and the actions and operations that bring about the outcome as described by Nardi [262]. In the present study the subject is a group of students, the object is the report and is acted upon by the subject. This object is the intention and focus of the activity. The tools in this case are the software tools available to transform the object. The activity itself consists of a goal-directed hierarchy of actions that are used to accomplish the task.

In developing a process to design the basic system **Intertac-I**, and later to design the agents for **Intertac-II**, we used an *Ethnographic*[1] process. This involves the designer in the user community to understand it.

[1]This approach is used on Intertac-I to develop the Interface.

Some of the general issues of design processes that had to be covered in this research are:

1. changing of workplace processes to enable development of a product which can be implemented. The use of the system must be part of the original design [240]. The final software becomes part of the work process so is analysed in the same manner as the rest of the workplace activities [353]

2. need to employ methods to check for consistent requirements [235]

3. ethnographic analysis of task from one user's viewpoint can obscure the work practices of other users[2]. Design orientated towards work practices require opportunities to construct shared understanding of work, across multiple and often conflicting perspectives [353].

4. to represent human activity in software requirements the designer needs to use a coherent framework to describe the context, situation and practice of the workplace [263].

An Activity Theory model (see Figure 3.2) is used to elicit the requirements for the software in this thesis. The five step model is developed from work by Soloway et al [346] and is divided here into subsections that correlated to and expanded on an alternative model by Mwanza [261].

Context : Covers the visual interface; the overall activity on Intertac; any plan or process that the software must support and the activity relates to the student's general learning on the interface. Also includes any issues relating to the student or the community (group).

[2]for instance the user making edits and those viewing others' edits online have quite different needs from the editing tools.

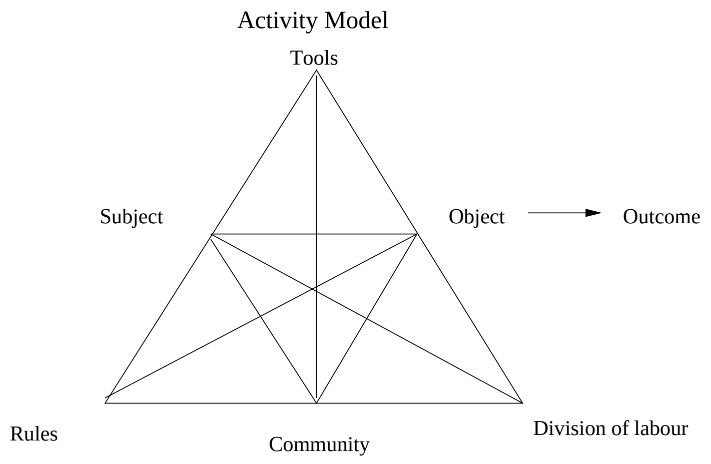


Figure 3.2: Activity Theory model used for Intertac Requirements Elicitation

Tasks : Describe the tasks the students need to do, and any variation in the order of steps they may take. Do different students tend to adopt different roles, are these related to each other in that information must be linked between roles, and are these roles supported? What architecture best supports this sort of design, for instance what is the basic Tool class to contain?

Tools : What features are needed on the tools? Will students be able to carry out the tasks on the existing tools or is further attributes needed? What do they need to see on the interface? Are there any rules the agents, or the tool itself, can be supporting in this activity?

Object : What outputs are needed, how can they be accessed and saved? What format of output is needed? What changes now this is done on software not collocated, or if new software features are added?

Rules : What non-functional requirements exist on system and any restrictions on the objects for a given context?

3.6.1 Activity Theory Methodology

The application of Activity Theory related to the development of the software interface and tool applications. The methodologies implicit in the area of HCI for interface development, as adapted from Nardi [262] are:

1. a research time-frame sufficient to elicit the student's objectives as well as actions.
2. attention to broad patterns of activity
3. the use of a varied set of data collection methods
4. a commitment to understanding things from the student's point of view

This provided the framework against which the data is analysed for requirements. Each area or category of design that is extracted from the students data is then described according to this framework and designed into the Intertac (I and II) software.

3.6.2 Activity Theory Process

Once the initial experimental studies are completed for collocated groups we develop a set of user requirements for the software and a software design is developed. The aim of the design phases are to extract the student needs and thence to translate these into the architecture, applications, tools and agents that constitute Intertac.

There are two phases in the design of Intertac. At first the researcher used data from the first experiment with groups working in collocated meetings to develop the requirements for a groupware system, called Intertac-I. These included issues of architecture, such as the need for rapid event-based communication of updates to all participants, and issues of applications or tools, such as the need for a planner and reminder system. Also there are overriding issues of design that affected both architecture and tools, such as the need to plug in new applications as tools that closely matched the preferred single-student applications of each group.

A search is made for a groupware system that provided these tools. When this is not found, the decision is made to develop the new groupware system Intertac. The desire to use Java limited the options for component reuse to two known systems, Habanero [159] and the Matchmaker server [250]. The need for complete access to the application code to enable the agents to plug in easily led to the use of Matchmaker. It is only after this system is tested that the limitation of this server became clear when not used on a Local Area Net. Also debugging messages in the logging system used in the server can not be removed when the system is updated, so a new server is written.

Intertac-I is a basic groupware tool, with a centralised server to provide synchronous communication, and various applications within the system that provided the tools needed to do the design work used in the software engineering workshops. In fact these tools are designed in a very flexible manner, so as to be usable in any learning domain as well as able to be extended as object to domain specific applications. The features that are of particular use for the diagram and document types used in this workshop are added on to the basic tools. These can be replaced with different features for different courses.

In the second experiments, when Intertac-I is used by groups working on the projects, the observations of how the system is used led to major changes in both the architecture and tools. This process of re-design continued when the next stage of development started with the design of agents. Experienced students are employed to do the coding. They brought their own experience to add to the project and greatly improved the existing system, as well as developing the scaffolding agents.

The five step model is used to describe the data collected from users in a standard format for design. One of the more difficult areas to cover but one that the lecturers considered necessary, is Decision Making in groups and this is used as an example to illustrate how activity analysis is carried out.

Decision Making

This section deals with the Activity Theory analysis of the group when they undertook decision making. The study is first make in collocated groups, then with the groups on Intertac. The design study is made with the following guidelines:

Context : What is the type of phenomena on which there is disagreement, how significance are these issues in terms of the overall design, are more than two people are involved and what sort of interchange occurs?

Interview Approach: What do students think is the main points they disagreed on, and why?

Tasks : Stages derived from collocated studies are – Exchange, aggregation, constraining, enabling, results. Are these stages distinct? Are there signals that mark a change of stage, for example, when students are adding and deleting parts of a diagram or report, when there are repeated comments referring to the concept or phenomena being debated? Are there roles taken in the debate, etc? Does the interchange in chat that occurs show any pattern for each stage? These activities will then be collated into the aspect of the software design they relate to, as well as the stage of the process they relate to.

Interview Approach: Are students aware of the stages at any point, and compare this to the group activities.

Tools : The students are using pen and paper in collocated groups. When using Intertac, files from the Intertac editors are used (documents and diagrams). Do they work together or alone then individual editing work then taken to the group? Is there any other learning that is referred to in order to reach a decision such as diagrammatic rules, etc? What is helpful in reaching the final stage?

Interview Approach: Were students aware when they had reached a decision? Does this matter?

Outcome is an diagram or file that is agreed on by the group. To ascertain this has occurred, we decided that when a diagram or file has not edited or opened for some period, it can be taken as agreed upon by the group. The outcome is changed little when using the software instead of collocated editing. The main change is 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. Another aspect of change is whether there is difference between groups that have good and poor decision making practices, in terms of their design.

Interview Approach: Groups that had long arguments can be asked about whether they think the discussion is fruitful. Compare this to the design changes that followed that discussion.

The data from this analysis is then collated for use in the design and implementation of Intertac. Where requirements are contradictory, or not feasible, the decision is based on what is considered the most significant feature to learning and priority given to those. Hence the tool itself is not always ideal, while the support is emphasised, as this is the focus of the research.

We link a design aspect to each aspect of the requirements. In the decision making activities the main requirement is for groups to be aware that there is a process that can be followed and that this improves the decision in terms of time and outcome.

3.6.3 Limitations of Activity Research

Software design also involves verifiable forms of research. Activity Theory provided the structure to categorise requirements for the system and check for completeness. There are five areas of analysis and they are processed one by one in the software development. The components of the activity describe the area of requirements gathering.

The same structure is also used to express each individual requirement. In each case, while one attribute of the model is the principle motivation for the requirement, all other attributes have to be described, as there is sometimes two or more variations of the one task or outcome, for example, that must be supported.

Through iterations from Intertac-I to Intertac-II the requirements are further refined and enhanced. By providing a basic system for students' use, the activity in this new Intertac-I environment are analysed for a new set of requirements. However the requirements related to the context in which the software is used, which is the workshop course and the students level of familiarity with software systems.

3.7 Action Research on Software Usage

The final process of Action Research involved the users in the design process. The process of Action Research, or collaborative enquiry into educational problems is developed first by Lewin [223] and further enhanced by Kolb [199]. It is a continual spiral of research cycles consisting of four major phases: planning, acting, observing and reflecting. It is characterised by iterative cycles of problem identification, systematic data collection, reflection on feedback, analysis, data-driven action taken, and, finally, problem redefinition.

During the period of online experimentation we used action research to develop the agents to support learning. To relate students' actions on the computer to their

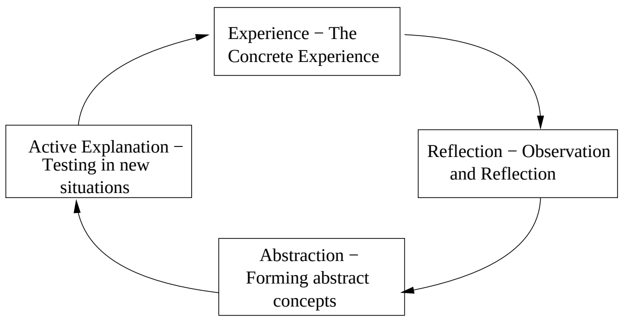

Experience − The Concrete Experience
Reflection − Observation and Reflection
Abstraction − Forming abstract concepts
Active Explanation − Testing in new situations

Figure 3.3: Kolb's Experiential Learning Cycle [199]

learning experiences required a process of extracting patterns of activities, hypothesising the intention of these, and verifying this with the students involved in learning on the Intertac system. Action Research is used in this process as it can handle the breadth and complexity of the activities being studied.

3.7.1 Action Theory Methodology

Action Research involves both action and research, and utilises a cyclic process of alternating action and reflection.

Corey [88] defined action research as the process through which practitioners study their own practices to solve their personal practical problems. In this case the research involved students proposing changes to the basic Intertac-I system to suit their needs, combined with tutors and lecturers proposing changes to the outcomes of the learning that are desired from the groups.

The role of the agents is to identify the trigger that instigates any change in the system, in an attempt to pre-empt the student needing guidance or needing a change of focus. The research involved examining the Intertac log for repeated sequences of actions, and relating these to trigger points in the learning or the interaction.

The activities that are selected for analysis in patterns is determined by the Intertac system. The activities included addition, moving or deletion of drawing objects, use of token sequences in Chat and addition or deletion of text in the report editor. Patterns or sequences of activity occurrences did not have to be consecutive, but related by close proximity (less than five interchanges between steps in the pattern).

3.7.2 Action Theory Process

When the enhanced software groupware system, Intertac-II, is developed, it is used by students in small projects similar to those used in the workshop course. As each group attempted the problems and provided their own solutions, the researcher monitored them both online and after sessions, over viewing the files they created and reading the log of the session which recorded chat, and all interactions such as additions, deletions and changes.

Intertac-I included provisions to regularly save data from group sessions. The data included logs of activity, plus the various files they are working on (including the chat interchange). The data is automatically saved at regular intervals, as versions. This data can be compared to features that might show some pattern of interaction.

There are a multitude of patterns that can be identified from stepping through the data. Since the log included an entire record of all tools and users, in time sequence, this can be analysed for different contributors, and how the use of different tools are linked. However a scope is placed on the search by being guided by the activities that are identified with collocated groups engaged in the various depths of learning activities during the first experiment. For example we are looking for patterns of interchanges involving all participants in discussion of a concept.

Once patterns had been identified, they are classified against the learning categories developed in the Phenomenological research. The areas of learning and interchange which are so categorises, are selected as important areas of learning by lecturers, mentors and the students. Hence a sequence of interchanges in Chat can be considered to show high participation of all members of the group, which is reflecting a learning approach to motivation for group work that fitted a category of either Individual Expertise or Internal Cohesion. This pattern can be enhanced by identifying a single conceptual word as repeated throughout the interchange, forming a major component of the subject matter.

Categories are selected on the basis of discussion with the group who had worked on Intertac and produced the data, in an attempt to relate what they considered is happening in their group to what the computer can detect. This verification is used to decide the categories to put patterns.

The learning categories are:

Approaches to Learning Aim to increase the depth of outcome to their learning

- **Report Writing - Document design** Conforming to a template structure which is set in the workshops.

Group and Work Processes Aim to increase the depth of learning of group processes

- **Change patterns** The patterns of insertions and deletion in files.
- **Synchronous patterns** The patterns of activity in different windows at the same time.
- **Contribution patterns** The patterns over time of individual contributions to group chat, document edits and diagram edits.

Conception of Course Constituents Aim to increase the depth of learning in the domain of learning

- **Decision Making - Discourse patterns** A topic already heavily researched, in particular in the area of conflict resolution and conflict that engenders explanation for learning. In Intertac these patterns can only be drawn from the chat line tokens for computer analysis, as no language analyser is used.

Data Flow Diagrams - Diagram design which conform to various rules which can be expressed as patterns.

Keyword patterns The use (or absence) of various technical keywords, concepts or jargon from the discipline that showed some level of immersion in the subject.

3.7.3 Limitations of Research into Agent Patterns

As mentioned above there is a very large range of patterns that can be studied and this work only focusing on a subsection. However we believe that the patterns analysed provide the entire range of the types of patterns that can be extracted from the Intertac data. A more highly sophisticated system can analyse the language of the participants or produce feedback more closely related to the language usage of the group, however that is not relevant to the present work

Also the types of groups studies is limited. They all worked together fairly effectively and produced a design for the system they are asked to analyse online. The groups are chosen from those who volunteered. Groups are formed by selecting those who may exhibit some of the independent attributes of a group that are of concern in the learning experience. These included:

Experience in the group Some groups consisted of students with equal expertise, others had a **learner** in their group.

Experience of the group Some groups are composed mostly of students who had done one or more workshop courses. Other groups had done none.

Motivation for learning Interviews before Intertac gave the research some idea of the group motivation to learn the skills and concepts of the domain.

The subjects are not informed of the focus of the research. They are told that their work is considered for how easy it is to develop a design in the Intertac environment. They are interviewed afterwards as to how the software supported or did not support their learning. The sort of problems that we are expecting are not discussed with the subjects.

Another factor that had to be filtered out in the analysis as it is beyond the scope of the present work is the familiarity of the groups with written English. This is however a major input of noise in the data.

3.8 Conclusion

This research aims to contribute to the design and implementation of scaffolding for learning on-line. Given the variety of learners who will use any system, the scaffolding has to be designed around case studies of a multiple set of users, or in this work, groups of users.

The importance in the present work resides partially in using the individual course lecturers to provide the form of the scaffolding and the concepts and their expression for use in developing the scaffolding agents. This is to ensure that the feedback is related to the course goals.

This provides some validation to the research as any need for scaffolding that is extracted from the present students' actions can be compared back to the learning goals. For example, the lecturer provides the concepts that they feel the students need to be adept at dealing with by the end of the workshop. These are followed in the group discussion and then the analysis of student use of concept can be examined by the lecturer for how deep they consider the approach. These are then used to provide feedback on various approaches to understanding a single concept in different contexts, where the concept may be used and can use previous year's projects to provide examples of uses of the concepts. Biggs has also come to the same conclusion through his approach to learning [40].

The aim of the present work is to more closely integrate the field of study and the environment in which students perform the assessment task, in this case their project. At present even collocated groups are rather thrown into the situation without scaffolding and the students have little support for learning good collaborative skills. So whatever the field of learning, the Intertac-II system aims to support this aspect.

The present case studies look at the learning approaches of students working in collocated groups on a software engineering design project through Phenomenological research (see Chapter 4), the actions of these students when working on-line in groups and develops suitable changes in the environment (Intertac) to support and change these approaches through Action Research (see Chapter 6 and Chapter 7). This work is enhance by designing the original software using an Activity Theory framework (see Chapter 5).

One reason such a detailed approach is taken to developing the learning support arose from studies of expert behaviour in many fields which have suggested that experts solve problems by applying the appropriate model or process ([19] and [113]) the importance in teaching is not so much to show a correct models or processes to students (possibly then understood out of context), but to encourage students to see the need for correct patterns of approach in each context. While experts can apply processes they know to problems, novices need to learn how to develop the processes or models and assess which process or model is appropriate in a given situation [174].

Chapter 4

Phenomenography of Learning

4.1 Introduction

This project has a long term aim to motivate and guide students towards experiences that enable the generation of the desired conceptions in their study, as well as enable their elaboration and differentiation between conceptions. This will provide the opportunity for students to enact a desired depth of activity and a range of skills in whatever domain of learning they are in.

This guidance is to be implemented in the form of knowledge-based agents in a prototype version of Interac-III. The research involved developing examples of such agents for the existing learning concerns raised above, and used this experience to develop a process for converting learning goals into agent support.

This process involves collecting the results of the Phenomenographical study of the learning activities described below to ascertain the **goal** for a particular learning approach. The Phenomenographical approach is taken based on the work of Booth [45] who categorised the various depth of approaches to domain conceptions and we use this categorisation to guide students towards a desired depth of activity in learning the domain. **Depth of approach** is a term used in the Phenomenographical study of learning, and has different significance in different domains, however here we are limited to studying the depth of activity. This chapter focuses on an educational mode of research, hence differs from later chapters in approach. However the work relates directly into the analysis in later chapters.

This chapter presents the research used to extract from the existing student groups the different conceptions of the particular course concepts of interest and the observable effect on the students' activities of different approaches to learning in the course. At any stage during the course their particular conceptions form their pre-conceived notions of concepts in the course, and are hence crucial for developing their mastery of the domain and the type of feedback which will be suitable.

4.1.1 Planning

The research is carried out through interviews with, and analysis of collocated groups. The areas of interest for this analysis are selected from lecturers and mentors interviews and open questions devised to extract students views on these topics, rather than attempt to look at all conceptions of the domain.

The areas studied in this research are firstly categorised, as by Booth, into the ways that students understand some of the phenomena of the domain. Phenomena are divided into what Booth [45] called **framework** and **technical** constituents of the course. Framework constituents are background concepts that are important to the understanding what it means and what it takes to learn the course material, but are not thematic, or expressly explained, in the instruction. Technical constituents are aspects of the course that are assumed knowledge, or explained in the course.

These phenomena formed the basis of the questions asked of students during interview:

Approaches to Learning :

1. Report Writing. This included the sections used, what is included in each section, and if the sections linked to each other. Also the style of the report, whether it fitted the context of the project.
2. Group and Work Processes. This included planning and tracking their work and how they handled different group situations, as well as what they felt about working in groups in general.

Conceptions of Course Constituents :

1. Framework Constituents:
   (a) Decision making. This included the steps or attitudes they thought important in making decisions.
   (b) Integration. This covered the integration of the DFDs with the requirements and how this is handled.
2. Technical Constituents:
   (a) Requirements. This included how they understood requirements and how they dealt with requirements that we irrelevant or difficult.
   (b) Data Flow Diagrams. This included what they through is represented in the diagrams and how they related to the overall design.
   (c) Specifications. This covered what they understood as a specification document and how they built this up.

Many of the conceptions deal with aspects of software engineering that are very much technical constituents of the software engineering courses at The University of New South Wales (UNSW), hence will not necessarily apply to other courses.

Number of groups	Number in group	Same Year/ Experience	Year	Comments
4	2	2	3	Both had one individual worker Both had worked together before
3	8	4	3 & 4	6 worked together before 2 had not

Table 4.1: Groups involved in first experiment

4.2 Collecting Data

The researcher initially developed a list of concepts and subject areas which are crucial to the course from their work as tutor/mentor on this course in the previous year. These categories are used to start the initial interviews and expanded where needed.

The first experiment is run on collocated groups who volunteered to be involved in this project. The groups analysed are shown in Table 4.1. The students are involved in a Software Engineering workshop. They are meeting regularly with their group to analyse a given set of requirements and produce their own Specification Report.

The study covered ten groups of three to four students each group, who are meeting in self selected groups. They are analysed as to how often they had worked together (most for two years), their previous academic record and general motivation interests in the course. While the amount of groups is small as we have to rely on volunteers, the amount of data is large as it is collected over one session (14 weeks).

An interview is conducted with each member when they first volunteered to go on the project and at the end of their workshop course. They are interviewed as to their learning approach and understanding of the various concepts considered particularly relevant to the domain. Then the researcher attended the group meetings when possible and simply observed their actions and how they learnt. After the meeting the group members are interviewed as a group to ascertain their impressions relating to the meeting and how it enabled learning of the categories and concepts the interviewer is drawing out of the data.

The method of obtaining data on the students' conceptions and approaches involves both observing their work in groups and interviewing them as a group after meeting and subsequent to the initial individual interviews. Some students are also interviewed individually if they are having particular problems or disagreement with the group members in the work or with their responses in interview. An example of categorisation of interview data is shown in Section 4.2.3 relating to categories of students conceptions of group work.

A grounded approach is used to extract and sort the categories of student learning in this domain from this data and compared with mentors perceptions of group approaches. The category in which quotes from student interviews are placed, is verified by discussion with mentors. However, sometimes the significance of the statements made is difficult to clarify. For example in looking at the approach to group and work processes, a statement that *I always work with my friends and do not want to work with others* is a different comment to *I find working with my friends better as I would not know what to do in another group* in that lack of knowledge is recognised. The former comment may assume that the speaker understands the need for different approach if they move outside their standard group. Hence follow-up questions are required to elicit more details in some cases.

4.2.1 Analysis

The results of this research are presented below giving the categories developed for each area researched. Also a brief description is given of the students' approaches or conceptions that fall within that category. The categories provide the basis for analysing students' actions and postulating the level at which the group is learning, so that appropriate assistance can be provided according to their present depth of activity.

Once the categorisation of each aspect of learning is derived from the data, it has been correlated to categories developed in previous work by Phenomenographers in the area of software and writing. Also the categories are evaluated by at least one of the mentors or tutors in the workshop who are asked to comment on the validity of the summaries made of each category of approaches or conceptions, and on the completeness of the categories, and on the validity of the division between categories. In particular they are asked to add any category that they felt better represents the approach or conception they are trying to achieve in their teaching.

The depth of each category is evaluated in terms of tutor/mentor expectation of a thorough grasp of the subject, as opposed to a minimal comprehension. Such evaluation are subjective but are carried out by at least one mentor and reasonable agreement with the researcher was found. It is not considered appropriate to seek evaluation from the lecturer as the teaching style is such that much of this material is handled in the groups with mentor rather than lecturer assistance.

While the depth of the categories is not established in this thesis through observing the long term learning benefits of different approaches (compared to control groups), we are fairly confident that the order of categories given in this thesis reflect increasing depth of learning involved. This is compared to the final group outcome at the end of the session, however since there are other avenues of learning during the course the on line learning is not a complete picture. Also we tended to see the groups with an apparently deep approach still held other less deep conceptions, and they often found these useful to learning at different stages.

4.2.2 Depth of Categories

As is noted during Booth's research into approaches to programming, depth of conceptions usually arose from experience of alternative views and situations where the skill or concept is applied. So in a very real sense, the context of learning affected the approach used. This is similar also to the findings on the *self-explanation effect* in the domain of learning physics [75]. If students are aware of the conditions under which specific actions or skills are used they tended to be better able to differentiate the different aspects of the skill and its applicability in practice.

Generally the comments made by students showed limited range of conceptions, and did not always match the deeper goals of the mentors. However the mentors themselves are high achieving students from the previous year, so there is an issue of delayed learning effects that occur when the students is distant from the actual learning context, and when having spent time to summarise their experience to the next year's students.

4.2.3 Details of Phenomenographical Results

This is an expanded discussion of the examples from the survey results related to one aspect of the students approaches to learning. The example chosen is students' conceptions of group work which was part of the learning domain but is a framework constituent. This concept was chosen as it is similar to other aspects of the study so forms a good example case to expand in this Appendix.

To provide an analysis of the method used in extracting the different categories of conception, a selection of student responses are given in each category with comments on how the distinction was made. Then a summary is given at the end to show how the category was solidified.

In analysing a conception, we had to look at *what* the students understood as group work rather than *how* they learnt and practised the skill of group work as they did, or their approach to learning.

Once the data was collected on the different approaches, this was used to categories the activities of students when taking a particular approach. In this way their expressed approach is linked to their actions. This meant the Phenomenographical data is used also in the next chapter to extract student needs for the interface design. For example if they wanted to know when others had finished editing, there had to be some sort of hand over step, and this would occur when they are sharing design work, or could be used to encourage the sharing of design work.

Social :

```
I only have worked with friends, it is easier... easier to get
on (approach). I find groups work best when you are already friends
(conception).

We tend to avoid disagreement.

We want to get on (approach).

I know we were supposed to have roles and things... it was just easier
to chat as friends and not get too heavy.

I would not know how to get agreement out of other people if I did not
know them (approach).

I would watch what the others did. When they had done their bit, I put
in my ideas.

I would do some changes at night then when we joined the session we
all opened our files...we did not talk about them much.

We worked when we could. I did most of the first part,
as I already knew a lot about DFDs and had some time.

If someone was interested in that aspect of the design, they did the
edits. We just watch [Encoder - found no related comments in Chat
during edits]

Question: Did you ever ask the others what they thought of your
design.
Answer: No, but some problems came up which I did not
understand so asked them what they thought.
```

```
Question: But did they agree with the changes you made in the ...
(aspect of DFD)
Answer: We all got on well and had much the same idea of what
we were building.

I had a lot of trouble following what the others were doing and
thinking
[Rest of same group] He [previous subject] was really slack... did not
do much work, so we all gave him a low contribution figure [peer
assessment of contribution to group]
```

Prescriptive :

```
I always work with my friends and do not want to work with others ..
Question: Why do you not want to work with others?
Answer: ...It is harder.

We found it harder when working with someone who did not know the
group process.
Question: What did you do then?
Answer: Well, we just did not work with them, we told the lecturer
they were no good.
[same subject]
We hope we never get someone slack again.  Question: What would you do
if this happened again.  Answer: Well, we could explain it to
them... how to work together.

I don't like working with new people as we have to tell them
everything about what we do.

We tend to get everyone to do a bit of everything, that way we can
compare notes and things...

We all want to know how each part works, as we may be examined on
that, so we all do it, ... it take a lot of meeting time arguing over
disagreements.

When she [the good drawer] was putting in things in the DFD, we tried
```

to stay focused... She did not answer us when we asked, as she was busy, but we could talk about what she was doing.

We have good meetings...
Question: Can you explain what they are like.
Answer: We all talk in turn, so everyone gets a go, and everyone puts their ideas in.

We did have one person not working much, so we just went on without him. This was not fair... but we did not tell on him.
Question: Did you try to get him involved.
Answer: He had a lot of home troubles... still he should have tried harder.

Conceptual :

When we start we have to decide how we will work, if anyone has a pet approach, or something they like doing best.

We try to all work on the whole project, so we can follow the problems that we find with fitting the bits together... If one person does the B code and the rest don't we find we do not know what they are talking about.

We did need a co-ordinator, as we had different copies or versions of all the files.

We would set a co-ordinator early, so their account held all the up to date version.

We let him [dominant student] be the co-ordinator as it saved arguments, and he had a pretty good idea of what we had to do.

[Group of four]If two people have very different ideas, we had to spend some time getting them to agree... that was hard. We didn't want to put them off as they were both good workers.

To save time we divided up the work. We spent a lot of time at the end trying to fit the design together.

Contextual :

```
[This was not found much in groups due to a lack of experience in differen
groups]

Some of us had roles as that is important for organising work, but we
were not sure of what the others were to do.

When we chose what part we wanted to do, we still had to discuss all
the parts together. That helped in the end, as we at least had some
terms in common... some of the processes and so on had the same names.
```

It is interesting to note that the logical structure, or possibly depth, changes between the different categories. This shows up in how they discuss the design. Either they discuss the whole design, or how it fits together. However this would be hard to analyse from the textual interchange online.

At the basic level, they do not even discuss each others' work in detail (chat and drawing not connected), and cannot see the relation between the parts of the design, which is something which should be taught in the process. Then at the next level they cannot relate their work to others, unless they tackle the whole project (contributions to all files from all group members). Finally at the more advanced level they are trying to find ways to co-ordinate the work (separate files edited by separate people, which leads to concept discussion only).

4.3 The Context of Learning

The students are writing specification documents for a software system that is usually selected from the web. So the product is a commercial product, that they are asked to design a solution for, using a formal approach to programming.[1] The mentors and lecturer act as the client for the purposes of the exercise.

The students do not produce an implementation, or coded version of the design. However the specification they develop is presented in B (developed by Abrial [1]), which is a language that can be animated for the student to verify that the basic functionality they wish to develop is present in the design.

The use of the language B also enables the students to be exposed to other concepts of formal programming design, such as models, refinement, decomposition, guards, proofs for consistency and modules. These are concepts they are required to implement through a design specification which they develop and write up into a report.

The report is divided into sections. Originally students are asked to produce a report with some ideas of what should be mentioned, however in some courses it is found that more guidelines had to be given. Hence the notion of a document template is developed during this research from feedback from the students. The template at present is implemented as a web resource. The template is:

1. Executive Summary.
2. Requirements which are to be divided into functional and non-functional requirements.[2]
3. Team work which involves a report on their planning and meeting logs.
4. Data Flow Diagrams (DFD's) which should present a diagrammatic design that satisfies the requirements in the second section
5. Integration of the Requirements and DFD sections which explains any difficulty in doing this, and should cover consistency between the two sections.

[1]There may be an existing commercial product based on this design which the students will be referred to as an aid in understanding the requirements.

[2]This is an attempt to have students discriminate when requirements are not part of the area of code they are dealing with in a design.

6. Code written in B, which is a formal specification language, that is developed into separate machines or programs that are included or extended into a structured design of the software.

7. Integration of the B and DFD sections that explains any difficulty achieving this.

The report is produced by a group of at least two or three (and at most four) students working together. Most groups meet rarely due to time constraints, so they tend to allocate tasks at the start, such as separate sections of the report, then do these as individual contributions to the document. As a result groups tend to spend a lot of time trying to integrate the different parts of the report at the end. It is for this reason that the **Integration** section is added after a few years of running the workshop.

Also the **Team work** section to the report is introduced as a requirement due to findings in this research that students showed little understanding of group processes and work processes, or how to develop them. Unfortunately, the success of the learning relied on a successfully functioning group, and the mentors have to be aware when this is not achieved and alert the lecturer.

The groups are provided with formative assessment[3] in the form of a mid-session interim report that is marked within a week or two of submission, and a final assessment of the final report. They also have weekly meetings with their mentors to discuss their specific approach to the design. In some versions of the workshop, seminars are given on aspects of the report, such as the rules of DFDs, etc.

[3]Feedback given on students' progress by providing assessment during the session.

4.4 Approaches to Learning

The results are organised by phenomena. The first is the area of general student approaches to learning. Within each category we view student activities in meetings, and discuss with the students what support they considered would help them carry out the required tasks or processes, or link the various concepts which we identify as crucial at each level or category. The aim is then to provide this support as scaffolding through agents.

4.4.1 Report Writing

While work on the learning domain of text writing has already been done, it is studied again for the specific context. Also as a substantial part of the group work, this analysis provides a method of analysing students' actions when using the software to write or edit text. Biggs [37] presented an analysis of writing by breaking it into these activities:

1. Intentional, including affective and aspirational aspects prior to writing, or motivation.

2. Para-writing, including planning and knowledge updating.

3. Writing, including composing, transcribing, reviewing and revising.

Since the approach of a group or individual to writing is a disparate area to analyse, we use the approaches to the writing process as described by Biggs and examine how to support these approached as they are observed in this domain.

Scaffolding for Report Writing

We interview students as to how these processes can be supported. This is a combination of requirements elicitation with an extraction of how students approach these learning activities. This is due to the need to limit any scaffolding to the sort of feedback that relates to actions that can be analysed by the computer software used.

1. Intentional motivation. In terms of motivational issues, the learning requirement is to increase interest in the work as the rate of feedback and problem answering is increased, and with decreasing hindrance to the learning caused by arranging group meetings. From analysis of meetings there is no visible way of analysing the student's present affective state, except noting a lack of activity by a student tended to result either from a lack of motivation or an inability to develop new ideas for themselves.

2. Para-writing is supported by planning tools, document layout templates and process guidelines that direct and assist students in these processes. In terms of analysing these processes students expressed the following issues:

    (a) Need to record and be reminded of deadlines and approaching milestones.

    (b) Need to incorporate templates into their documents

    (c) Need to understand the steps of each activity as they come to enact them.

3. Writing is a process that involved:

    (a) Collating information which is supported through minutes of meetings and versions being saved, to provide a complete report.

    (b) Integrating ideas where keywords or summary of ideas are needed in discussion to compare individual approaches to the project, and provide a consistent report.

    (c) Maintaining structure to the report to provide coherency.

4.4.2 Report Structure

We found the deep and shallow categories developed in previous research describing approaches to learning, differ slightly to the categories developed previously by Biggs [37] when considering learning activities and when studying a computational and highly structural domain of learning. The approach categories are expressed in terms of the report outcome of the activity at each level:

Pre-structural Not all the required parts of the report are handled and the separate sections are not tied together. Often aspects of the DFD design will contradict the requirements list. There is no sense of focus in the design which will usually deal with generic issues of client access (login), rather than the specifics of the system being designed.

Uni-structural No attempt is made to discuss alterations to the requirements or to separate them into those that fall within the scope of the work and those that do not. The DFD generally follows the requirements that do fall within the scope. The document does not address specification but rather implementation.

Multi-structural There is some attempt to present alternatives. Requirements are included that do not fit within the scope of the design, and there is discussion of decisions on the design that may have been controversial. The design is not complete in that not all functionality has been handled.

Relational The complete functionality has been handled including difficult concepts such as time or synchronicity of data. The report attempts to restrict to specification abstractions rather than implementation of the requirements. Also communication to the client[4] may be poor.

[4]in this case the hypothetical company requesting the development of this software.

Extended abstract The report acknowledges the scope of the specification domain as database only, and does not attempt to deal with aspects that are out of scope of the project. The requirements are developed into an abstract specification that can be animated but is not a detailed design for implementation. Also the document is a clear description of the system as requested by the client, with clear diagrams and design explanations to enable the client to verify the design as presented.[5]

While a detailed quantitative analysis is not carried out, due to the difficulty in analysing exactly how many times each category of learning activity is expressed in any one group, it is clear that the categories mentioned first occur more often in the reports with lower final grades, than those with higher grades, and the reverse for the later categories.

Scaffolding for Report Structure

These results provide some guidance to create an environment in which students may take different approaches to learning report writing. Students are asked how this scaffolding might be provided. For example:

Document Views To enable students to relate themes that re-occur throughout the report, such as the relation of requirements to the specification, an editor should be designed that provides the ability to describe the theme of sections of the document, and then view the document by a single theme alone.[6]

Templates In a more basic sense, the students need their view of a **Specification Document** limited by the provision of a template, to ensure that they cover the

[5]This is what is desired by the mentors, but not achieved in practice.

[6]Software for practitioners developed by the CREWS-SAVRE project [235] uses a similar approach except it also provides linguistic analysis to verify the link from requirements to specification.

necessary areas and in the correct order and detail. The template can include a brief description of their purpose in the document, to be seen under certain **views**.

Conceptions The basic conceptions in the report such as the attributes of the specification software and the need for clear communication with the client,[7] suggests that these are key conceptions to analyse in interchanges of the group. For example the need to have other groups members read over other sections can be monitored.

4.4.3 Conceptions of Group and Work Process

Categories of the conceptions of group and work process, which forms part of the learning domain in this research, formed four categories. These categories were developed for this research:

Social Group work is a required skill, and it is more fun to work with friends. These groups tend to see it as a social process. Also they already know what each person is good at, so allocate tasks automatically. Members sometimes inform each other of any decisions on how to deal with design aspects as issues arise. Groups tend to go through the work step by step, with one individual doing one section each week, according to whether they have time that week to contribute to the project. Each member of the group will have some aspect of the project they are interested in, and tasks are assigned on the basis of each individual's values.

Prescriptive These groups work with people whom they know are good at their work. Groups find out what each individual is good at and allocate the tasks

[7]This is a problem that extended from the students' poor understanding of group communication and is dealt with in the next section.

in parallel. If there is any disagreement over how to deal with a problem in the design, they tend to go with the design that seems easiest. One of the main concerns in this group work is provide a central repository for final versions of documents.

Conceptual These groups like to work with people who have done well on that work before, or who have a similar approach to the design. If not, these groups have problems with individuals trying to dominate the process. They can then usually discuss any minor differences without much problem, otherwise they tend to do parallel designs and wrangle over the final integration. These groups tend to assign the job of combining documents and diagrams into the final report, to one person.

Contextual Groups tend to allocate tasks on the basis of the interests of each of their members, as well as their strengths. This requires a discussion of the design as the first activity of the group. As the group tries to implement the design they regularly report back any changes that they see will affect other parts of the design. The whole group will then discuss the value of any design decision.[8]

The latter two approaches involve an awareness of the skill of working in a group. The last category goes beyond the experience of most students in the present work, as students have little experience with working in groups beyond their friends (i.e. the social conception) on software projects so have not the background to develop a deep approach to groups. Also the groups are very much performance oriented and the roles are mainly set by the sections of the report that has to be composed.

The groups are formed by self-selection in most cases. Friends will combine to form social groups, or people will try to work with others they think are competent

[8]This is more an ideal conception that is only vaguely suggested by group interview.

in ritual groups. Late comers and those who do not a have clear set of friends tend to be assigned to groups by the lecturer. In general male and female students will mix, although given the ratio of gender of computing students, male only groups tend to dominate.

Given the large number of Asian students and non-English speaking background students, many groups are formed of people from a single language groups, and they conduct their meetings in that language. However the software is limited to English support, which forces students to communicate and develop the report in the intended language of presentation.

Scaffolding for Group and Work Process

Much of the scaffolding required in this area should be done during group formation. At present these are self-selected groups, and the students require more support making this selection. It is possible to provide an analysis of the likely group synergy in the software through a questionnaire at the start of session. This assists the development of the group model.

The support for learning derived from this analysis is analysis of interaction on conceptions such as integration. Also participation in the design discussions is important. Finally roles can be supported through separate functionality available to different members.

4.5 Conceptions of the Course Constituents

There are, of course, a wide range of *constituents* in any course. The aim of this research is to analyse key types of constituents in the hope that all others of that type can be dealt with in a similar manner. The constituents that are studied are selected by lecturers and mentors in the workshop as aspects in which the level of student understanding is a concern.

In preparation for developing patterns of concept learning and interaction, it can be noticed that each category in the following analyses is a ‘trigger’ or precursor for the category at the next depth of approach. In that way, the order of scaffolding support is already determined, although the level at which the students developed skills may not always follow this order.

Also in each area we find that the groups obtaining higher grades, tend to present more comments in the latter categories. However, again it is not possible to provide numerical analysis of either how many time each category occurs in each group, and therefore this can not be correlated to mark.

4.5.1 Decision Making

Further work has to be done on specific aspects of effective group work, including defining a common goal, conflict resolution, setting roles and co-ordination. The first two aspects rely on group decision making. While there are working tools for decision making in industry, these are generally too structured for students to learn their own approach to this difficult problem.

The aspect of decision making that is usually focused on is group conflicts and their resolution. This notion of conflict does not include the practice of **flaming**[9] that is commonly used in public domain chat rooms. In some approaches conflict is seen as necessary for generating a design, in others as a hindrance. Students tended to describe their conception of decision making in design in one of the following ways, which are related to the previous categories developed for conceptions of group process:

Social Conflict is minimal as the group already work well together.

[9]Refers to abusive language used by people communicating on-line.

Prescriptive Problems with conflict and therefore the group is seeking a formula for solution (such as setting time limits).

Conceptual Conflict is a necessary part of developing new ideas but social mores and deadlines necessitated resolution.

Contextual Conflict is due to the difference in view between members of the group and a compromise can always be reached, if not total agreement. It is better to spend the time early resolving differences that may lead to later misunderstanding later.

Scaffolding for Decision Making

Some work has been done on the importance of conflict in learning (Suthers, Weiner, Connelly & Paolucci ([355]) and Hoppe, Gaßner, Műhlenbrock & Tewissen ([171]), Lowry & Johnson ([224])). At other times the need for helpful explanations is emphasised (see Webb [386]).

The main purpose of the research in this area is then to interpret the existence of conflicts in student interactions, and the length of answers or preferably constructive comments that students provide each other. At present this relies on students' descriptions of their own comments with tokens, such as high use of 'disagree' and other contradicting sentence starters.

The aim is to analyse the conflict stages to verify that resolution is being reached. However such an analysis of decision making stages is highly complex and beyond the scope of this work. At present the speed of communication in groupware is too slow to enable student to thoroughly convey thoughts by keyboard, so much discussion is done in short hand, and any conflict can be extended through two to three sessions. Such analysis will be left to future learner modelling.

4.5.2 Integration

This process of integration, or linking the requirements with the system design, is in part a technical constituent of the course, but there is a large amount of learning on the part of the mentors and lecturers as well as the students to relate the concept of integration to the particular group design and approach. That is integration will in fact depend on the design approach taken. While some approaches may seem 'better' to the mentor, they are trying to encourage the group to develop their own design, so they need to accommodate this while recommending any new approach to developing and integrating the design.

The conceptions of translating Requirements into a DFD design which were developed from this research were:

Post-check The DFD has to be drawn first, then we can check it for the requirements being present in a diagram

Direct-Design The requirements describe the processes and data flow at the lowest level. The design of the DFD then involves combining these into related sections at higher levels. The DFD is designed bottom up.

Pre-Design The requirements are sorted into sections that will be designed into a single DFD high level process and then each group of requirements is used as a guide in designing that DFD process. The DFD is designed top down.

Conceptual The requirements are analysed for data flow aspects and these used to develop the design of each major part of the DFD. Where requirements suggest links between higher level processes of a DFD, these links must be checked that they are represented.[10]

[10]If a requirement is not about some form of Data Flow, then it is usually a non-functional requirement.

Scaffolding for Integration

It is interesting that in the software developed by Maiden [235] the user is required to stipulate which requirements a DFD process is implementing, or related to. Thus the aim for a professional is to take at least a **pre-design** approach to this process.

To support such learning the software provides tools to link text in the report with the text contained in the Data Flow diagrams. Hence the integrated nature of the groupware is an advantage for this work.

4.5.3 Requirements

In this workshop a student is given a set of requirements that describe the system to be designed. They are what the client wants the final product to do in terms of functionality or in terms of capacity/robustness etc. The groups are asked about their conceptions of the role of requirements in the design, their relevance and how flexible they might be in altering these. Their approach related to how authentic they viewed the learning environment used in the software engineering workshop.

Social Requirements describe the system to the user in text form, so the group changes them to describe what they program. The workshops are about producing a specification document.

Prescriptive Requirements may include features that cause conflict during programming, so they have to be changed, or specific requirements selected for removal. The workshops are about producing a specification document that satisfies the requirements.

Conceptual Requirements provide a view of the user's perception of the system, which the program tries to implement, but if this is not possible, the user should be told that the requirements have to be changed. The workshop is about trying to present a coherent specification document in the domain of the requirements.

Contextual Requirements provide a framework for the design, but often there are more features that the design must implement, and these can be described through DFDs or other means. The workshop is about present a coherent document that satisfies the requirements and the user needs in the domain of the requirements.

Scaffolding for Requirements

These categories are much more inclusive than previous ones, in that the learning about requirements seemed to progress in steady steps throughout the course. Some students go further in their steps than others, and this is reflected in their final marks. However some groups do express little complexity in their understanding of the requirements, yet still provide a good report.

Much of the learning about requirements related to how students change the requirements and if they note their changes with their reasons for these. Also tracing requirements changes back into the design is a difficult task and can be supported by more sophisticated software.

As in document writing, the need to take an overview of the document and to view aspects of the document together is considered a good tool.

4.5.4 Data Flow Diagrams

Students are shown how to draw Data Flow Diagrams (DFDs) for various systems and given various 'rules' of how to draw them. There are, of course, some alternatives in the approach to DFDs and their rules, but in practice the style of the School of Computer Science and Engineering is laid out and it is expected that the design will follow this style. It is acknowledged that different workplaces will have different standards.

It is interesting to observe the resulting DFD designs. In many cases the basic rules are not adhered to in any detail. No student has stated they have previous experience designing DFDs which might cause them to follow other DFD standards so this must be related to other experience in graphical representation. In some cases students did note that it is too difficult to change drawings, so as the drawings became too complicated they tend to just accept the poor design.

Students' comments often show they are confused about the non-sequential or state representation nature of DFDs. This may be because they are relating their learning to flow chart diagrams. Yet the use of the B language as a non-sequential language reinforces this view of the modelling (see conceptions in Section 4.5.5).

Views about what DFDs represent are variable and are presented here in no particular order:

Technical Diagram of the flow of data in the software system through various filters or processes.

Requirement A diagram of the processes that are required in the system to satisfy the requirements and how they interact.

Graphical A method of expressing a state design solution diagrammatically.

Closely related are the different approaches to the activity of drawing DFDs for which we have used the same categories since the groups tended to express similar categories bout each aspect:

Technical Take the data that comes into the system and follow it through the system (can lead to confusion with use case and flow chart diagrams). Insert processes that are required to extract, combine or manipulate the data. Do the same for outputs.

Requirements Take each requirement and draw the processes associated with that functionality. Then link these by the data involved in carrying out the process.

Graphical Draw the general functions of the system, break them down, and then draw data flow links necessary for them to function in a consistent fashion, for instance if a data flow includes data from two distinct data stores, there must be data flows from both stores to a process that generates that data.

Scaffolding for Data Flow Diagrams

While DFD design is inherently a top down process, some students tend to start from the bottom up. This occurs when they work too closely from the individual requirements, which generally relate to lower level functionality. Work on a bottom up approach generally leads eventually to an overview of the system, but the use of top down design provides a clearer structure to the reports and generally gives a more consistent and coherent design.

The aim of the research will be to determine how the students develop the top levels and how these processes relate to the requirements, look for linkages to the requirements.

4.5.5 Specifications

The level of understanding of the role of specifications in program writing is often quite low. Students tend to want to implement the design they have in their heads, thus bypassing this stage. It therefore becomes hard to explain to students why the information about imperative operations should be left until the implementation step in design, after the design has been verified as following the requirements, and forming a robust system. At UNSW the specification is written in the Formal Language *B*. The fact that all specification operations are parallel in B tends to highlight this problem of producing non-imperative code. Students still attempt to write sequential procedures.

The following approaches are found from students and tutors who are interviewed. The categories developed by Booth [45] are used to compare with the programming approaches found in that work. The approaches to developing a specification in B are:

Expedient Writing those aspects of the system that can be expressed by modifying examples previously seen in B code, such as those covered in lectures.

Constructural Using known constructs to model the system as best as possible, again relying on what has been shown in lectures, but abstracting the structure rather than copying the code.

Operational Writing the processes of the DFD in code, or writing the requirements as code, hence relating the coding to the problem domain.

Structural Designing and verifying the compatibility of all the required functionality of the system, similarly to the previous approach, but further abstracting the coding from the context.

In terms of how the code is drawn together, three of the four categories observed by Booth [45], of expedient approach, constructural approach and operational approach, are observed amongst the students. The general inexperience of students with specification and formal languages, heavily mitigated against a structural approach. Tutors did tend to have this view from their longer exposure to a variety of programs specified in B.

Scaffolding for Specifications

Much of the support required for the development of learning in these categories relates to the development of concepts involved in specification and the B language. Also it involves looking over code examples, which had to be done in private if the B toolkit is to be used. Also much support can be achieved by linking the various aspects of the report, as for integration support.

Some of this type of support have been considered in previous sections and the work of the next chapters is aimed at providing a common framework for support that considers these common themes.

4.6 Reflection and Evaluation

This initial study of how groups approach learning activities covered a wide range of issues dealt with in the domain. This enabled the subsequent analysis of patterns of learning activities to consider scaffolding to for a wide range of types of activities. This does not mean the agents will provide a framework for all learning but it does mean the coverage is thorough if not complete.

The type of support that is extracted from this analysis is summarised in the following table.

Again we do not claim this is a complete list of support that enables students to move between different categories of learning activities as discussed in this chapter. It is however a fair overview of the types of process that need support and the type of support that can be given by computer.

Also it does not limit the type of analysis to the present simple tools. Each type of support can be extended by language analysis or sophisticated algorithms. Only when the agent language is specified is any limitation set on their design.

Source	Category	Aspect to Support
Report Writing		
	Intentional	Increase rate of feedback
		Increase ease of organising meetings
	Para-writing	Planning tools including course
		milestones and reminders
		Document templates for layout
		Process Guidelines listing tasks
	Writing	Provide Document Records and Collating tools
		Summaries discussion of conceptions
		including decisions and conflict
		Maintain structure of report for coherency
Group and Work Processes	Support group formation	Analyse group experience range and individual roles
	Support learning	Interaction types on conceptions
		Participation in discussions
	Roles	Separate functionality added to individuals
Decision Making	Conflict	Analyse length of discussion on concept or issue
		Analyse Interaction sequence using tokens
Requirements	Learning	Tool to link text and diagrams
	Change Management	Analyse changes linked to comments
		Tracing requirement into design for change management
DFD	Views	Able to view diagrams from different views and details
	Encourage top down	Check levels of drawing completed
		and support level expansion
	Rules	Verify DFD rules followed
Specification	Concepts	Follow discussion of concepts

Table 4.2: Scaffolding Categories

4.7 Conclusion

This chapter presented the categorisation of results from interviews and surveys of students in the software engineering workshop course. These are the main conceptions of the course that we study and that we are designing the software to support. The initial links have been made here to approaches for scaffolding and these are independent of the software context.

What we did in this chapter is to separate concepts and approaches into types, such as group (social) or individual (task) learning. As these types tended to have different descriptors to their approaches they formed separate approaches to analysing students activity. Within each type, a similar approach to agent development can be used. Also across some types there are common agent scaffolding needs.

This stage of the research has analysed the aspects of learning approaches we want to examine in a flexible learning software system, for a particular domain, which is a design course for software engineering. While we mainly look at learning approaches that are generic to many domains, we also consider specific concepts that may generalise to other domains.

However the differing depth of learning activities analysed at this stage were found through interview as well as some manual analysis of students' work while using the software. Hence we do not expect to be able to automate a strategy to distinguish all such levels or activities in a software agent system.

The next chapters continue with the development of the software, researching the activities which can be distinguished on the software, and matching them to the levels found in the Phenomenographical research. The final stage of the work will be to design agent augmentation for scaffolding for the different levels.

It should be emphasised that these categories provide a list of conceptions that can be encouraged in students, not necessarily a hierarchy. Also some conceptions such as their understanding of requirements, will effect others such as an understanding of integration. This interrelation between conceptions is not analysed in this work, but has necessitated the need to look into the process of combination of advise from different agents when activated at the same time during a learning session, and how they might be integrated.

Chapter 5

The Design of Intertac

5.1 Introduction

This Chapter describes the development of the software system through three stages. The lifecycle includes the development of the requirements from the activities of the groups, the design of a suitable architecture for Intertac-I and the implementation of further components in Intertac-II after analysing the student's activities in the first version. The following chapters use this environment to analyse the patterns and finally develop the simple agent language for the next version of Intertac.

The requirements elicitation for Intertac-I uses Activity Theory which is carried out concurrently with the Phenomenographical research of collocated students summarised in the previous chapter. Further requirements for Intertac-II are elicited during the initial stage of the on-line experiment looking at Patterns in the interactions. The requirements that are elicited are summarised in the Technical Report by Kutay [204].

This chapter aims to present some standards for the development of groupware architecture. The Architecture used is described to enable the reader to understand the system as it is implemented and evaluate the design.

This chapter shows the physical context in which the students are working when they are developing their specification online in their groups by presenting features of the final system.

5.1.1 Evaluation

To approach this design we used the following stages:

1. Planning. The software is only developed after studying students working face to face and a thorough analysis of existing software in the area of groupwork. When there is not other groupware that support document and diagram editing, we used an existing groupware server to support interaction through the interface we developed.

   Once this decision was made we had to plan the various phases of data collection and the detail which is obtained at each stage. We also had to arrange for student groups to use the software, initially funded for their time, then voluntarily.

   At each stage interviews are designed and piloted with students. These are used to elicit requirements for the software, as well as extract the categories of learning activities the groups undertake online. The interviews are designed to look at the students approach to learning and the activities they used to achieve learning. [1]

   In order to provide complete requirements for Intertac-I the future students using 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 (Winograd and

[1]When discussing requirements we refer to the user of the system as **student**. This is important as students are a specific and unique subclass of users, with highly individual needs, a great sense of their own expertise (as mentioned by Wiate [382]) and a need for very focused feedback.

Flores [392]). 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 evaluate the changes in the learning that will result from the change to distance mode.

2. Data Collecting. The interviews used in data collection are based on the model described in Section 5.2. Guidelines for the design of the software in this thesis are developed from Soloway et al. [346], who list the features of the design as: **context** for the software; **tasks** that the software will perform; **tools** required for each task; and **interface** to tools. This model template ensures completeness in the analysis of each learning activity and reduced the need to repeat research on particular learning areas.

   Besides the case studies by Nardi [264], there has been little work in developing a design model for implementing Activity Theory in software design. Therefore, a model is developed for this thesis for designing software for a learning context (see Appendix 5.2), which involved four steps. Recently, an Eight-Step-Model was hypothesised and tested for designing software using the Activity Theory model.

   The data is collected in the form of interview notes and tapes for Intertac-I and also computer files in subsequent experiments. The format of the data collected by the computer is shown in Section 6.2.1. The log format is designed to describe the physical features of the interaction that can be used for analysis. These included time the log is saved, the object positions and editing actions as well as actor or person doing the action. Log updates are saved every 15 minutes during sessions.

   The log files can also be read by Intertac to create a pictorial snap shot of the file at that time the log is saved. This enabled quicker assessment of the student's view, while at the same time maintaining the history of the interactions to date in each **Session**.

3. Analysing.

   The data collected is analysed for the design categories listed in Appendix 5.2. Each requirement extracted is described according to this model. This ensures that sufficient information is collected on each learning activity. Also it provided a format to verify if the requirement is fulfilled.

   The requirements developed are designed to enable the functionality required by all groups. However the subsequent analysis for pattern development is focused on distinguishing between groups. This will be described in the next chapter.

   Where users took a different approach to learning and use the system in a different manner, or requested different functionality, we tended to look at what is available either through programming ourselves, or open source or through existing system software such as version control on files. Hence the functionality is limited by the small scale of the project.

   Much of the functionality is centred around the two points made above, that is, assisting the construction of knowledge, as in linking information and ideas, and the flexibility in learning where student can create their own environment to work within. This often meant not including requested functionality as it conflicted with these broader goals.

4. Reflecting.

   During each stage of development, the system is developed with increased functionality then offered to students again to use. This provided an opportunity to access the effect of the changes on the group process, and in some cases design decision are reversed in the next stage. The suitability of the new system is analysed through interviews and analysing the log files for difficulties.

The entire process is repeated three times, firstly for face to face groups, then two stages of on-line groups are analysed. At each stage the depth of understanding of the groups improved and the complexity of the revisions reduced dramatically. The final version at the end of this analysis is Intertac-II which is ready to include the agent support.

5.2 Application of Model

The Requirements model used in this thesis is used as an approach to requirements extraction, so an example is given here. If the activity is **decision making**, then we obtained requirements as follows:

Context is 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 are 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 are pen and paper used in group editing, individual editing work which is taken to the group, any other learning that is referred to in order to reach a decision such as diagrammatic rules, etc.

Object of the work 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 extant logs of discussion about the phenomena, but the activity itself is changed greatly by the use of Chat for discussion instead of speech.

This was used to describe the data from users. One of the more difficult areas to cover but one that the lecturers considered necessary, was Decision Making in groups.

5.2.1 Decision Making

This section deals with the Activity Theory analysis of the group when they undertook decision making. The study was first make in collocated groups, then with the groups on Intertac. The design study was made with the following guidelines:

Context : What is the type of phenomena on which there is disagreement, how significance are these issues in terms of the overall design, are more than two people are involved and what sort of interchange occurs?

Interview: What do students think was the main points they disagreed on, and why?

Tasks : Stages are – Exchange, aggregation, constraining, enabling, results. Are these stages distinct? Are there signals that mark a change of stage? When are students adding and deleting parts of a diagram or report, when are there repeated comments referring to the concept or phenomena being debated, are there roles taken in the debate, etc? Does the interchange in chat that occurs show any pattern for each stage? These activities will then be collated into the aspect of the software design they relate to, as well as the stage of the process they relate to.

Interview: Were students aware of the stages at any point, and compare this to the group activities?

Tools : The students are using pen and paper in collocated groups. When using Intertac, files from the Intertac editors are used (documents and diagrams). Do they work together or alone then individual editing work then taken to the group. Is there any other learning that is referred to in order to reach a decision such as diagrammatic rules, etc. What is helpful in reaching the final stage.

Interview: Were students aware when they had reached a decision. Does this matter?

Object is an diagram or file that is agreed on by the group. To ascertain this has occurred, we decided that when a diagram or file has not been edited or opened for some period, it can be taken as agreed upon by the group. The outcome is changed little when using the software instead of collocated editing. The main change is 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. Another aspect of change is whether there is difference between groups that have good and poor decision making practises, in terms of their design.

Interview: Groups that had long arguments can be asked about whether they through the discussion was fruitful. Compare this to the design changes that followed.

5.2.2 Data Collation

The data from the previous analysis was then collated for use in the design and programming of Intertac. The framework used to categorise the information achieved in the previous work into a set of requirements for Intertac was:

Context : Covers the visual interface; the overall activity on Intertac; any plan or process that the software must support and the activity relates to the student's general learning on the interface.

Tasks : Describe the tasks the students need to do, and any variation in the order of steps they may take. Do different students tend to adopt different roles, are these related to each other in that information must be linked between roles, and should these roles be supported. What architecture would best support this sort of design, for instance, what is the basic Tool class to contain.

Tools : What features are needed on the tools. Will students be able to carry out the tasks on the existing tools or is further attributes needed. What do they need to see on the interface. Are there any rules the agents, or the tool itself, should be supporting in this activity

Object : What files are needed, how should they be accessed and saved. What format of output is needed. What changes now this is done on software not groupware, or if new software features are added (such as an efficient server!)

5.2.3 Design each Aspect

Once the results were collated, we had to provide a design for each aspect of the requirements. In the decision making activities the main requirement was for groups to be aware that there is a process that can be followed and that this improves the decision in terms of time and outcome.

5.2.4 Integrate Design aspects

When integrating the different design aspects for decision making we found the following problems:

1. Groups were not aware of the stages so tended not to show any progress in making the decision. The outcome seemed rather ad hoc from the text files

2. Students do not like changing others work, so that they would tend to tell others to make changes rather than doing it themselves, so the decision making was purely text based in Chat, hence hard to analyse.

3. When a design aspect came up in the chat they did not tend to use standard terminology for concepts, but would adopt new terms separate even to those used in the user requirements provide for the software project they were working on.

5.2.5 Implement design

The design was implemented through a menu in which users select the decision stage they have reached. The aim is that when each stage is finished, they will select the next stage. The stages are described when selected and related to the previous stage.

The first agent support will be to provide feedback when the group seems to be stuck on some aspect of the design (user cutting and pasting on same design product repeatedly) or when the Chat contains keywords that are part of the course concepts that have been difficult for students to grasp.

The feedback will be a question about their design or their approach to the design and suggest the menu as an option for progress towards an answer.

5.2.6 Test Design with users through computer monitoring

The design of the interface has not been tested. We expect some resistance to students using any tool that is not directly involved in "getting the product out the door" so will need to wait until sufficient agents are implemented to support each stage. At the same time we want to have agents implemented to analyse the stages as they occur (given the students providing information on the start and stop time of stages) and then consider the possibility of the agents learning the patterns of these stage.

5.3 Context - the 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 (see Figure 5.1), with a panel overlaid for scribble drawing and notes. Within the main frame various internal frames are able to be opened by the student on request. The internal frames contain the tools for viewing text or diagrammatic files. The position of these frames are selectable by the student.

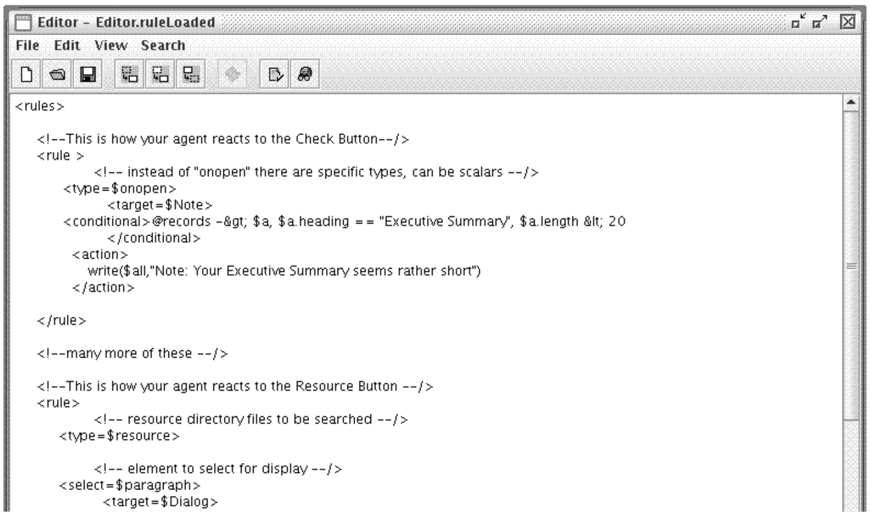
The Intertac-I interface is based on a Java Windows Class, with similarity to the Microsoft Word (TM) text editor interface (see Figure 5.2), rather than rooms (as used in Teamwave). This provides an analogy with the computer as a window onto the files being viewed and edited and matches the student's concept or model of design tools used in the Software Engineering program. 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 shows the following shared features:

1. **Awareness** Text Line included at top of screen which lists students are on-line.
2. Overlay window for rough notes and sketches that can be selected by the student as visible or not (see Figure 5.3)
3. Visible representation of the state of the Overlay On/off button (see Figure 5.3).
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 (see Figure 5.4).
5. Overlapping interfaces to the **Tools** that have been opened for group view[2] (see

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

Figure 5.1: Intertac-I Interface



```
<rules>

    <!--This is how your agent reacts to the Check Button--/>
    <rule >
            <!-- instead of "onopen" there are specific types, can be scalars --/>
        <type=$onopen>
              <target=$Note>
        <conditional>@records -&gt; $a, $a.heading == "Executive Summary", $a.length &lt; 20
              </conditional>
         <action>
            write($all,"Note: Your Executive Summary seems rather short")
         </action>

    </rule>

    <!--many more of these --/>

    <!--This is how your agent reacts to the Resource Button --/>
    <rule>
            <!-- resource directory files to be searched --/>
        <type=$resource>

            <!-- element to select for display --/>
        <select=$paragraph>
              <target=$Dialog>
```

Figure 5.2: Window for Document Editing

Figure 5.1).

6. Each student's contributions to the session to be distinguished eg. by user name or colour (see Figure 5.4).

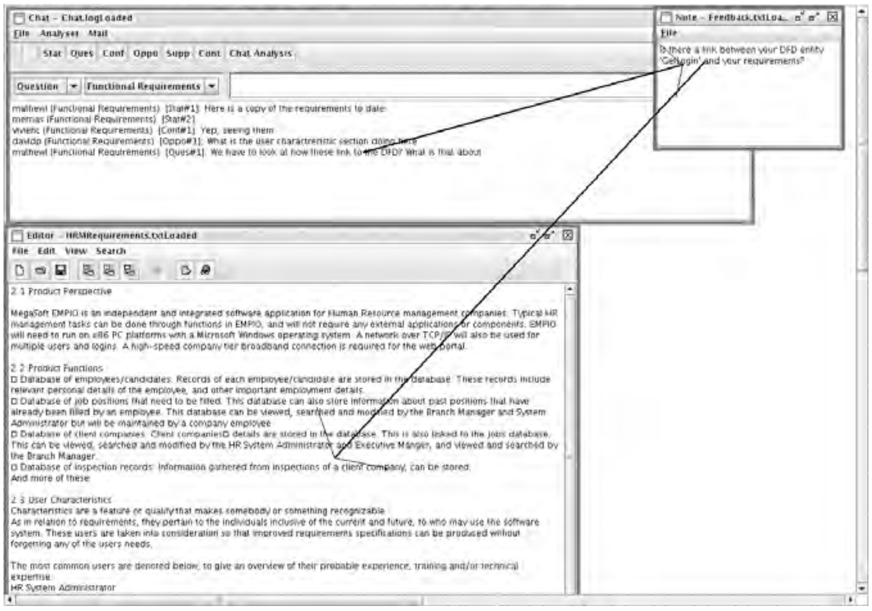
7. Select on menu to start or join a group session (see Figure 5.5).

8. Select on menu for a role such as Facilitator, Record Keeper and Librarian (see Figure 5.5)

9. Separate tools available to students who select a role to assist these different roles (see Dialog note for minute for Record Keeper in Figure 5.6).

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

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


Chat - Chat.logLoaded
File Analyser Mail
Stat Ques Conf Oppo Supp Cont Chat Analysis
Question
Functional Requirements
Note - Feedback.txtLoa...
File
Editor - HRMRequirements.txtLoaded
File Edit View Search

mathewl {Functional Requirements} [Stat#1]: Here is a copy of the requirements to date
mernas {Functional Requirements} [Stat#2]:
vivienc {Functional Requirements} [Conf#1]: Yep, seeing them
davidp {Functional Requirements} [Oppo#1]: What is the user charactreristic section doing here
mathewl {Functional Requirements} [Ques#1]: We have to look at how these link to the DFD? What is that about

Is there a link between your DFD entity 'GetLogin' and your requirements?

2.1 Product Perspective

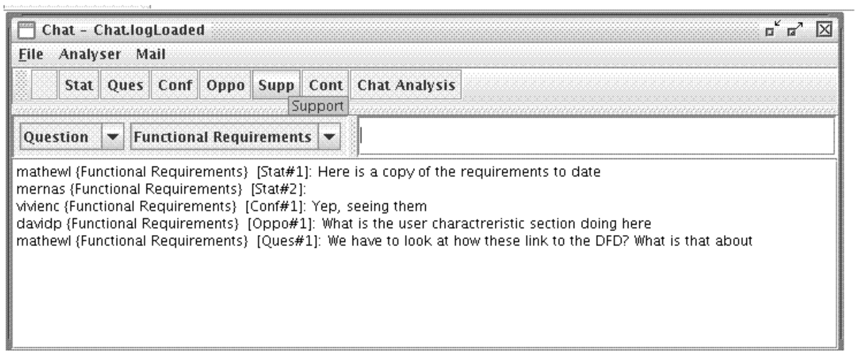
MegaSoft EMPIO is an independent and integrated software application for Human Resource management companies. Typical HR management tasks can be done through functions in EMPIO, and will not require any external applications or components. EMPIO will need to run on x86 PC platforms with a Microsoft Windows operating system. A network over TCP/IP will also be used for multiple users and logins. A high-speed company tier broadband connection is required for the web portal.

2.2 Product Functions
□ Database of employees/candidates: Records of each employee/candidate are stored in the database. These records include relevant personal details of the employee, and other important employment details.
□ Database of job positions that need to be filled. This database can also store information about past positions that have already been filled by an employee. This database can be viewed, searched and modified by the Branch Manager and System Administrator but will be maintained by a company employee
□ Database of client companies. Client companies□ details are stored in the database. This is also linked to the jobs database. This can be viewed, searched and modified by the HR System Administrator and Executive Manger, and viewed and searched by the Branch Manager.
□ Database of inspection records: Information gathered from inspections of a client company, can be stored.
And more of these

2.3 User Characteristics
Characteristics are a feature or quality that makes somebody or something recognizable
As in relation to requirements, they pertain to the individuals inclusive of the current and future, to who may use the software system. These users are taken into consideration so that improved requirements specifications can be produced without forgetting any of the users needs.

The most common users are denoted below, to give an overview of their probable experience, training and/or technical expertise
HR System Administrator

Figure 5.3: Overlay



mathewl {Functional Requirements} [Stat#1]: Here is a copy of the requirements to date
mernas {Functional Requirements} [Stat#2]:
vivienc {Functional Requirements} [Conf#1]: Yep, seeing them
davidp {Functional Requirements} [Oppo#1]: What is the user charactreristic section doing here
mathewl {Functional Requirements} [Ques#1]: We have to look at how these link to the DFD? What is that about

Figure 5.4: Chat window with user names

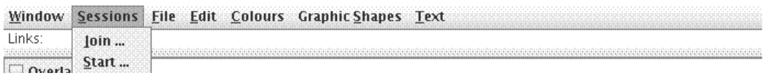


Figure 5.5: Menu for sessions

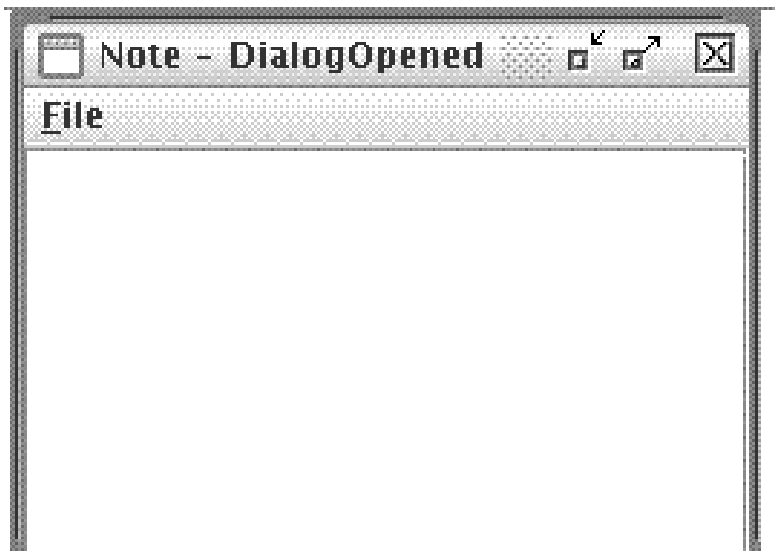


Figure 5.6: Window for Minutes recording

1. Role selected by student 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 by saving a **profile** of the students' window, 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 are based on generic models rather than being designed for a specific domain. They can then be extended with features adaptable to the domain of learning.[3]

5.3.1 Subjects

Students either work alone in what is effectively a **non-synchronous** mode or work in a group. Groups can have up to five participants at a time, although four is optimal. Delays in transmission of changes are too high for more members. The system can be improved by locking all other editors while one student enters, but this may not be suitable for group flow of interchange.

The students in collocated meetings are used to being able to tell who is present in the discussion, who made each contribution and to edit each others' work as a group. The software provides this. In using the software the students tended to manage their contributions to reduce confusion. Edits are made by one person at a time on any one document. In collocated groups students want to use different subgroups to discuss and edit different documents at one time. In software this requires separate windows to open to provide totally separate views.

[3]For instance the original concept diagram application is extended to draw Data Flow Diagrams.

5.3.2 Community

In the present learning environment there is little community support. Having material on line will provide the opportunity to improve on this. In projects groups often do not share designs as there is fear that others will gain from this and hence compete with them in terms of marks. However if all groups obtain the same support this prejudice may be removed. When previous years' work is available for students, it is often not in a useful format as the workshop each session uses a different project and the previous reports form too great a bulk of data with no method of indexing for relevance.

Computer Supported Collaborative Learning (CSCL) is seen as a means of increasing community in people's work or learning. The community provides opportunity for viewing alternative problem solutions. CSCL can enable a group's file to be saved in a format that is suitable for analysis and collection into a limited database of specific aspects of design for future sessions of workshops. The scaffolding for the learning can then be supported by a database of previous years' files.

5.4 Tasks

The tasks that students can undertake are limited by the tools that are available in the system, so the design attempts to reduce that limitation. However the slowness of the server does hamper rapid interchange. The software is also designed to monitor the tasks performed and attempt to analyse these with agent support. This is discussed in later Chapters where we look at what tasks we can analyse and how we can provide support on the basis of this analysis.

5.4.1 Steps of performing a task

The tasks for the learner in the present collocated groups are to be enabled through software. Clearly, not all tasks can be translated into the new context as for instance communication through facial expressions are lost without a video link or animated agent. Yet also new tasks can be performed that previously are not possible in groups. The most simple example is the automatic record-keeping ability of group sessions conducted on computer that enhances the performance of meetings, which can be enhanced by ability to select items from the chat log to form notes.

5.4.2 Division of Labour

The are various roles that the students are encouraged to use, such as facilitator, record keeper and librarian (of electronic files). These roles require specific tasks performed in separate **single student** windows.[4] These roles are used to support the learning of group processes, for instance documentation and summary of minutes can be made from the chat window.

Since the project is treated as a collection of separate tasks that individual group members perform, which must be integrated at the end of the session, project management becomes a part of increasing group efficiency. Hence the workshops provide an opportunity of providing advice and tools for planning that will improve the team processes.

5.4.3 Co-ordination of Labour

Any division of labour requires co-ordination of the labour towards the output or product, the final report. The group will usually divide the document into separate

[4]That is, windows that are not editable by other students, and may not be shared in the group view.

sections that must be compiled into the final report. This necessitates developing a design that is consistent across the different formats in which it is presented. Any design decision must be made by the group and recorded as reminders. Also traceability of requirements through the document from text to graphics (which is stored in textual form as XML) provides support for this work.

The software provides a window to store cut-and-pasted comments for future discussion to be managed by the facilitator as shown in Figure 5.6. Also, the record keeper uses a linked window that retains all such comments discussed in a session or earmarked for the next meeting. This provides continuity and consistency in the learning, just as minutes do in collocated groups.

As part of the support for project management in collocated groups, students use a Gantt chart planner. This can be augmented online with a template for the course structure to encourage the groups to start using the tool. Then each milestone or task can be used to send alerts to the group to check the required tasks have been completed. Also the Gantt chart has certain rules or restrictions on its use that can be used to filter the input of data and encourage better planning.

5.5 Tools

The tools in a constructivist learning environment must be designed as flexible environments with primitive objects (such as text and diagrammatic figures) which the students can manipulate. Many tools have already been developed to enable learner-centred instruction, or independent learning.

The present system is provided with a text editor, a simple notepad editor also used for feedback from the system, a drawing tool that can be unstructured, as in the overlay window, or structured into compound objects as in the Data Flow Diagram window and Gantt Chart window, and a chat window.

The tool applications themselves are originally developed at the University of New South Wales. Some have been replaced in the second version of Intertac with open source versions of the tools. The new Gantt[5] and Editor[6] tools are open-source applications that have been extended for use in this context. The original framework Architecture made the plugging-in of these external tools fairly easy, while retaining the desired functionality for the Intertac system.

Basic functionality is provided by two classes of tools being text and graphical editors. These are then refined for each individual tool by including the rules and requirements of that interface.

5.6 Object

Particular emphasis has been given to the flexibility of the environment required to enable students to construct their own learning by working on object (report) and in an environment that is unstructured where possible, as well as through the tool-based scaffolding required for support (as opposed to later additional scaffolding from agents).

5.7 Architecture

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

1. Front-end:

   Event-based messaging for maintenance of the synchronous interface.

   Updating in blocks to reduce the rate of updates.

[5]derived from Egantt at http://www.egantt.com/.
[6]derived from Ekit http://www.hexidec.com/ekit.php.

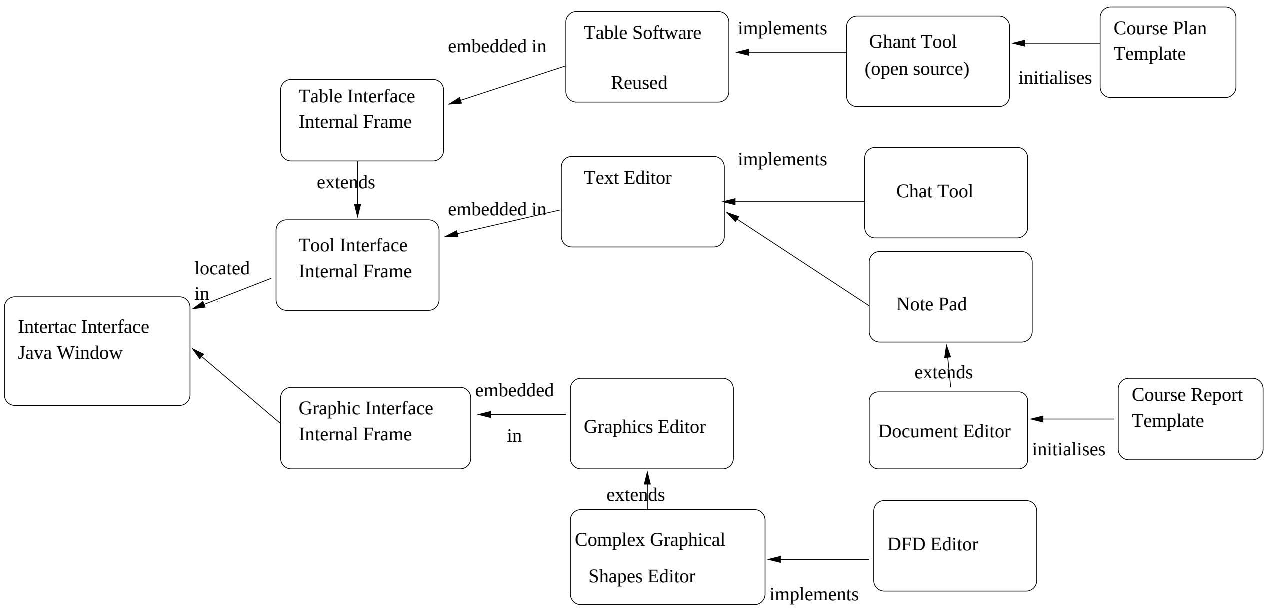


Figure 5.7: Intertac-I Component Architecture

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

Common format of tools for easy analysis of activities across tools.

2. Back-end:

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

Local repository for individual reflection and review, including Activity Log.

The original design of Intertac-I is to have the tool as the main component to be manipulated. With the need to adapt open-source applications to the system to improve the available tools, Intertac-I is redesigned to make the Java JInternalFrame class the main component (see Figure 5.7), as this is an easy adaptation of the JFrame class used as the main component in the format of many tools. The tool is then added

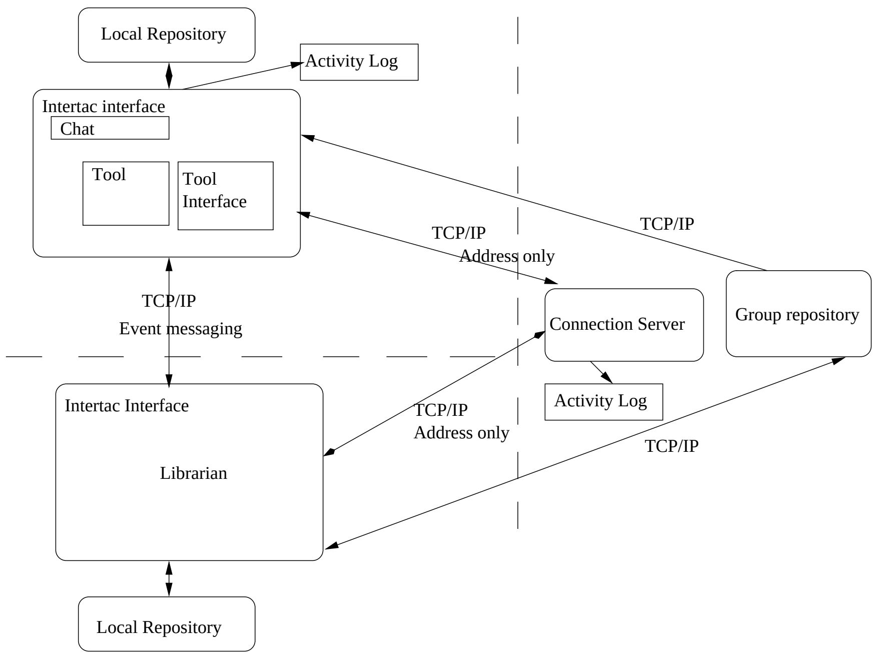


Figure 5.8: Intertac-I System Architecture

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. Synchronous sharing is through the server with an event based messaging system. Asynchronous mode is through file sharing. One of the most crucial aspects of synchronous systems is the efficiency of exchange of change events from other users.

Intertac-I used a centralised server (see Figure 5.8) 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 student can join a **session** on the server and receive a broadcast of the current screen display. Changes can be tracked between different students and different times. Also the centralised log included details about which windows are selected by each student.[7]

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

5.8 Analysing Actions

Data from the main window and application tools is stored in the Activity Log. The log can be analysed for patterns but we need to start with an assumption as to what sort of patterns we are looking for.

The research looks for patterns that of activities that relate to three aspects of learning as described by Dillenbourg and Self [100]. The aim is to find learning activity patterns which can extracted from students' actions on the computer and appropriate learning activity patterns which can be enacted by the computer in response to these activities by the students. These aspects of learning that are considered are:

Verbalising strategic decisions by computer modelling such decisions:

1. Observing actions requiring explanation and feedback, such as when additions and deletion patterns suggest conflicting actions.
2. Analysing conflict between different individual's knowledge require resolution
3. Analyse conversation patterns which suggest conflict.
4. Using the integrated tool to analyse common threads running between the tools.
5. Estimate progress of group through the project and time required to complete.
6. Analyse documents for adherence to rules of the domain, such as graphical rules or limits on length and layout of text documents.
7. Compare structures in student documents and diagrams to a template or previous years' examples, where appropriate. [8]

[8] A course planner is provided as a guideline for milestones in the project.

Acquiring a better model of learning by computer suggested approaches:

1. Observing errors in learning and delays in learning by missing activities.
2. Observing conflicting activities and suggesting an approach.
3. Observing omissions in knowledge or use of concepts and provide suggestions.

Acquiring reflective skills by:

1. Proposing a deeper approach to the domain at appropriate stages through alternative views, such as when students reach a certain depth of understanding on a concept.
2. Providing alternatives approaches to the learning to raise questions

Many of these points requires some form of detailed analysis of interactions from the Chat tool and also across all tools. The limitation of developing patterns results from the present fairly unsophisticated online systems.

While the students will make various keypad and mouse actions during their interactions with the computer (or in fact 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.

The server keeps a record of every action of the group members, including deletions. This is to be replaced by a Group Modelling Agent in Intertac-III. This data is used to develop the learning activity primitives. The server includes information on:

Source Includes the student who is subject of the action, the time of action, the object changed, the change made to object in tool (add, delete, move) or file (save, new).

Proximity For example, start and end of an arrow, or position of inserted text and any links to other objects eg. object in tool (DFD links) or file (sub-level DFD).

Conceptual Occurrence of important concepts in the course.

The patterns derived in this thesis are designed to be combined through the pattern language to derive more complex agent analysis relating to a sequence of actions. From these individual actions an agent can analyse:

Duration How long did each group of actions take to form?

Opposition Students deleting and replacing objects with different or similar structures.

Completion Activity patterns that are completed.

Absence Absence of an expected part of an action.

5.9 Verification Testing

As in the development of any software for general use, it is important that it be used by people other 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 can be improved.

Once the Computer Supported Co-operative Work (CSCW) software 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. Some work has been done in this area by researchers such as Rein and Ellis [302]. However this provides a linked focus on the ability of groups to carry out activities using CSCW and what tools can be developed to maximise benefit of the environment.

The present work has a broader focus, that of developing a process elicited students' needs in terms of requirements for support for learning activities, and then translating them into software agents. Field studies are conducted using the software, without controls. The students' activities, learning processes and requests for changes are recorded. From this we developed a design for agent support as described in the following chapters.

5.10 Conclusion

The research has provided a framework for developing groupware for use in distance learning in a design course for software engineering. It is believed that many of the features of the generic interface are suitable for other domains, and the ability to plug-in open source tools increases the suitability. However, it is a fairly bare bones interface, suitable to computer scientists and engineers, but not so suitable in the humanities.

The next chapters continue with the development of the software, researching the design of agent augmentation for learning support. While we have provided a basic design in this chapter, rather than enhance the design itself, we aim to provide another form of learning support. By connecting agents to the system we provide immediate reaction to learning activities that analyses and enhances them.

We are assuming the students provide the intelligent designer and the system can only analyse according to information it holds, either in historical project files or rule files. The agents will work in such a learning environment as long as they provide scaffolding that students can use or respond to, rather than attempt to do the job of constructing the report themselves.

Chapter 6

Pattern Description

6.1 Introduction

This chapter describes the different **actions** derived from the activity of students' learning on the Intertac-II system and relates then to the different levels of approach derived earlier. The patterns are then described as Executable Semantic networks in the following chapter. Here we explain how the patterns are derived from the data and the existence and completeness of the patterns are justified within this learning domain and software environment.

The patterns describe the manner in which students execute their learning activities in the environment, and are a study of approaches to learning activity, as an extension of the Phenomenographical studies of approaches to learning in the abstract.

6.1.1 Planning the Conceptual Framework

The extraction of learning activity patterns relied on Action Research approach. The aim is to extract and categorise the actions used by students in their learning. The conceptions and approaches which had been analysed in the first experiment formed the basis of this research and the results have been cross tabulated in Section 6.5.

The patterns derived are to be used for implementing agent support, and the pattern must be derived from computer observable actions, rather than the answers to specific questions to students (as is done in Intelligent Tutoring Systems). The pattern expression is limited by the analytical tools, however the structure still encompasses future agents using improved analysis by computers.

The other dimension to this research is to establish if different depth of learning activities can be extracted as patterns or part of the pattern conditionals, that is, if different depths can be observed, then the pattern will implement different response to that process.

The objects and their attributes which can be observed on the present software on the computer (including the log files) are:

Words :

1. Concepts of the domain,
2. Tokens used in chat tool, or
3. Flaming words

which can be analysed for the sequence of actions.

Actions

1. Addition or deletion of a user's own or others' contributions, or
2. Opening, saving or closing of files

which can be analysed for the interaction between users.

Interactions :

1. Within one tool or within all tools, and

2. Within one user's actions or between users

which can be analysed for attributes such as length, location and timing.

Timing :

1. Schedule of document production,
2. Occurrence of conceptual words in content of chat log, or
3. Delay in contributions such as response

The timing may be compared to a course schedule that gives estimates of expected timing or timing of lecture structure.

Structures in student documents and diagrams compared to templates where appropriate.

Rule adherence where rules existing in the learning domain[1] and can be expressed in terms of proximity, direction, repetition or commonality with archival examples. Rules can also be developed in terms of expected actions or interaction sequences that may be incomplete and hence the next step can be recommended.

As part of this research approach, it is also assumed that the aim of any pattern analysis includes isolating the varying levels of development in each aspect of learning and providing feedback that emulates the next level or scaffolding that supports the progression to the next level, when students have achieved a prior or alternative level of learning activity.

[1]such as graphical rules or limits on length and layout of text documents.

6.2 Collecting Data

Data is collected on students working on the software on a group project. This second experiment is run on students working in groups of three or four on the software Intertac. Students are asked to volunteer and are selected as a group, or as individuals who are assigned to a group, on the basis of providing a variety of cases for study, as discussed in Section 3.3.2.

There are 20 groups run in three sessions of 7, 7 and 6 groups each. The groups are listed in Table 6.2 below. Each session ran for about 3 weeks, during which time the students meet when they can, on-line, and achieved as much of the project in the time they had available. Most of the data is collected from the later two sessions, as the earlier groups had problems with the server and tool that had to be reworked according to the specifications developed during the early work (see Chapter 5).

The students actions are recorded as log files of the commands transmitted by the server, and as regular version of the files they are working on. An example is shown in Section 6.2.1. Time information on the opening and closing of tools and transmission of changes to the tool contents are included in the server logs but only used for clarification. This data is used to develop a dynamic picture of events mostly at a descriptive rather than quantitative level.

In this experiment, student groups are asked to engage in one of two different specification projects (one easy, the other more complex). Students with little software engineering experience are given the easier project of having to develop diagrams of a recipe for an Asian meal, a project which is used in the course to assist students to attempt DFD's and obtain feedback on their use of the diagram format. Other students with more software engineering experience are given a more complex task of having to produce a specification document for a proposed software system described on a commercial web site.

6.2.1 Log file

The data is stored in the Activity log as:

```
"No";"Type";"Label";"Class";"Method";"User";"Exception" 1;ADD
;1;MmServerSession.first;writeObject;"Object "//3" written with value:
[ckutay|black]"; 2;ADD ;1;MmServerSession.first;writeObject;"Object
"//11" written with value:
+|false|10|java.awt.Color[r=0,g=0,b=0]
|java.awt.Font[family=monospaced,name=Courier,style=plain,size=10]|
|117|12|191|139|21;|false|1|java.awt.Color[r=0,g=0,b=0]
|java.awt.Font[family=monospaced,name=Courier,style=plain,size=10]
|1|150|53|157|42|21;|false|1|java.awt.Color[r=0,g=0,b=0]
|java.awt.Font[family=monospaced,name=Courier,style=plain,size=10]|
|153|64|153|53|21;|false|1|java.awt.Color[r=0,g=0,b=0]
|java.awt.Font[family=monospaced,name=Courier,style=plain,size=10]|
|153|75|153|64|21;|false|1|java.awt.Color[r=0,g=0,b=0]
|java.awt.Font[family=monospaced,name=Courier,style=plain,size=10]|
|153|86|153|75|21;|false|1|java.awt.Color[r=0,g=0,b=0]
|java.awt.Font[family=monospaced,name=Courier,style=plain,size=10]
|All_Group|122|97|185|86|21";
```

6.3 Analysis

The data is analysed to extract patterns of learning activity. The aim is both to develop a suitable pattern structure for the definition, and also to isolate all patterns possible in this system. While this is only one system, the generic nature of the components and the component based architecture of the tools suggests that such patterns are generic to groupware systems. Albeit more sophisticated system enables more complex analyses. However much of the sophistication of groupware tools lies in how the student use the tools (for instance if students use the tokens in chat a much greater analysis of chat patterns is enabled) rather than the tools themselves.

6.3.1 Evaluation of the Theory

The patterns of student activity derived from this research require evaluation in terms of the method used to extract them from the large amount of data provided by the student's files. This is achieved in part by comparing the results to previous research in the area of group interactions in Chat and the few articles which document research into other learning aspects, such as contribution to the group. It is found that all patterns or cliches which have been previously isolated, are also able to be isolated in this work, with similar analysis of the appropriate response. This is discussed in Section 8.1 where we discuss the implementation of the patterns and look at examples which match previous research.

While there is no opportunity to compare expert advice to the computer responses which would occur from the implementation of the patterns as agent, the pattern implementation is developed in consultation with project students who are employed to develop the agents. These students had in the main completed the workshops courses themselves, and are asked to consider what they would prefer as support during their learning. The patterns as presented for them are found to be clear for them to understand (if not implemented) and they are in agreement with the analysis and response to the learners incorporated in the patterns.

Finally the patterns form a complete set within the domain researched as they are extracted using the basic activities on the computer, hence any other patterns must be derivable from combinations of these patterns.

6.4 Research Approach

In order to analyse patterns of interaction in the log of the groupware activity, a methodology of encoding speech acts and other activities has to be established first. To do this we decided what sort of patterns we can look for and hence what actions to analyse. The types of patterns selected are patterns of increasing depth of activity, of group processes, of efficiency in **document** production and efficiency of learning.

In Section 5.8 we described the information that the tools provide the agents. This forms the data which we collected, together with the text documents and diagrams which formed the product of the group work. Also the chat logs provided information on the group which can be interpreted by human analysis to support the analysis of what can be interpreted by the computer from the students' interactions.

6.4.1 Review of Prior Experiments

Work has been done previously on students' patterns or processes in groupware. Most previous studies have been focused on one type of interaction, such as knowledge sharing in Chat analysed by Soller and Lesgold ([345]) and participation using problem solution differences in prepared designs in work by Constantino-González et al. ([87]). These previous studies have not developed a pattern structure or language. In this section we describe the sort of patterns that have been studied before, as a basis for the more complete description of patterns developed in this chapter.

While Intertac does not use the same attributes for analysis, a similar approach to Barros and Verdajo [28] is used to construct the Executable Network designs for Chat interactions from the experimental analysis. In the networks, there are sequences which, when traversed, provide certain assumptions about the group progress, or

Reference	Focus	Rel. [2]	Aspect Queried
Soller and) Lesgold ([345]	Knowledge Sharing	Yes	Tokens
Soller and Lesgold[342]	Active Learning versus Encouragement	Yes	Dialogue Analysis
Constantino-González et al.([87])	Participation	Yes	Uses prior designs by individuals
	Contribution	No	
Boud[50]	Argumentation	Yes	Chat only
McManus and Aiken [231]	Problem Solving in Chat	Yes	Chat only
Berzeny [33]	Dialogue skills sensitive to relationships	Yes	
Veerman, Andriessen and Kanselaar [373]	Structure and focus content of debate to task goals	Yes	
Shaw et al [334]	Giving information is facilitation	No	Dialogue Analysis
	Asking for information is Performance	Yes	
Webb [387]	Questioning requires Immediate feedback for learning	Yes	
Barros and Verdajo [28]	Sentence Openers related to group attribute	Yes	Conversation Graph created plotting flow through nodes relating to attributes

Table 6.1: Previous research in area

suggest which certain advice can be given for the next stage. These sequences form special cases of the generic Chat interaction patterns.

6.4.2 Process of Analysis

The students worked on their project for various periods alone or in **sessions**, at both times using Intertac-I. Where possible, given time constraints, the researcher joins the group on-line to monitor conversation live. This enables the researcher to gather a 'feel' for the interactions and ask immediate questions to verify the impression obtained of the groups focus, process or present stage.

For instance the researcher notes frequent editing, and highlights that to inspect the logs for this pattern. Or the researcher finds at times the conversation seems inflammatory when the group replies that they are just used to talking that way to friends. Given the large amount of data collected, this extra data provided a means of sifting out the patterns significant to those working in the group, as opposed to the patterns that are observable in the final design, for example.

The files collected from the students' sessions are encoded for manifest patterns in the objects, based on the criteria for collection as discussed in the previous chapter. The encoded sessions are then reviewed with the aim of understanding these patterns of interactions and then the students are interviewed to relate this to their perceptions of their learning experience during that session.

The results of this data analysis are then compared to latent variables found in group interviews that covered their experience during the session. For example, apparent patterns of poor communication can be compared with students' views of how the discussion went. After investigating such issues we went back and compared these answers with the data once more to assess whether the group's perception of their experiences correlated with any patterns in the data.

6.5 Results

Given this brief sketch of the data available, we then looked at the pattern areas which we wish to study:

Depth of Activity This is linked to the levels of approach introduced in Chapter 4 (relating to report writing, group and work processes) and conceptions (relating to group and work process as learning, decision making, linking document sections, requirements, DFDs and specifications). (Section 6.7)

Efficiency of Document Production This includes templates of the document structure, which can be provided, and sequences of insertion and deletion that signify design decision problems. (Section 6.8)

Efficiency of learning This includes all the resources that can be made available to students to improve their learning in the domain. In particular, if files from previous years can be searched for specific features as alternative examples for students, what sort of search patterns can be used? (Section 6.9)

Given the broad scope of interactions that fall under these headings, it is necessary to restrict the review to particular aspects that concerned the lecturers in these courses. Also the detailed discussion of the patterns of activities under group and work processes (such as roles, sub groups, etc) is not included as they were found to be specific examples of the generic Interaction Patterns and hence this would be repetitive. They are included in the patterns tables in Chapter 7. A discussion of the results of tracking and analysing these interactions and learning approaches is summarised below and organised in terms of the listed aspects.

Within each aspect the categories of activity patterns are extracted that appeared to reflect different activities for learning. These activities are named according to the most similar category in the approaches to learning that aspect extracted from interviews in Chapter 4. The similarity is derived comparing the category of approach used by the group during the interviews during the experiment, and matching this to the most commonly occurring activity pattern for groups in that category.

Tables are presented in each section showing the most apparent correlation between expressed views of the approach to learning from Chapter 4, group range of experience (low, medium, high or mixed) and the patterns of activity taken through the software interface. In many aspects the mixed group show the most variation across category. However this may arise from the difficulty in extracting a coherent group view on either approaches to learning or a coherent pattern of activity by the whole group. In the mixed experience group some members are frequently more dominant than others in collocated meetings. Often different people would dominate in the computer communication forum. Hence if the group view is developed from the dominant members, this will change between interviews and computer interactions analysis. The mixed group showed more variation within the group due to many factors including the lack of familiarity of the group members with each other, and the lack of confidence at times by those less experienced.

The type of groups involved in the study are detailed in Table 6.2 below. This shows that the range of experience of mixed groups was great, in that some contained second years, some only third and fourth years. The details of the effect of different variations across groups is not considered in this work.

6.6 Computer Analysis of Learner

At this stage it is worth looking at what it is possible to physically analyse from computer data. Beside the various activities that students can carry out, the important aspect for analysis is the relation between these activities. These attributes were described above and are summarised here:

Group Number	Number in group	Experience	Year	Outcome Design
Gp 1:				
1	4	Mixed	3/ Mature Age	Poor
2	4	High	4	Very Good
3	5	2	Medium/High	Medium
4	5	3	Medium/Low	Poor
5	4	2	Low	Good
6	5	3	Mixed	Good
7	4	2	Medium	Good
8	4	2	Low	Good
Gp 2:				
1	4	1x2, 3x4	Mixed	Very Good
2	4	1x2, 3x3	Mixed	Good
3	4	2x2, 2x3	Mixed	Very Good
4	4	4	Mixed	Very Good
5	4	1x2, 3x3	Mixed	Good
6	4	1	Mixed	Poor
Gp 3:				
1	4	1x4, 3x3	Medium	Good
2	3	3	Medium	Poor
3	3	3	Medium	Good
4	4	2x2, 2x3	Mixed	Good
5	4	2x4, 2x Mature	High	Poor
6	4	3	High	Very Good
7	4	3	High	Good

Table 6.2: Groups in second experiments

1. Sequence of activities
2. Scheduling in relation to course progress
3. Length of contributions
4. Frequency of contributions from each user
5. Conflict in contributions such as chat arguments (which may be productive) or editing other's contribution in documents and diagrams (which may be counter-productive)
6. Similarity between example designs or templates and the student's design or document.

Combinations of these may form good learning, deep learning, or poor learning. The challenge is to isolate as much as possible from the data. We have some idea of how the groups are learning from interviews and watching their progress, so we want to see how much of this can be extracted automatically from the computer data.

In the following discussion of the activities that we observed of users online we include the name of the Patterns that were extracted to describe the type of analysis that was performed to distinguish different learning approaches, or different learning, group process or document writing problems. The Patterns are described in the following chapter, but this chapter provides a link between each pattern and the analysis that derived it.

6.7 Depth of Activity

This section looks at patterns that correspond to the levels introduced in Chapter 4. Only three categories are analysed, these are one of the approaches to learning (group and work processes) and two of the domain conceptions (linking document sections and requirements).

This section provides an analysis of how groups can be distinguished as to their depth of activity which is then related back to the depth of their approach, or the depth of the outcome. Some aspects of these differences rely on how the students use the tools of Intertac, so they may not be able to be analysed in all groups. For instance if students use Chat Tokens the meaning of their chat interaction is clear, if they develop their own project schedule we can compare progress to this timeline.

One feature of groups working on the projects is the changing focus of discussion and of their work. Even when dealing with a small design, students often talk about other issues, especially as they wait for a member to do the editing. When they do have difficulty with their design they will discuss many issues in quick succession, not fully resolving them. Sometimes they will link these changing foci and build their knowledge progressively, while other groups just seems to search around, without direction. Within each project, there can be a limited number of foci for the design, and by analysing what the students did in terms of these foci there emerged a picture of the patterns for approaching design. For instance the design may have a login, a reporting system, various types of records and related functionality and a related database structure.

6.7.1 Group and Work Processes as Learning

When working in a group, students can take different approaches to both the learning of the process, learning through the process, and enacting the process. We suggest that these are linked, so that the process that can be observed by computer is a manifestation of the approach. We describe here the processes and pattern types observed, in relation to each conception developed in the earlier research in Chapter 4.

Expedient Processes Tasks are allocated in the first session. Discussion within the group is limited, as each student works alone and does not compare with other members to share ideas. There is little conflict or decision making over design issues. There is never much conflict, even near the end in trying to collate the design. Not much problem with deadlines, as whatever is done suffices.

This is analysed by the interaction sequence on chat and the amount of editing on-line. This is handled by the Chat Analysis Pattern.

Constructural Processes These are similar to expedient processes in that tasks are assigned early, but there is some discussion of how and why the work is divided as it is. In addition, some conflict will arise over design, but is quickly solved. The word *easier* is used often in such discussions. Some cross-editing of each others' documents. Often delays at the end trying to collate the design. There are often problems with deadlines.

This is analysed by use of the Topic in Chat "Task Allocation" or scheduling of such tasks using Chat Analysis. Also it can be analysed by observing the split in files or sections of files handled/edited by each person using Document Template and Document Editing. Also the Decision Making Pattern can analyse the decision process near submission milestones using Planning Schedule.

Operational Processes There is more discussion of the actual problem and a little task allocation at the start of the work. Conflict is more drawn out in discussion, and more concepts introduced. Overall there is little argumentation, more explanation, and fewer problems with deadlines than the previous category. A good deal of cross-editing of each others' documents is in evidence.

This is analysed through Chat Analysis, and Planning Schedule. Also the Explain Change Pattern can analyse the length of time spent in discussion and Document Editing linked to different users.

Expressed View	Experience	Activity
Social	low, medium and mixed	Expedient
Ritual	low and mixed	Expedient
Ritual	mixed, medium and high	Constructural
Political	low	Constructural
Political	mixed, medium and high	Operational
Political	high(1)	approaching Structural

Table 6.3: Relation between Approach expressed in Interview and On-line

Structural Processes [3] Discussion of the design as a problem at the start, with little task allocation during the sessions. There is an appearance of things happening without the effort of co-ordination in that roles are clearly defined. Discussions of conflict in design do not involve argumentation. Rather questions are made and explanations given as to what the design signifies. Sometimes design sections will be redone by other members of the group without conflict.

Use of User Roles and less Sub-group patterns found. Learning analysis through Chat Analysis patterns show efficient learning. Also cross editing analysed with Document Editing but not showing associated conflict in Chat Analysis

Many of these developing skills of the group are already a part of the design of Intertac-I: planning and allocating tasks; providing explanations either to groups or individuals; ability to edit or redo others' documents, etc. Where increased use of these processes is considered important, further analysis is undertaken in later sections to provide motivation and guidance.

Table 6.3 shows the correlation between expressed approach and patterns of activities described above, as varies for different group experiences.

The groups found that even if they had coherent views of the learning domain of group processes, the lack of experience in the domain of learning can affect the

[3]This is an ideal conception that is approached by only one group, but can be encouraged as a progression on the previous developments.

approach to groups as they are enacted in their online activity. Hence their understanding is ahead of their activity. While providing support for learning may be productive, there is some need to ensure that groups are initially formed in a manner that enables them to function well and have related experience to avoid this situation. The types of patterns abstracted from this data are described in the Section 7.7.1 in the following Chapter.

6.7.2 Document Integration

These conceptions formed part of the learning part of the process, but use slightly different categories. When dealing with the conceptions, rather than the approaches, the enactment is not as closely linked to the conceptions, and hence harder to analyse. Each of the patterns explained in the following chapter relate directly to each of the sections discussed here.

Post-check Separate sections of the design are rarely in **view** at once. Only when discussion is complete on a section will it be compared to previous design sections.

Direct-Design The design of the Data Flow Diagrams is based directly on the requirements, and the enactment of each requirement is used to create separate sections of the diagram that are then linked together. This requires a lot of editing of diagrams, usually by their originator. Often this results in excessive inputs and outputs to processes in the DFD designs.

Pre-Design An outline of the DFD is developed first and then details are filled in. This may be done by other members of the group. There is more editing of the lower-levels diagrams, but still some editing of the contents of processes at the higher level. Requirements are still referred to more later in the work.

Expressed View	Experience	Activity
Post-check	low and mixed	Post-check
Direct-Design	low and mixed	Post-check
Direct-Design	low, mixed, medium and high	Direct-Design
Pre-Design	mixed	Direct-Design
Direct-Design	mixed	Pre-Design
Pre-Design	medium, mixed and high	Pre-Design
Conceptual	medium and high	Conceptual

Table 6.4: Relation between Interview and On-line approach to integration

Conceptual The requirements are reordered into sections first, then the DFDs are started. If the sections are already ordered into groups in the project, the group still discusses this. The design makes links between wording of sections in the requirements and DFD top level processes from the start.

It is possible to analyse what **view** of the document the student selects; who is the original author of a file and then if someone else edits it; and commonality of work between requirements and DFDs. However, by themselves these are not enough to distinguish between levels of conception. Many agents will have to be linked to form a **multi-agent system** that develops a plan of the group's conceptions. Alternatively, the separate types of actions related to different levels of this concept can be analysed by single agents and seen as equally valuable in encouraging the students to develop the range of skills to deal with different design problems.

The results suggest that the activity patterns extracted do relate to the approaches to integration that are expressed by the students. While the level of experience of the group will effect this relation, it can be expected that students with lower or mixed levels of experience with DFD will not be able to perform to the level of their understanding of the design process in general. While they can express the approach they feel should be taken, actually carrying out the process online may be difficult due to their lack of expertise.

Mixed groups caused the most cross over in the above table. This is possibly as one view tended to dominate the group online, but not necessarily the interviews. It is interesting to note that those who are verbally adepts at expressing their views, may not have been able to express them online.

6.7.3 Requirements

The conception of requirement is visible through the focus of changing requirements; deleting requirements; talking about changes; related changes to DFDs; informing the client[4] of changes; and relating the requirements to the DFDs. These are linked by the following actions, in order of depth of approach.

Null Pattern Few changes to the requirements, and the changes that occur are not linked to discussion of the topic of the requirements.

Explain Change Requirements Pattern Changes to requirements are temporally linked to the design of that section of the DFD or B specification. Usually requirements are removed. The group often forgets to include reasons for this change for the client. Also changes can be made late in the workshop course.

Template Course Plan Pattern Changes are made during the design phase,[5] and linked to an explanation in the document. Changes tend to be less than in previous categories.

Context Design Pattern Changes to the requirements start in the first session and include some additions as well as deletions. There may not be a strong relation between the wording of the requirements and some of the processes and data of the DFD, as extra features are added.

[4]The projects involve developing software for a realistic client, based on the requirements the client has provided.

[5]phase of course can only be identified by course plan template

Expressed View	Activity
Social	Null Pattern and Explain Change
Prescriptive	Explain Change
Conceptual	Template Pattern
Contextual	Context design

Table 6.5: Relation between Interview and On-Line approaches to editing

The processes which can be implemented by agents to support deeper conceptions include the removal of requirements could require some explanation in the document relating to this removal, or change, using Explain Change Requirements. In addition, when requirements are discussed in workshop seminars as proposed on a course planner template these could be tracked in the student work.

As applicable to all the patterns dealing with the level of approach or conception by the group, any sign of variation in depth of approach must be analysed using more than one basic pattern by collating the multi-agent analysis in a Group Model proposed in Kutay and Ho [207].

6.7.4 Cognitive Context

The main difficulty in tracking the conceptual development of students is the lack of any accurate way to connect their discussion and actions with each concept of the domain. The approach used is similar to that used in CHAOS [337], which provides a Lexicon of terms or the jargon from the domain of learning. The lexicon is used here to support conversational grounding by comparing the history of a student's contributions with the lexicon. The same lexicon can be used as a Data Dictionary for the Data Flow Diagram development.

From the analysis above, it is proposed that by providing a summary lexicon as topics of a student's contributions to Chat,[6] students are encouraged to focus their

[6]These keywords linked to Chat contributions are called *Topics*.

conversation, or *ground it* in the course. The students can also then 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, decisions made, and so on?

This can be matched to the expected level of engagement with that concept in the course, at each stage of the workshop, using Topic Chat Analysis pattern and Template Course Plan. For instance, the software expects more engagement when a particular topic is introduced into seminars. In particular the Chat files can be searched to verify if the topics are being taken up by the groups.

Feedback relating to the concepts and the context of the learning can be placed in the chat window to form part of the discussion. The human mentor went online during discussions by groups during the experiments and used this avenue with some good results in terms of assisting the student's discussion through feedback related to a simple analysis of the token sequence and use of key domain words. The difficulty expressed by students is in needing the interjections pitched at the correct level of understanding, or at the correct time in their work. This can be assisted by the use of a course plan that provides a time line for the expected development of various concepts.

6.8 Efficiency of Document Production

Many documents or diagrams produced in the workshops have a set structure or rules of format. While some of this can be presented in templates, sometimes it is useful for students to have their work analysed according to these rules automatically, rather than repeat the same errors in their mid-session and final reports. These rules and also templates change with the domain.

Students recommended that options be set up for them to view document by structure. That is, the student can select to view the headings and summary only of each section, or headings and descriptions of the type of contents relevant to that section. The template including headings and a description of the role of that part of the document can be included as the document template for the course. Students can then edit this directly, removing the extra information for displaying and printing a normal view.

The main problem with the document development is for students to maintain consistency during changes to the design. The first aspect of this is maintaining consistency within one document or diagram since DFDs have to maintain consistency between diagram levels. Students claimed that by being able to trace views of a concept or subject helps them to develop consistent documents. This can be done by students selecting sections of the document relating to a certain aspect of the design as they develop the document, and then view these paragraphs and diagrams alone when reviewing this aspect, similar to the ideas used in CREWS-SAVRE [234].

The second aspect is maintaining consistency between documents, or parts of the document. There is no access by students to the ideas developed in another's document during offline editing, except through the chat description by that student. This can be analysed by Explain Change pattern. In practise users are presented with a finished worked-over document at each meeting, without a view of the changes. Changes can be included as a separate view option, similar to the use of **views** discussed above. The following aspects are analysed with the corresponding patterns explained in the next chapter.

Document Design The course has certain expectations for the layout of the document which can be used to generate automatic document advice by agents. If the students are to be marked on these aspects of layout, they should have them reinforced during their learning.

Expressed View	Activity Pattern needed
Pre-structural	Design
Pre-structural	Description
Uni-structural	Consistency
Multi-structural and Relational	Construction
Multi-structural and Relational	Feedback

Table 6.6: Relation between Interview and On-Line scaffolding for document writing

Document Description A template can be included to provide the suggested sections and a description of the significance of each section.

Document Consistency The students can link their document by **threads** of subjects or concepts, which they can display in separate student-selected views.

Document Construction Students are continually changing the document. This may be done off-line, or using other software, and when it is shown to other members of the group they may not be aware of the changes between versions. A versioning difference list can be generated when in group session, to provide information to generate queries for the students about alterations and present these as part of group discussion.

Diagram feedback This study used Data Flow Diagrams (DFD) in the document, as an example of a diagrammatic form used in the students' projects. In each discipline different diagrammatic formats are needed to represent various parts of planning or design. The design of Intertac is such that it is envisaged that the rules of each diagrammatic form can be linked to the system through agents, rather than requiring that they be programmed into the tool each time a new diagram type is selected.

The Table below shoes the categories of approaches to document writing and the activity patterns that can support groups using that approach.

6.9 Efficiency of learning

Improving learning also involves linking students to resources that help them change their view of their own understanding of the domain. Creating diagrams is an area of design that requires scaffolding in the form of alternative suggestions, since it is difficult for the computer to analyse if a design is good or bad, beyond whether it fits the rules of that format of diagram. In terms of diagrammatic rules, the feedback rules are identical for all diagrams so it can be easily implemented by rule-based agents, as done for Entity Relationship (**ER**) diagrams in COLER [84]. With other aspects of the design, such as alternative approaches, the feedback will depend on the design that students have developed so the appropriate advice is difficult to automate.

The main request from students is for examples from previous projects. The workshop projects change each session so the requirement is to find the similarity between each current project and linking them to similar products from previous session projects. Not only does the software have to find resources that are similar to the design being developed by this group, but also has to find this similarity in a different project context.

To search files from previous years for specific features as alternative examples for students, what sort of search patterns can be used? In the research, the diagrammatic analysis relied on selecting the basic aspects of each drawing primitive and looking for patterns of similar designs.

Visual issues such as joining data flows in close proximity to each other is a problem. Better-spaced alternative designs can be displayed.

Keyword searches for keywords missing from the processes and data flows can be undertaken. 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 *timing* as a requirement 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, hence abstracted from the details of the particular processes of the student's design.

Similarly, documents from previous years can be searched for changes in patterns. Note that the searched documents will not be about the same software project as the one the students are working on. The patterns observed between distinct projects which are worth noting for search categories are:

Length of section Where students' 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 can be shown an example for them to see what this section covers.

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 can be displayed.

These alternative designs are to be presented in a way that encourages the student to consider why their design/document differs, and how it is dependent on the context they have assumed. This is done in many areas of distance learning with hyperlinked pages to alternative approaches. Intertac is designed to provide a similar environment for group interaction and links can be made within the Intertac software using agents to analyse the patterns of the present design for the selection of suitable scaffolding.

There is no study in this work of the categories of approach to learning efficiency. This aspect of the pattern analysis in fact is more focussed on the generic learning and how to support learning many ways, irrespective of the learner's approach. The patterns are described in the relevant section of the following chapter.

6.10 Conclusion

The common feature of most of the analysis of student learning in this domain has been that many students need to see formats, processes, rules and skills in context. We can give then feedback on the web, in seminars, with their mentors, about the rules of DFDs, the format of their document, the need for integration in their report, etc., but we still get many students who miss out. When this information is provided directly to them, and related to their diagrams, their document, or their terminology, they can at least engage with the information, even if they may still not relate it to their learning.

Where possible we aim to relate the information to the level at which the group or at least some members of the group are working. This is partially to reduce the redundancy in help systems such as that used by Microsoft and also to provide support that is most useful.

From analysing the students activities, the type of support we wish to give to each approach, and the sort of tools students may use to assist their learning, we have described various patterns of activities that can be analysed and used to implement agent support. The following chapter describes the Patterns and how the support will be implemented using a pattern structure.

Chapter 7

Pattern Structure

7.1 Introduction

When the specific patterns of collaboration are isolated they can be used for encoding the data log from the computer-based groupware. This operation can be performed by the agents which will function in a similar way to the human encoder in this research, developing a model of student progression through various learning processes. The agents will then be able to providing appropriate when a process or plan is well identified. To set up the functionality of these agents, we had to initially do this work as human encoders for the specific domain of learning we are studying.

In order to develop a process of supporting learning of any domain in collaborative software, this research started with a study of how this learning is done collocated and online. By experimental monitoring of students' use of the prototype software, ideas are gained of the needs of the students and of the patterns in their processes. A second version of the software is developed to incorporate the proposed modelling and agents. The patterns that have been isolated and incorporated into the software are presented in a pattern structure in this chapter.

The previous chapter looked at the generic activity patterns types that arose during the initial stage of Action Research. The study looked at the complex and multitudinous acts of students on the Intertac-I system. This research now looks at analogies between the different patterns to see if there is any common structure, and to simplify the support, by developing templates for similar patterns of learning activities or interaction.

Then to implement the patterns for learning we link the activities to the scaffolding or **feedback** to be provided by the software for this action. In this chapter we provide a pattern structure for Implementation Patterns. This the second stage, or action stage of the Action Research, as it will enable the development of agent support for the software. The agents will support the learning environment in a way that is designed to improve learning, thus completing the Action Research process.

7.2 Pattern Approach for Implementation

Design Patterns are developed as a vehicle for recording and conveying design experience. They are developed originally by Alexander [13] in the area of *architecture of the built environment*, and are now used in software design (see Gamma, Johnson, Helm, and Vlissides [129]) to analyse the commonality of features in building and their interdependence.

The patterns developed by Alexander [12] consists of the following structural features:

Name A short phrase to refer to the pattern.

Context Abstract description of situation.

Problem to Solve Specification of problem.

Solution Recommended solution and consequences of the solution.

Other studies of group work (Burton and Brown [67],Constantino-González and Suthers [84],Gianoutsos and Grundy [135], McManus and Aiken [231],Műlenbrock and Hoppe [259] and Soller [344]) have looked at how users interact with the each other through the computer, and described interaction patterns as discussed in Chapter 6. These patterns have been implemented in software in various ways. Most involved developing semantic net representations of the knowledge included in the patterns.

Semantic nets (see 2.11.2) have been used to support computer reasoning about knowledge such as Petri Nets used in analysis of contributions for constructive conflict and Neural Nets for analysis of participation and co-operative problem solving. In this thesis, an executable network form of semantic net is used to illustrate the patterns extracted from the work described in the previous chapter.

Also the general patterns of learning approach and the interaction patterns that are observed and described in separate sections, are now combined into one pattern system. These patterns are combined, as the conversation or HCH interaction supported by interaction patterns is as significant in the learning process (see Pask [281]) as the generation of domain concepts which is supported by the learning pattern implementation.

Since the pattern structure is designed for implementing the pattern analysis, we include implementation aspects as part of the initial pattern structure. This chapter deals with Implementation Patterns which are based on analysis of students' interaction and learning which leads to some computer response. These are different to the original descriptive patterns described in Chapter 6, in that the implementation step (or response to be initiated on detection of a pattern) is now included.

7.3 Structure for Implementation Patterns

The aspects that are chosen to describe a pattern which reflects the framework used to analyse the pattern, that is Activity Theory, and the aspects that came out of that analysis, in Chapter 6. Activity theory is used both for the design of the Intertac interface and tools, and the agents to analyse students use of these tools. This is particularly appropriate as the aim of the research is to develop patterns of student's use of the interface and tools, and Activity Theory provides a framework which is based around the interaction between the components of the system, in this case the students and the software, as well as the learning artifacts.

The aspects of Activity Theory that are important in the analysis of learning and interaction patterns are:

Context of the activity, including Interface of the system, Activity of interest, Objective of the learning, Subjects of the activity, and the Community.

Tasks or the content of the learning as represented by the structure of the knowledge concepts or skills used in the approach to the activity including the Steps to be performed, Division of labour between the subjects, and Co-ordination of this labour.

Tools used in the activity including the Means of performing (such as the pattern type. Pattern types are described in Section7.4), and any Rules and regulations to be followed.

Object as the visible representation of the knowledge of the individual or grounded knowledge of the group in terms of what is in the Interface (files diagrams etc). This includes the Outcome of the pattern which is the scaffolding provided or the computer response, and any Activity change as a result of scaffolding or support given.

The aim of developing patterns for learning is to provide learning objectives in a format that can be easily translated into agent support. Hence the patterns are determined as much by the group activities as the outcome or computer response. Given certain aspects that we are trying to teach in the domain, we have certain learning goals. These goals need to be translated into patterns, as the outcome event of that pattern. In particular the support initiated as the outcome when a learning activity pattern is analysed, will aim to provide timely support to the students' learning activities. By analysing their present view, or present state of interaction, the computer agent can provide scaffolding that relates to the present learning goal that is not being achieved.

We separate the learning concepts and approaches supported by the patterns into types, such as group or individual learning. Within each type, a similar approach to code development for agent implementation can be used. This nomenclature of type refers to the focus conception or skill to be learnt, rather than who is learning, or whose actions are analysed by this pattern.

The patterns developed in this analysis are described using the template shown in Table 7.1.

7.3.1 Pattern Weight

Each pattern is assigned an initial weight which provides the value of the significance of the pattern. When implemented in agents this can form a variable that can be altered, such as by a Learner Model. This weight value is based on Schoderbek et al.'s [323] four determinants of complexity based on

1. the number of variables in focus,

2. the number of attributes of these variables,

Name	To provide easy reference.
Context	(Student or Group) and Tool this pattern is related to.
Objective	Description of pattern effect on learning
Subjects	People involved (Group, Facilitator, Single)
Content	Problem to be solved or skill/concept learnt from implementation
Type	Type of learning processes to be extracted See Section 7.4
Initiation	How the analysis is initiated
Tool	Tool that will be used to observe pattern
Implementation for analysis of pattern:	
Conditional	What prior actions signifies this pattern is achieved
Action	Effect of implementation
	Conditional - Further Conditional on subsequent action Action - Response to further conditional
Cognitive Change	Changes that are proposed
Example	Give example/s of the pattern
Weight	Weight of this patterns to be enacted with other patterns Initially assigned on basis of pattern complexity.
Role	Role this scaffolding plays in learning, for the formation of pattern language semantics.

Table 7.1: Pattern Structure

3. the number of interactions among these variables, and
4. the degree of organisation, i.e. the existence or lack of predetermined rules and regulations.

For example, the pattern User Chat Analysis (Six Token) is an Interaction Pattern which looks at the interplay of six tokens (Request Info, Rephrase, Acknowledge, Suggest, Clarification and Explain) out of a possible ten in response to a question, that is 0.6, and the line number referenced, that is if an Explain refers to the line number of a Request. The tokens have the attribute of existence only, so are given a value of 1. The only interaction analysed is the lack of one token amongst groups of the other (Request Info with no Explain), so this is 1. The are no rules or regulations involved. Total weight is 3.6.

A more complex form is to look at all tokens in response to a question, or analyse the explanation received. In the User Explanation Pattern now the tokens, line referenced and the length of response is analysed so there are 2.2 variables, with now two attributes but the search is for only one interaction pattern (i.e. Explain that is short linked to a Request Info), and again there are no rules. So the weight is 5.2.

The concept of weight for a pattern is important in two respects. The higher the weight, the more useful the feedback or scaffolding tends to be if it is linked correctly to the interactions. This is dependent upon the second effect of weight, which is that the higher the weight, the less reliable the analysis that the original interaction warranted the scaffolding.

This arises from the previous discussion in Chapter 4 where the Phenomenographical study of learners' approach and conceptions are related to the focus of the task they draw from their learning activities. As the way students link or navigate these foci increase in complexity, their skills relating to the domain of learning or the particular concept they are learning increase and so does the apparent depth of their learning outcomes. This means that more sophisticated scaffolding is likely to be necessary and assumptions about the student possessing or understanding (or not possessing) certain knowledge cannot always be assumed.

As with any pattern development, it is desirable to analyse the interconnection between patterns in the form of similarities. This connection is made through the type, in that only one type of agent is providing scaffolding at once (see Kutay and Ho [207]), and the role of the patterns. Within a type, if two patterns are complementary, they can support each other, where they are contradictory, they negate each other so may not take action together. Then the pattern with the highest weight in that context will be implemented first.

Once the pattern structure is developed through analysis of pattern similarities, the dissimilar patterns must be linked through the pattern language, which can be used to combine and link the individual patterns. The nature of the patterns and the aspects that they are patterns of, form the structure of the pattern language. A pattern language is the semantics of how the patterns relate to each other, or are distinct. Clearly, in implementation of these patterns, this structure is important for co-ordinating a multi-pattern system.

7.4 Pattern Types

By necessity the development of patterns is never complete, there is always a possibility of finding further patterns. However the study described in the previous chapter is a thorough research into a wide range of learning aspects on a groupware system. By basing the pattern analysis on the fundamental activates on the computer (listed in Section 6.1.1), we have provided a complete basis on which to develop further patterns. Since the groupware software used is quite basic, it is expected that advances in such systems will enable improved analysis of the learner on line, such as modelling the learner and the learning group. However, we claim the implementation patterns for support will still be developed on the same structure.

Firstly we look at how the patterns are divided up into separate types. The first division in patterns relates to the interactions that are analysed. Patterns are divided is between those that deal with interactions within the group and those that deal with learning. The generic Interaction pattern is dealt with in Section 7.9.2 and the generic Learning Activity pattern is dealt with in Section 7.9.3

These are then divided into those that look at each individual and their interaction with the group, and those that relate to the group as a whole. Most patterns fall into the latter category, so unless specified otherwise, a pattern relates to the group as the combined contributions of the students. If a student is working alone, then the group and the student coincide. Note that patterns that deal with a single student are called **User Patterns**. The term 'User' refers to the fact that the agent design is generic to any user participating in a group, although the scaffolding is designed for students learning group interaction.

The next division is between implementation types, or the type of computer response that is generated. The pattern types are developed from Interaction patterns that provide scaffolding on interactions and Learning Activity patterns that provide scaffolding on learning approaches. Using the constructivist principles of learning as enumerated by Savery and Duffy [319].

The teaching roles for constructivist learning environments, as used in the present work, are:

Scaffold to provide scaffolding particularly in planning and design

Feedback to provide feedback for reflection both instantaneously with programmed text or through email to tutor

Alternative to provide alternative views and contexts and encourage the metacognitive skills required in students developing alternatives themselves.

Reflective Provide opportunity for and support for thoughtful reflection.

These are broken up further in terms of how the agent is designed to deal with the students' knowledge.

The scaffolding category is further divided according to the categories of manipulation of knowledge and reflection on knowledge. Manipulation patterns are again divided into patterns that generate actions that can be classified as scaffolding for manipulating conceptual knowledge by arrangement, transformation or deducing or inducing structure. Feedback Patterns are split into Interaction patterns that automatically generate information on procedural skills or actions to encourage reflection, and Learning patterns that respond to to questions on design or group processes.

Finally there are two types of reasoning patterns. Deduction is reasoning from the more general to the specific. This is called a top-down approach, working from theory to confirming observation. Induction is reasoning that works the other way, from example observations to theory generation.

Figure 7.1 shows the Pattern Types developed, using analysis from section 5.8.

7.5 Pattern Summary

What follows is a table of the main patterns derived from the experiments which related to approaches to learning and conceptions, document production and learning efficiency as discussed in the previous chapter. Tables 7.2 and 7.5 show the Patterns with the Content they deal with, their Type and any linked Patterns.

Name	Content	Type	Linked
Interaction Pattern	Analysis of interaction	Reflection	Generic
User Chat Feedback	Analysis of discourse	Reflection	Sub-pattern Chat Analysis
User Chat Explanation	Analysis of Contribution length	Reflection	Extension of Chat Feedback
Chat Analysis	Analysis of	Reflection	Generic
Topic Chat Analysis	Topic used in a Chat type	Reflection	Extension of Chat Analysis
Sub-group	Summarise Discussion	Arrangement	
User Role	Setup for Role	Arrangement	Opens tools for Role or Interface
Planner:			
Template Course Plan	Provide a template	Deduction	Used by many
Group Planning Schedule	Maintain Plan	Arrangement	Template Plan
Editor:			
Template Document	Document Outline	Arrangement	Used by many
Editing	Process of Editing	Reflection	Uses Edit Plan
Edit Plan	Workflow	Arrangement	Used by many
User Explanation	Analysis of Contribution length and time	Reflection	Extension of Chat Explanation
View Make	Make Topic Views	Arrangement	Template Doc.
View Use	User Views	Reflect	Template Doc.
Document Rule Checker	Format Document	Deductive	Template Doc.

Table 7.2: List of Patterns

Name	Content	Type	Linked
Design:			
Change Design	Search and Display changes	Deduction	Design Pattern
Context Design	Search and Display single keyword context	Arrangement	Design Pattern
Alternative Design	Search and Display Alternatives	Provide Alternative	Doc. Rule Checker
Explain Change	Change in file from last version	Deduction	Change Pattern
Re-Design	Students to give alternatives	Develop Alternative	Rule Checker
Justify Change	Late change in design	Reflection	Design Pattern
Rule Checker	Checks rules	Deductive	Uses Design Pattern
Diagram Rule Checker	Check Diagram	Deductive	Design Rule
Diagram Complexity	Attain top down view	Deductive	Alternative Design
Design Pattern	Search similar designs	Provide Alternative	Used by many
Require Model:[1]			
Decision Making	Stages to follow	Transform	Initiates Stage agents
Learning:			
Learning	Follow Learning process	Generic	generic by many
Lexicon	Use of concepts	Induction	Used by many
Concept Extension	Extend use of concepts	Induction	Chat Feedback

Table 7.3: List of Patterns (cont.)

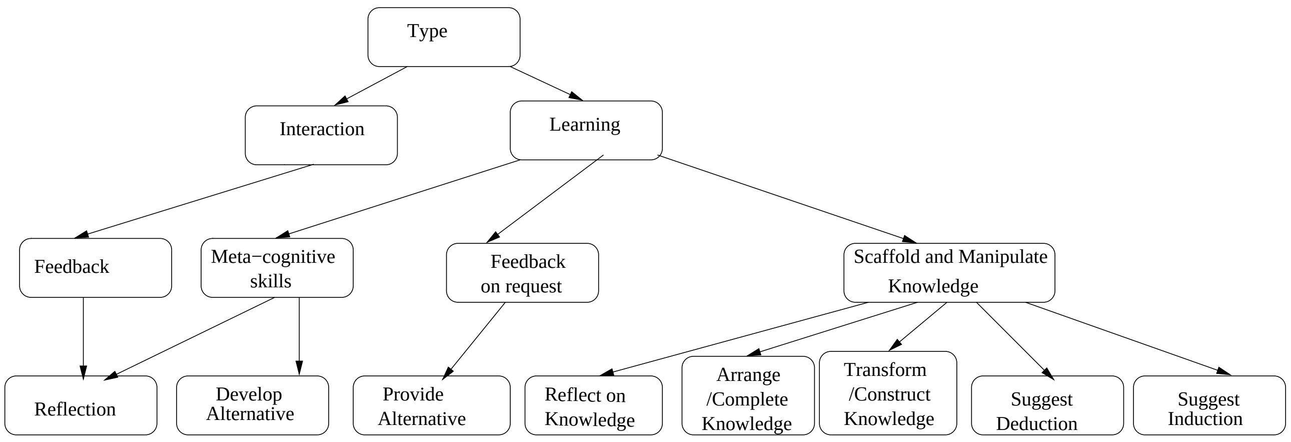


Figure 7.1: Tree of Pattern Types

7.6 Expansion of Pattern Types

This chapter lists the patterns resolved in interactions and learning within the software environment. We claim the set is complete, however each pattern represents many different implementations, in that a pattern may be used to analyse for different depths of activity, depending on the rule used for the conditional.

All patterns may appear in either a deep level of activity or shallow, however their form will differ between different depths of activity. These patterns are used to analyse depth where one enactment will show a good depth of activity and an alternative enactment will show less depth. The conditional is the pattern rule which sets the method for analysing this depth. The implementation of the agents will provide feedback relating to the particular conditional of the enactment.

Some examples of specific patterns are given in the Section 8.1. However first we present the types of activity which can be defined as a pattern, and which are therefore available for such analysis.

Notation

When listing some of the pattern types developed in the present work, the following convention is used. Where features from Intertac-II are used, such as *Section View* which is an option to view a document as all its sections each with a summary, this is written as starting with capital letters. Where the concept has not been previously introduced, this term will be explained.

Pattern names are written as starting with capitals, and there is much cross reference between patterns and their sub-patterns in this chapter. In particular many patterns rely on the *Planning Schedule Pattern* to determine the stage of the workshop that has been reached in relation to themes that have been covered, work that is complete, and deadlines for submissions. The Planning Schedule keeps track of tasks, their completion times and their dependencies.

As mentioned above, patterns that analyse a single student at a time, rather than the group as a whole use the prefix *User* and are often just specific applications of the more generic rule for the group, which has no prefix word.

Diagrammatic Representation

The patterns are also represented visually as networks to assist the development of the agents. It is noted that these are event based networks rather than state based networks, in that the analysis does not seek to analyse the state the group or user may have reached, but rather analyses the transition events and uses these to develop a response that does not necessarily assume a certain interaction state has been previously reached.

This is similar to work by Soller and Lesgold [345] where the state of the learner is only known to a certain probability from the previous states and the transitions made. To simplify the diagrams this research has removed the reference to the unknown states. The network diagrams provided do not provide all details where a pattern is repeated within another pattern, to enable them to be read easily.

What follows is a section for each type based on sections for results in Chapter 6, which are:

1. Depth of Activity
2. Efficiency of Document Production
3. Efficiency of Learning.

A further section on Group Processes covers the processes that are highlighted as areas for learning in the course and is included for completeness. This section includes the area of Decision support which gives an example of combining agents to form a more thorough analysis of the activity.

7.7 Depth of Activity

The section looks at patterns of Learning that deal with improving approaches to learning activity in group processes and improving the use of domain specific concepts.

1. Group Learning Process,
2. Document Integration and Editing,
3. Requirements, and
4. Concepts.

7.7.1 Group Learning Process

This section covers specifically patterns of simple processes that reflect good or poor learning from the group interaction.

Much of the development of process skills by the group such as participation, decision making and conflict are analysed and supported already in other groupware systems. The approach here is similar, but it is worth presenting the types of patterns developed in this area.

The possibility of scaffolding for different approaches to the group process is limited at present. With the development of a Learner Model a closer approximation can in future analyse the level of the students or group awareness of group processes, through analysis of the history stored in the model.

This section looks at the specific group processes which form patterns of group interaction. These combine different Users activities through User patterns and then analyse as an interacting group.

Chat Analysis

The patterns relating to analysing chat contributions are generic to all patterns that relate to analysis of the discourse, either through sentence openers or tokens. Rather than describing specific patterns for every group process encountered, such as sharing knowledge or effective decision making, the generic patterns provide a structure for all such pattern.

These patterns arise from the interaction of contributions in the Chat window. The contributions are indexed by a token indicating the intent of the comment. While the manner in which students use tokens will vary according to their understanding of the words, the analysis of the patterns from students Chat logs locates the most significant paths that lead to the outcome that is being analysed.

That is a pattern is a series of steps through the tokens in sequence that is likely to give an outcome, such as when a student asks a question, there are paths that are more likely to signify that the students will not feel they have been answered sufficiently, and are less likely to understand the concept being questioned at the end

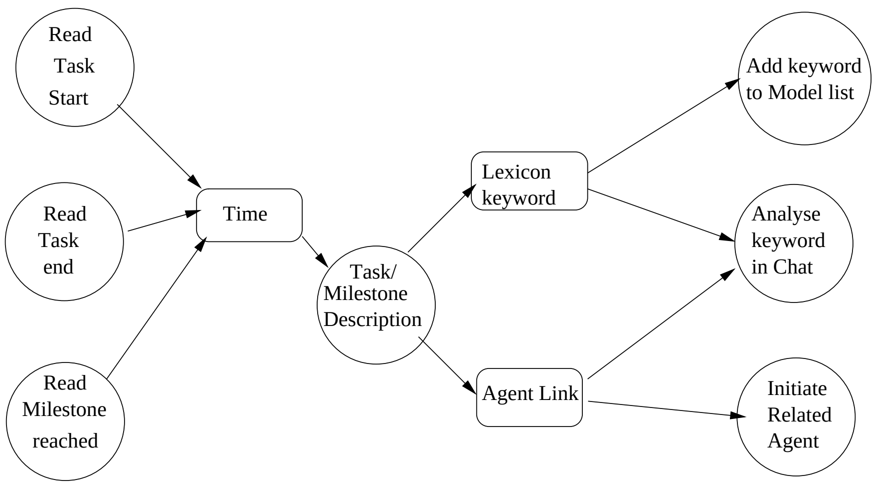


Figure 7.2: Pattern — Planning Schedule

of the sequence, or even the session.

There is also a second token included in the Chat contribution giving the subject of the discussion, or the main topic. This can be used to initiate comments online from the information about that topic, stored in the system.

These patterns of Chat Analysis are described later in Section 8.1 as they are similar in approach and results to previous work in the area, so provide a benchmark to compare the implementation of the agent with alternative analysis of feedback for groupware.

Course Planning Schedule

However depth of work in the group also involves the interactions during editing of documents and diagrams, and how this relates to the the course timetable.

There may be many Planning Schedule Patterns (see Figure 7.2). The Course Planning Schedule Pattern is linked to the original template supplied by the course lecturer. The Group Planning Schedule Pattern is linked to the edited version of the plan made by the group. If this does not exist, they form the same pattern. A

Name	Course Planning Schedule
Context	Gantt charts links to other agents
Objective	Alert students to time based activities
Subjects	Group
Content	Track activities that have deadlines and maintain a record of their completion. Supply Group and other agents when key points of time in the planning are reached, such as milestones.
Type	Arrangement
Initiate	on start of Intertac using Course Plan
Tool	Planner tasks
Conditional	If task end-time has been reached on Intertac startup
Action	Alert students to need to complete and record task finish flag in Group Model for use by other agents.
Changes	Deadlines have to be realistic and followed
Example	Milestones of submission of Reports are fixed. Also covers Course Planning Schedule, Group Planning Schedule and Topic Course Planning Schedule.
Weight	6
Role	Linked to all course activity triggered agents

Table 7.4: Planning

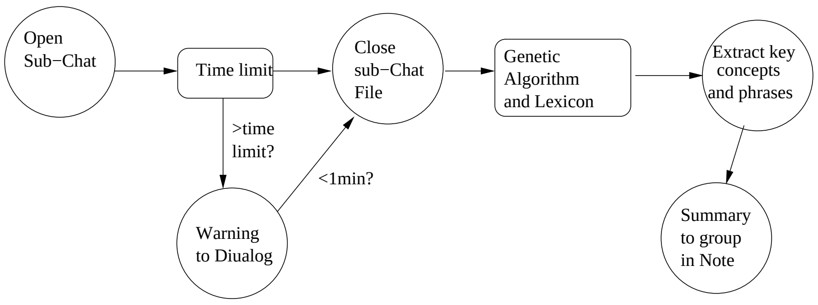


Figure 7.3: Pattern — Sub-Group Chat Pattern

sub-pattern of the Course Planning Schedule Pattern is the Topic Course Planning Schedule Pattern.

The Group Planning Schedule Pattern will include requesting the group flag items if they are completed, once these items fall due. This is aimed at keeping groups aware of schedules and the work involved in keeping them up to date and relevant, without taxing their effort too greatly.

Sub-Group

The main role of the Sub-group Pattern is to develop a summary of the work done in the small group, to present to the larger group session. It should not be used by any other pattern, except User Patterns (i.e. patterns relating to a single student at once) relating to the students in the sub-dialogue or sub-session.

The Sub-Group Pattern is used to analyse each new Chat window that is opened to provide the support of the session (see Figure 7.3).

User Role

When a student selects a role, agents are used to monitor their behaviour within that role, and scaffold their processes. For instance the facilitator can be given a window in which to summarise discussion, or to add matters that have been raised but not

Name	Sub-Group
Context	Sub-Chat
Objective	Encourage students to report small group findings to all of the group
Subjects	Sub-Group
Content	Provide summary to larger group
Type	Arrangement
Initiate	on selection of Chat button
Tool	New Note Window
Conditional	If Chat is closed
Action	Delay close until a summary is requested and entered in new window, or Note Window closed
Changes	Sub-Groups should not work separate from main group
Example	If subgroup forms while another individual is editing a diagram or document that student should be involved in the decision that may relate to their editing. The summary can initially include the information given to students by agents during the sub-group discussion
Weight	5
Role	Works with User patterns of the relevant students in these sub-session or sub-dialogue.

Table 7.5: Sub Group

Name	User Role
Context	Role Windows
Objective	Support use of Roles
Subjects	Student adopting that role
Content	Specifies set up for each role and the patterns for that role
Type	Transform
Initiate	on role selection button
Tool	Role Windows
Conditional	If role selected by a Student
Action	1. Set up windows for that role and display description of that role to first time student (information required from Learner Model. 2. Monitor Role Window with specific Patterns for that Role.
Changes	Student aware of function of the role
Example	Facilitator should have a window open that can be used to enter summaries from group work. User Facilitator Role can advise on how to summarise if this window is not used.
Weight	4
Role	Transform

Table 7.6: Roles

decided in the group, either by typing or cut and paste from other documents.

The separate roles use different instances of the generic pattern (see Figure 7.4). The roles designed for Intertac are:

Facilitator Has ability to highlight and cut and paste items from other windows to the facilitator window. Can delete these items when dealt with. Also become the broadcaster of the session, unless they become disconnected from the on-line session.

Recorder Has ability to view Facilitator window (as can all students) and can save it as a summary of the discussion.

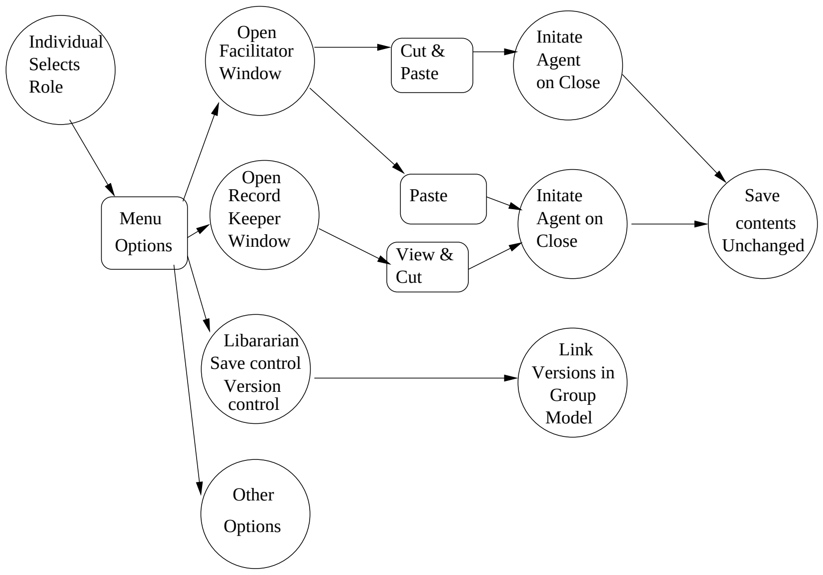


Figure 7.4: Pattern — Set up Role

Librarian Has the ability to save all files under concurrent version control (CVS). If not using CVS they are advised not to save a session's files until all changes have been accepted by group.

All roles are recorded in the Learner Model, which can then act as the record for an agent to recommend a rotation of roles each session.

Explain Change

This pattern (see Figure 7.5) relies on correct use of the Librarian Role, which ensures that each version is retained of a file until changes are approved by the group. This is similar to the Justify Change Pattern that will be used to ensure that major design changes do not occur later in the report writing, without comment.

One aspect of integration could in theory be checked by computer, and that is the searching for keywords common to the requirements, the DFDs and the specification. These keywords are not necessarily part of the course topics, so will be

Name	Explain Change
Context	Group/Editor and Diagram Tool
Objective	Enable group to keep track of changes
Subjects	Group
Content Type	Any changes made not in group mode should be explained Deductive
Initiate	On opening of document
Tool	Dialog window
Conditional	If the document, or diagrams, opened in a group session differ to those saved last session (using versioning).
Action	Conditional Action: At start of course: The group should be given a list of changes analysed by computer. Conditional Action: After a few weeks: The group should be given a summary of changes, using the Design Patterns Conditional Action: After the Interim Report: The student opening the altered file should be asked to summarise the changes in a separate window.
Changes	Summaries of changes should become automatic. 1. Changes have been made that may affect others' work. 2. Changes can be summarised for group. 3. How to summarise changes.
Example	When a use changes the DFD design, the other student developing the explanation in the Document may not realise to include the change.
Weight	6
Role	Relates to Sub-Group but not linked. Linked to Course Planning Schedule, and Design Patterns.

Table 7.7: Explain Change

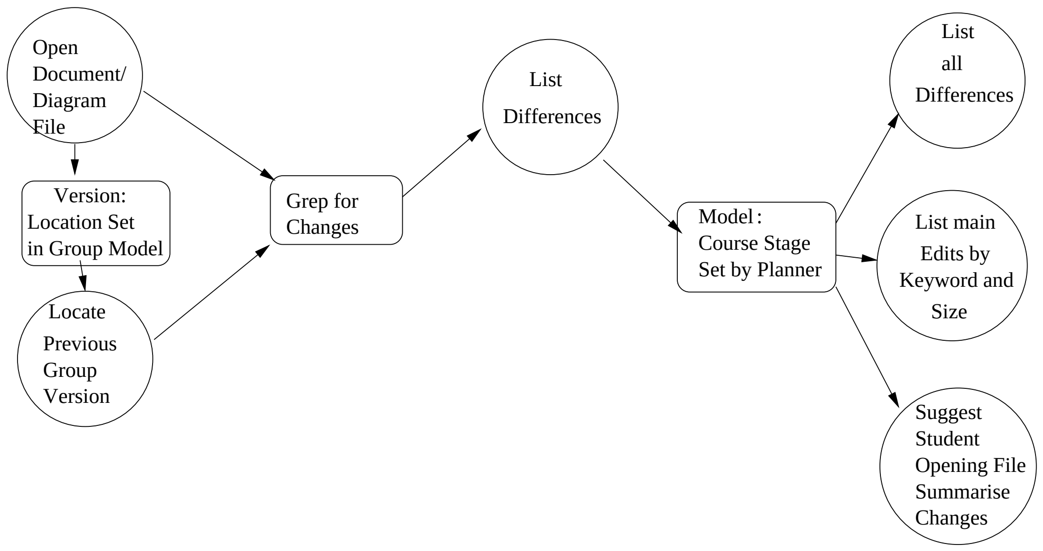

Open Document/ Diagram File
Version: Location Set in Group Model
Locate Previous Group Version
Grep for Changes
List Differences
Model: Course Stage Set by Planner
List all Differences
List main Edits by Keyword and Size
Suggest Student Opening File Summarise Changes

Figure 7.5: Pattern — Explain change made

difficult to locate and track. This is beyond the scope of the present research and experimentation.

Decision Making

The analysis of decision making processes is more complex than the other patterns which involved consecutive stages with the same conditional. In this case each stage is analysed by different conditionals. It has been found that teaching decision making concepts or process descriptions in the course seminars is not sufficient, and in fact the actual skills needed to be practised for students to be successful. This may become 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 McLeod [230] were observable as:

Exchange of information Either dialogue contribution with similar words, or files open with similar content or structure.

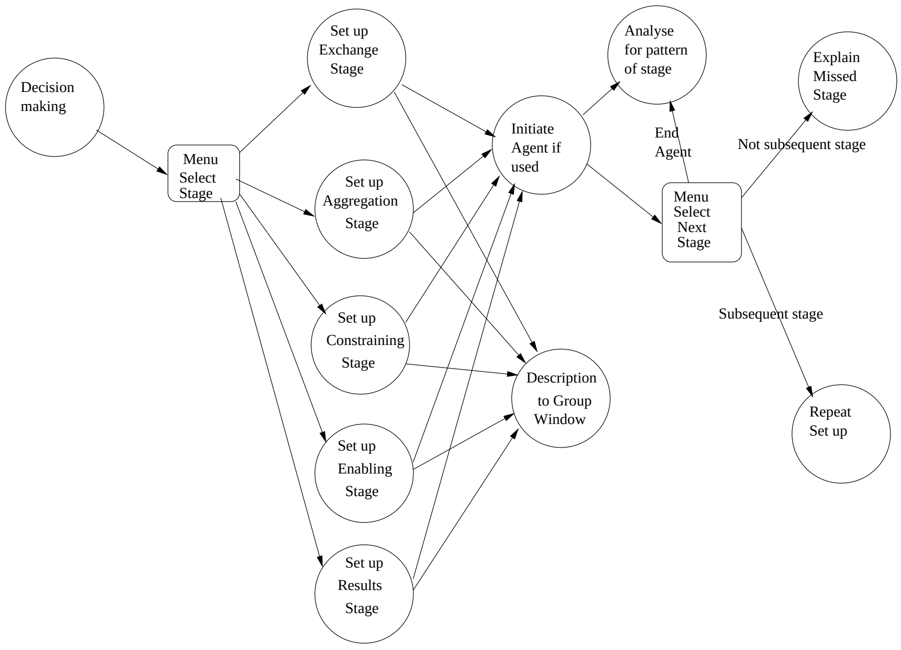


Figure 7.6: Pattern — Decision making Wizard

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 Wizard for Decision Making (see Figure 7.6) will activate the agents for each Decision Step in order. The group may only select the next stage and not skip steps.

Name	Decision Making
Context	Intertac
Objective	Support decision making stages
Subjects	Group
Content	Stages are presented in a wizard format then process is monitored by agents for each stage.
Type	Transform
Initiate	on selection of menu options
Tool	Wizard and support agents
Conditional	If Conflict is identified:
Action	Suggest use of stages in Wizard.
	Conditional: As next stage is selected in wizard:
	Action: Activate relevant agent and monitor activity
Changes	Aware that decision making follows stages and stages have certain attributes.
Example	If group is changing each others edits rapidly They can be advised to use Decision Process. As each stage is selected, wizard activates agents for that stage which starts with a description of the relevant process.
Weight	8
Role	Extension of Editing

Table 7.8: Decision Making

Name	Decision Making Exchange	
Context	Decision Making Stage 1.	
Objective	Learn the need to share information.	
Subjects	Group	
Content	Exchange involves various Chat patterns and repeated use of terms of debate.	
Type	Transform	
Initiate	on selection of wizard option	
Tool	Chat and Document files	
Conditional	If wizard item *Exchange* selected:	
Action	Summarise process on Java Message window.	
	Conditional Action	Message closed with non-null phrase: Monitor occurrence of words in phrase
	OR	
	Conditional Action	Message closed with null phrase: Monitor occurrence of repeated words
	Conditional Action	No monitored repeating of terms: Alert group to lack of common ground.
Changes	Knowledge of the *Exchange* stage and limited feedback on its occurrence in their process.	
Example	User select phase but little dialogue	
Weight	8	
Role	Extension of Editing	

Table 7.9: Decision Making Exchange

However they select the time when they feel they can move on. The first stage of Exchange is the only pattern expanded in full here.

The Decision agent is student 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, that describes each step in the process, with a button for the student 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 dialogue 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 Individual Model data during each stage and presents this as the data for analysing the re-occurrence of that stage. The Group Model has been designed with such applications in mind.

7.7.2 Document Integration and Editing

The next patterns described are those that deal with students integration of aspects of the design in their design documents, and the patterns of editing that can be analysed through the group documents. These patterns are built on some of the basic interaction patterns from the previous section.

View Use

View Use (see Figure 7.7) is similar to View Make which focuses on the *Thread Views* that students must set up in their document for tracking threads through the document and associated diagrams. In View Make the aim is to analyse if the group is making use of the facility by constructing threads through the document to track their ideas, or the requirements.

Name	View Use
Context	Editor and Tool
Objective	Encourage overview of documents
Subjects	Group
Content	Using different views on the report can produce different levels of understanding of the document, and helps in correcting the document.
Type	Course Plan Timer
Initiate	Permanent
Tool	View types selected on Editor
Conditional	If Group has not selected a particular view.
Action	Conditional: If have not used a particular view Action: Suggest they check the view for aspects that particular view supports, e.g. *Section View* shows what sections are included plus a summary of their role in the document. Conditional: A week before Interim Report is due, check if *Thread Views* have been used recently Action: Recommend *Thread Views* to be used for consistency checks. Conditional: Two weeks before Final Report if *Thread Views* has not been used since Interim Report Action: Recommend *Thread Views* for consistency checks
Changes	Students are aware of attribute of each section.
	Students check document entirely before submission, particularly for consistency.
	Reinforce consistency check.
Example	Students can be encouraged to select each requirement as a View and link the parts of the document that describe each requirement.
Weight	3
Role	Relies on Planning Schedule

Table 7.10: View Use

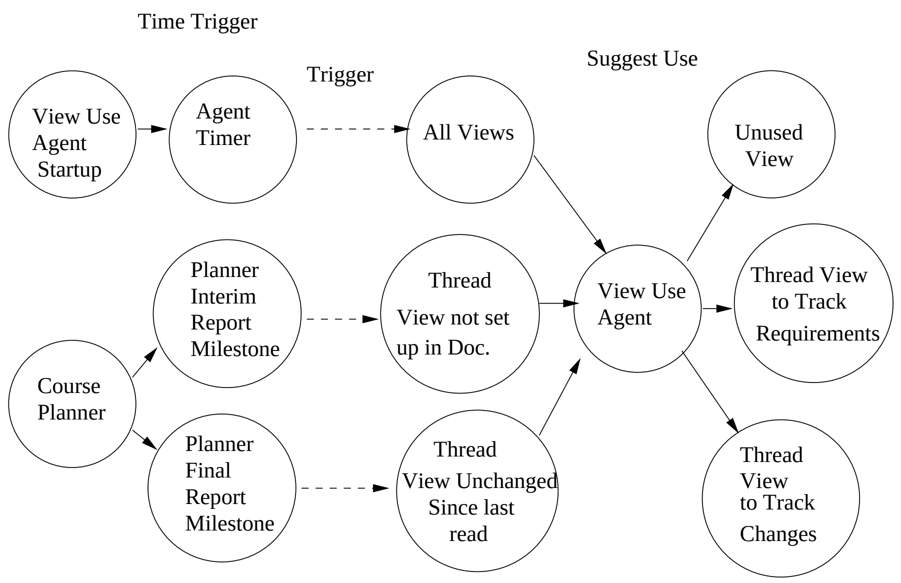

Time Trigger
Trigger
Suggest Use
View Use Agent Startup
Agent Timer
All Views
Unused View
Planner Interim Report Milestone
Thread View not set up in Doc.
View Use Agent
Thread View to Track Requirements
Course Planner
Planner Final Report Milestone
Thread View Unchanged Since last read
Thread View to Track Changes

Figure 7.7: Pattern — View Use

Editing Plan

The Editing Plan (see Figure 7.8) relates the editing of documents to the Schedule Plan of the course. Editing at different stages can reflect good planning as well as the level of changes expected as the course or project progresses.

User Editing

The editing pattern looks at the sequence of edits and deletes in the files, either by an individual, or in the group. User Editing follows a single User, but the Group pattern is similar in that the same sequences reflect the same level of cohesion of ideas, or conflict, in the group.

As an extension, the Editing Plan Pattern (see Figure 7.8) uses the Editing Pattern (see Figure 7.9) to monitor the editing by the group then used the Course Planner to determine the level or type of response.

Another pattern that is related to Editing Pattern in the domain of DFD design

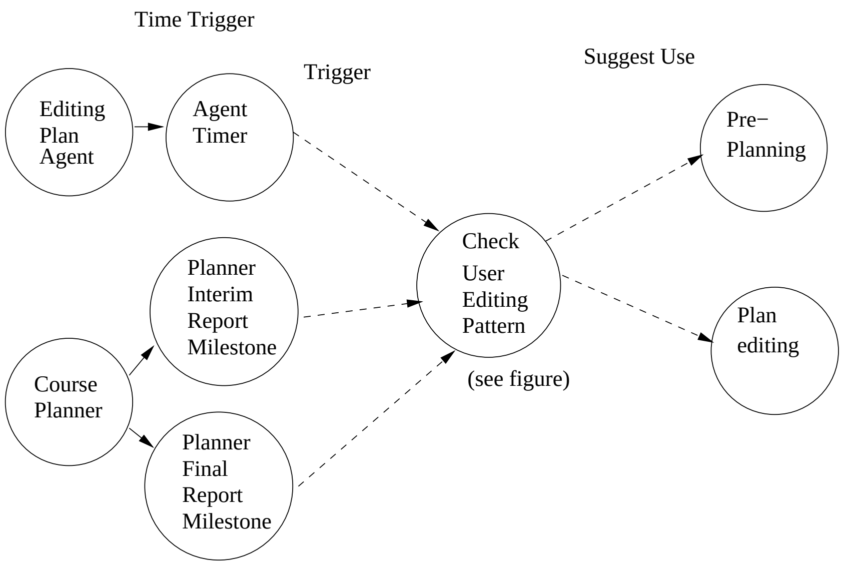


Figure 7.8: Pattern — Editing Plan

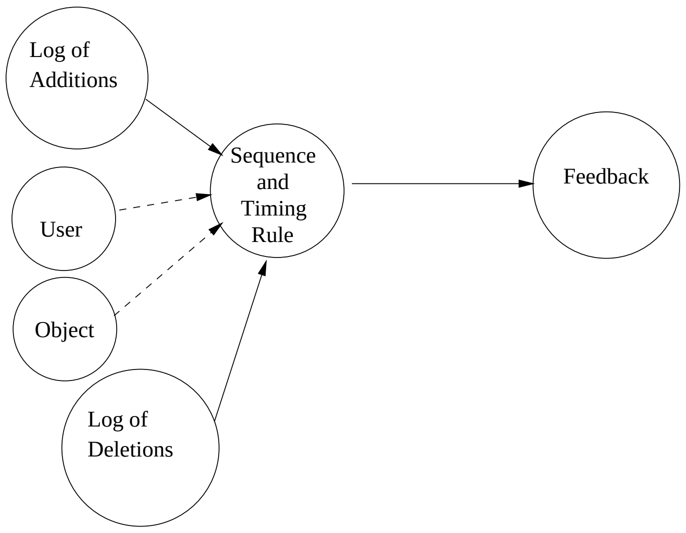


Figure 7.9: Pattern — Editing

Name	Editing Plan	
Context	Editor and Diagram Tool	
Objective	Encourage good organisation of work by pointing out bad practises	
Subjects	Group	
Content	Work Flow arrangement ensures that the basic design outline is completed early.	
Type	Arrangement	
Initiate	Permanent	
Tool	Deletions related to stage in course	
Conditional	If early section of Document or higher level of diagram is edited later in planning.	
Action		
	Conditional Action	First time pattern detected Ask group if that change could be envisaged earlier.
	Conditional Action	Second time pattern detected Ask group if they are considering other similar changes, and thus complete this part of the design now.
Changes	Incomplete initial design leads to more re-work later. 1. Encourage thorough initial planning and thinking through the design.	
	2. Encourage complete re-thinking of design before starting edit.	
Example	When groups do not discuss problem first but start with task, they tend to find errors in higher level design later in the design stage.	
Weight	8	
Role	Linked to Planning Schedule	

Table 7.11: Editing Plan

Name	User Editing
Context	Editor and Diagram Tool
Objective	Monitor session editing patterns
Subjects	Student
Content	Edits should not add and delete work repeatedly
Type	Arrangement
Initiate	Permanent
Tool	Log of adds and deletes
Conditional	If material just deleted was recently added
Action	Ask Student deleting if they need to plan more
Changes	Students tend to add and delete when not sure about next step.
Weight	6
Role	Used by Editing Plan

Table 7.12: User Editing

is the need to maintain a low limit of inputs and outputs to a process or the diagram become cluttered and overloaded with information. When designs are edited during the course, the design tends to be overworked, and this problem of information overload occurs. These types of analyses are done through the Diagram Rule Checker pattern.

7.7.3 Requirements

A special case of the Explain Change Pattern (see Section 7.7.1 is the Explain Change Requirements Pattern that ask the group to include an explanation in the report whenever they change requirements, so that the client will be alerted to the changes and made aware of why they are necessary, in case they do not wish to accept the change.

Also, the Course Planning Schedule Pattern (see Section 7.7.1) forms the structure for all patterns relating to the topics of the seminars planned at each stage of the workshop course. The Topic Course Planning Schedule alerts groups to the lack of use of thematic language or jargon in their discussion and report. This is an arrangements type of agent that encourages students to link this recently heard knowledge to their own work, and hence possibly incorporate it into their own knowledge arrangement.

When used in workshops with a course planner template provided, the template includes the planned time of introducing different topics into the seminars. These are compared to discussion to verify if they are being taken up by the groups, as a form of scaffolding.

7.7.4 Concepts

The patterns in this section look at the cognitive state or development of the group. There are two types of patterns that are derived from analysis of the group actions in relation to their understanding of the concepts of the domain. These patterns can be supported by tools which users can choose to use, as well as computer initiated activities.

From the analysis in the previous chapter, it was proposed that by providing a *lexicon of terms* and concepts which students can use as topics for their contributions to Chat, students are encouraged to focus their conversation by grounding it in the course. These topic tokens can be used to extend the capability of the analysis.

Also, the timing of interjections in Chat can be further analysed by the use of a *course plan* that provides a time-line for the expected development of various concepts. The timeline and Chat logs are compared to verify if they are being taken up by the groups. If not this is an alternative trigger for feedback.

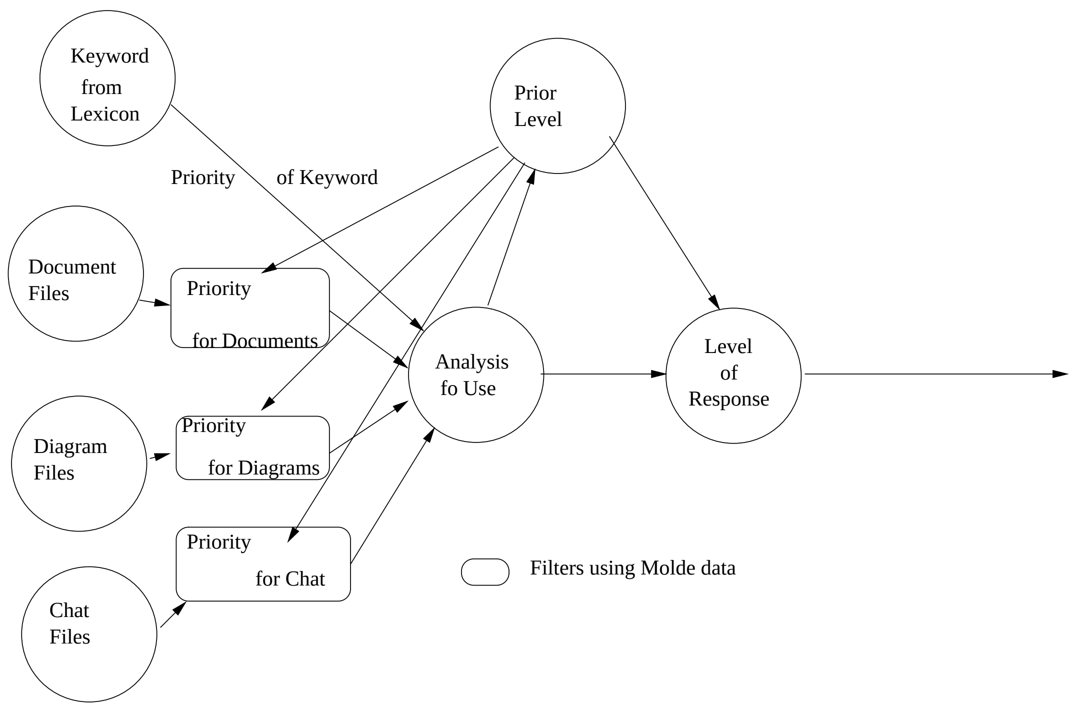


Figure 7.10: Pattern — Extending concept use

Concept Extensions

In order to develop different approaches to the important concepts the patterns in which a concept is used and be analysed and scaffolding developed with the aim of encouraging students to link their understanding of a concept to a different focus on the concept.

The Tool used for feedback will depend on the tool that contains the concept discussion. If this is in the Chat tool then advice is focused in Chat also (see Figure 7.10).

Topic Chat Analysis

The students can also receive support for how much and what type of conversation has been linked to each word, or phrase, in the domain lexicon. For instance has

Name	Concept Extensions
Context	Editor/Chat/Diagram
Objective	Extend the experience of the use of concepts in context
Subjects	Group
Content	Provide deeper use of concept as the lexical word is used during the group work
Type	Inductive
Initiate	Permanent
Tool	Dialog Window or Chat
Conditional	When use of concept reaches certain level
Action	Display information for next concept extension
Example	Each rule will change with each concept. For instance discussion of titles of the report sections one time in Chat causes a comment of this section of the report template to be displayed. Discussion of the concept *constant* used in B code either in Chat or in the Document five times causes a comment about this aspect of B to be shown in the Dialog Window
Weight	4+
Role	Extends the separate patterns of lexicons of document template, planning activities and specification terminology.

Table 7.13: Concept Extensions

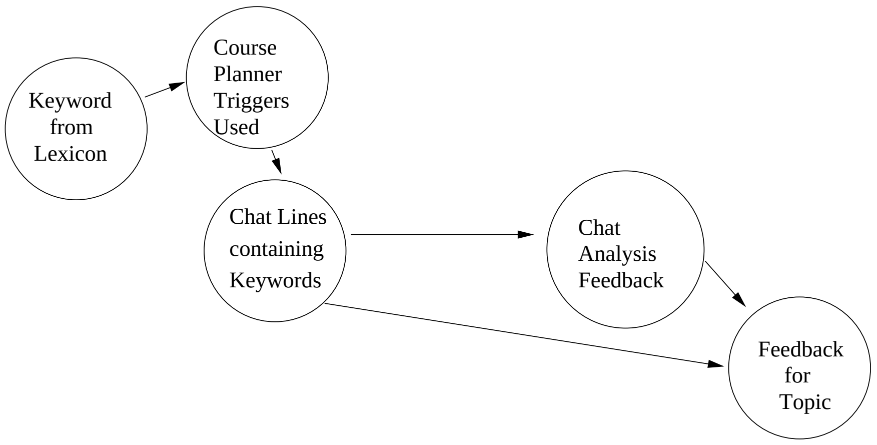


Figure 7.11: Pattern — Analyse Chat Topic

there been much explanation, disagreement, argument, decisions made, and so forth. This pattern relates to the Chat Feedback Pattern which has already been covered, as it deals with user interactions.

Topic Chat Analysis relates the topic or concept to the type of Chat Analysis used (see Figure 7.11).

7.8 Efficiency of Document Production

The areas where students needed support are analysed. Some of this support comes from recommendations for tools to be provided for use in learning. Other support arose from the need to analyse past and present designs for patterns. Patterns include the document structure and diagram structure as well as how students use the tool provide to improve the efficiency of their work:

Template To provide document description a template can be included with the suggested sections and a description of the role of each section.

Name	Topic Chat Analysis	
Context	Chat	
Objective	Encourage appropriate discussion of topics in Chat	
Subjects	Group	
Content	Verify the a topic has been dealt with in the Chat discussion according to Chat Analysis patterns.	
Type	Reflection with feedback from Model	
Initiate	Course Plan Trigger	
Tool	Topic selected for Chat line	
Conditional	Topic Course Planning Schedule triggers occurrence of Topic in workshops OR Topic is used in Chat	
Action	Verify Chat Analysis Pattern used with Topic selected	
	Conditional	If Chat Analysis Pattern found from Topic lines
	Action	Notify Students to manner Topic has been handled in Chat
Changes	Group aware of how they are dealing with Topics	
Example	Use Topics Course Planning Schedule to verify Topics are discussed or not using User Chat Analysis Pattern.	
Weight	4 +number of token/10+number of links	
Role	Extension of Chat Analysis Pattern	

Table 7.14: Topic Chat Analysis

Document Rule Checker Pattern To improve document design the course documents provide certain expectations for the layout of the document in the form of a template, which can also be encoded as automatic document advice agents. If the students are to be marked on these aspects, they have them reinforced during their learning.

View Make and Use Patterns To improve consistency, the students can link their document by **threads** of subjects or concepts, which they can display in separate student-selected views.

Change Pattern To improve document construction when students are continually changing the document, either when off-line or using other software, a summary of these changes can be displayed for group comment. This ensures that other members of the group are made aware of the main changes between version. A versioning difference list can be generated when in group session, to provide information to generate queries for the students about alterations and present these as part of the agenda for group discussion.

Diagram Rule Checker Pattern This study used Data Flow Diagrams (DFD) in the document, as an example of a diagrammatic form used in the students' projects. In each discipline different diagrammatic formats are used to represent various parts of planning or design. The rules of each diagrammatic form can be linked to the system through agents, rather than requiring that they be programmed into the tool each time a new diagram type is selected.

The students can be provided with a document template for the course that provides views already with sections and their definition or role in the Report structure. The View Make Pattern described above encourages students to make use of the *Topic Views*. They are then encouraged to inspect these views with the View Use pattern.

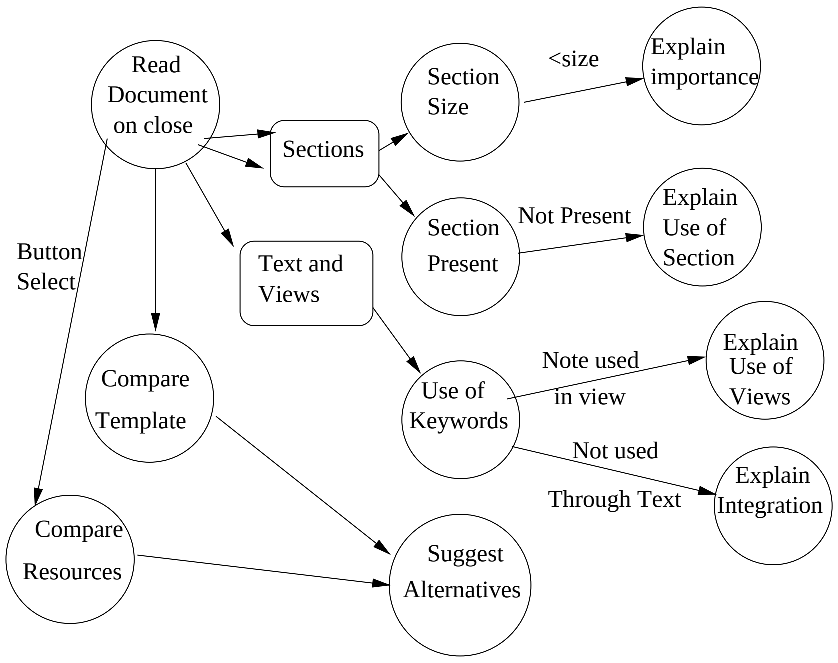


Figure 7.12: Pattern — Document Rules

Also Explain Change Pattern described above is used to ensure that the document remains consistent during editing. This will remove some of the construction problems seen in the documents that are submitted.

Document and Diagram Rule Patterns define the rules for the report layout and the diagrammatic rules for the domain. While these change between domains, the general form will remain the same for some particular patterns.

The Document Rule Checker (see Figure 7.12) for documents will look for aspects such as what sections are used, their relative length, etc. The Diagram Rule Checker (see Figure 7.13) has a definite list of rules for that diagram format, such as the extensive rules for DFDs.

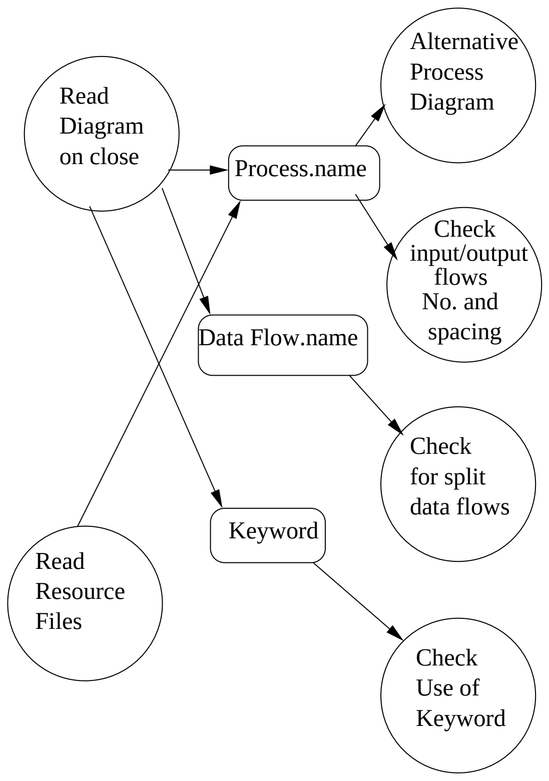

Read Diagram on close
Process.name
Alternative Process Diagram
Check input/output flows No. and spacing
Data Flow.name
Check for split data flows
Keyword
Read Resource Files
Check Use of Keyword

Figure 7.13: Pattern — Diagram Rules

Name	Rule Checker
Context	Editor or Diagram Tool
Objective	Check rules in documents/diagrams
Subjects	Group
Content	Check for each rule at finish of drawing
Type	Deductive
Initiate	on close of file
Tool	Note window
Conditional	If rule not followed
Action	Open Note with text description of Record found with error
Example	If DFD Tool.process found with too many inputs and outputs close together then display the text string of the Process and error
Weight	7 (Document) & 7 per rule (Graphic)
Role	Reads all design rules and searches using Design Pattern

Table 7.15: Rule Checker

7.9 Efficiency of learning

Next week look at the patterns that emerged as relating to the efficiency of groups learning new concepts or skills. The output is to provide timely alternative views and to question exiting views, to stimulate changes in approach. The patterns are described below.

Design Patterns The main request by students was for examples from previous projects. The course projects change each session so the requirement is to find the similarity between each current project and linking them to similar products from previous session projects. Not only does the software have to find resources that are similar to the design being developed by this group, but also has to find this similarity in a different project context. These alternative designs are presented in a way that encourages the student to consider why their design/document differs, and how it is dependent on the context they have assumed. This approach to feedback is used in distance learning where web courses use hyperlinks to alternative approaches to solving a problem, however in the pattern approach we provide feedback to any pattern matching solution provided by the user.

Interaction Patterns During the entire session on-line, the students are interacting through the various tools. Often their use of the tools to seek and gain answers to questions, or discuss differences, is poor. In particular students are often inexperienced in the steps required to resolve conflict or to even acknowledge and use conflict constructively. Some basic analysis can be made through the use of Tokens to describe the student's intention in Chat contributions plus their actions in other tools.

Learning Activity Patterns Similarly, the students are often inexperienced in learning course material at any detail. Students have been encouraged to learn for assessment, and avoid the extra work required to extract meaning from their courses. The main aim and design motivation of the workshop courses, and Intertac, is to motivate and encourage students into a deeper approach and conception of their learning.

7.9.1 Design Patterns

Design patterns are used to search files from previous years for specific features as alternative examples for students, or what sort of search patterns is used? In the research, the diagrammatic analysis relied on selecting the basic aspects of each drawing primitive and looking for patterns of similar designs.

To search files from previous years for specific features as alternative examples for students we use the Design Pattern. This pattern analyses similar features of designs. This can be used both to check for similar design aspects in previous years and also for differences between an older version of a design and the more recent version.

Visual issues which relate to rules of the diagrammatic from used in the domain. In the case of DFDs we found the issue of spacing in designs was a suitable pattern for searches.

Keyword searches can be specified for keywords specific to one years project, or across many years for generic features such as logins and the designing of time into the system.

Design Steps such as top-down rather than bottom up approach can be displayed in a sequence of diagrams from any problem domain.

Similarly, documents from previous years can be searched for changes in some patterns. Note that the searched documents will not be about the same software project as the one the students are working on. The patterns observed between distinct projects which is worth noting for search categories are:

Length of section relating to a template format specified for the course, for instance the introduction is shorter than the Requirements section.

Use of sections such as required sections missing or out of order, then examples using the correct template with description of the role of each section can be selected for display.

Expert Sections that show a good handling of sections that are generally not well understood, such as the Executive Summary in the domain of this research.

Finally, there is the Design Pattern which is not an implementation pattern. It is a pattern used by other implementations to match the Design Pattern of a group's design or part of their design to that of historical documents, using a comparison algorithm on the graphical representation of the data in the files. Table 7.16 below gives a list of search criteria available. Files are searched on the basis of similarity in *Attribute 1*, looking for alternatives in *Attribute 2*, as listed in the table. The **content** column gives a summary of learning content of the resultant designs that are found.[2]

The Design Pattern is used to:

Alternative Design Pattern (ADP) Search for alternatives to the present students' design.

Change Design Pattern (CDP) Search for changes in design between versions of a design.

Context Design Pattern (TDP) Search for related aspects of one design throughout the document by keyword search.

The Design Pattern is described by the following structure:

Name Design Pattern.

Type Document/Diagram.

Features Search specifications (see Table 7.16).

Weight Significance of this difference. Depending on the learning priorities the lecturer may like to set some rules are more important for searches.

7.9.2 Interaction Pattern

These are the patterns in how people interact on all tools, so use Chat Analysis with extra data from students editing strategies on other tools. These patterns are an extension of the Chat Analysis Pattern. The actions that are analysed for patterns are additions, deletions and moves in editing and the interchange of *Tokens* in Chat.

[2]Rule Checkers are applied before the alternative designs are displayed.

Patt-ern	Tool Searched	Object Found	Attribute 1 (search similar)	Attribute 2 (search dissimilar)	Content
ADP CDP	Document	Section	Name	Length	Importance of section
ADP	Document	Section	Name	Exists	Use of section
ADP	Document	Section	Integration	(Search all)	Expert view
CDP TDP	Document	View	Keyword	(Search all)	Use of views
CDP ADP	Diagram	Process	Name	Input data and output data	Different complexities
ADP	Diagram	Process	Name	(Search all)	Alternatives
ADP	Diagram	Data	Name	Split data vs. un-split	Synchronous Data
ADP CDP	Diagram	Process	Name	Number of inputs/outputs	Condense design
ADP CDP	Diagram	Process	(Search all)	Spacing of inputs/outputs	Visual design
TDP	All	Keyword	Lexicon word	Context	Use of keyword

Table 7.16: List of selection criteria for design patterns

Name	Chat Analysis	
Context	Chat Tool/Single Student	
Objective		
Subjects	Single Student	
Content	Learning in groups involves various actions by all group members	
Type	Reflection	
Initiate	Permanent	
Tool	Topics and token or sentence openers	
Conditional	If the features of a Conversation Act that support learning are not visible.	
Action		
	Conditional Action	Is the student aware of the situation? If they select yes, suggested action.
	Conditional Action	Is group aware of the situation? If one selects yes, suggested action.
	Conditional Action	Is situation gone? Reward the group/student that changes pattern with an appropriate Context Extension.
Changes	Make student aware of pattern, and verify pattern is problem.	
	Make group aware of pattern.	
	Link change of pattern to learning.	
Example	See User Chat Analysis, and Topic Chat Analysis.	
Weight	3+number of token/10+number of links. Depends on the particular analysis pattern.	
Role	May be linked to Interaction patterns.	

Table 7.17: Chat Analysis

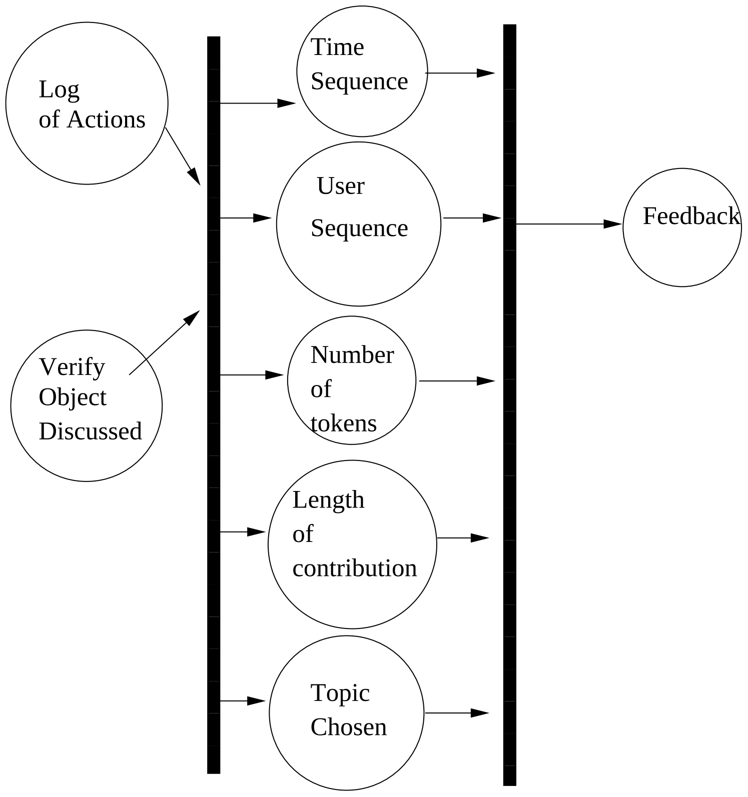


Figure 7.14: Pattern — Chat Analysis

Name	Interaction
Context	All Tools
Objective	Enable group to be aware of the type of editing and participation processes they are using
Subjects	Group
Content	Sequence of editing linked to contributions to Chat provide information about the group processes
Type	Reflection
Initiate	Permanent
Tool	Editing Pattern plus Chat Analysis Patterns
Conditional	If pattern observed
Action	Notify group that pattern has been observed
Changes	Try to remove unsuitable patterns
Example	One student adding material to a document that another student deletes
Weight	Depends on the Interaction detail
Role	extends the Chat Analysis Patterns and Editing Pattern

Table 7.18: Interaction

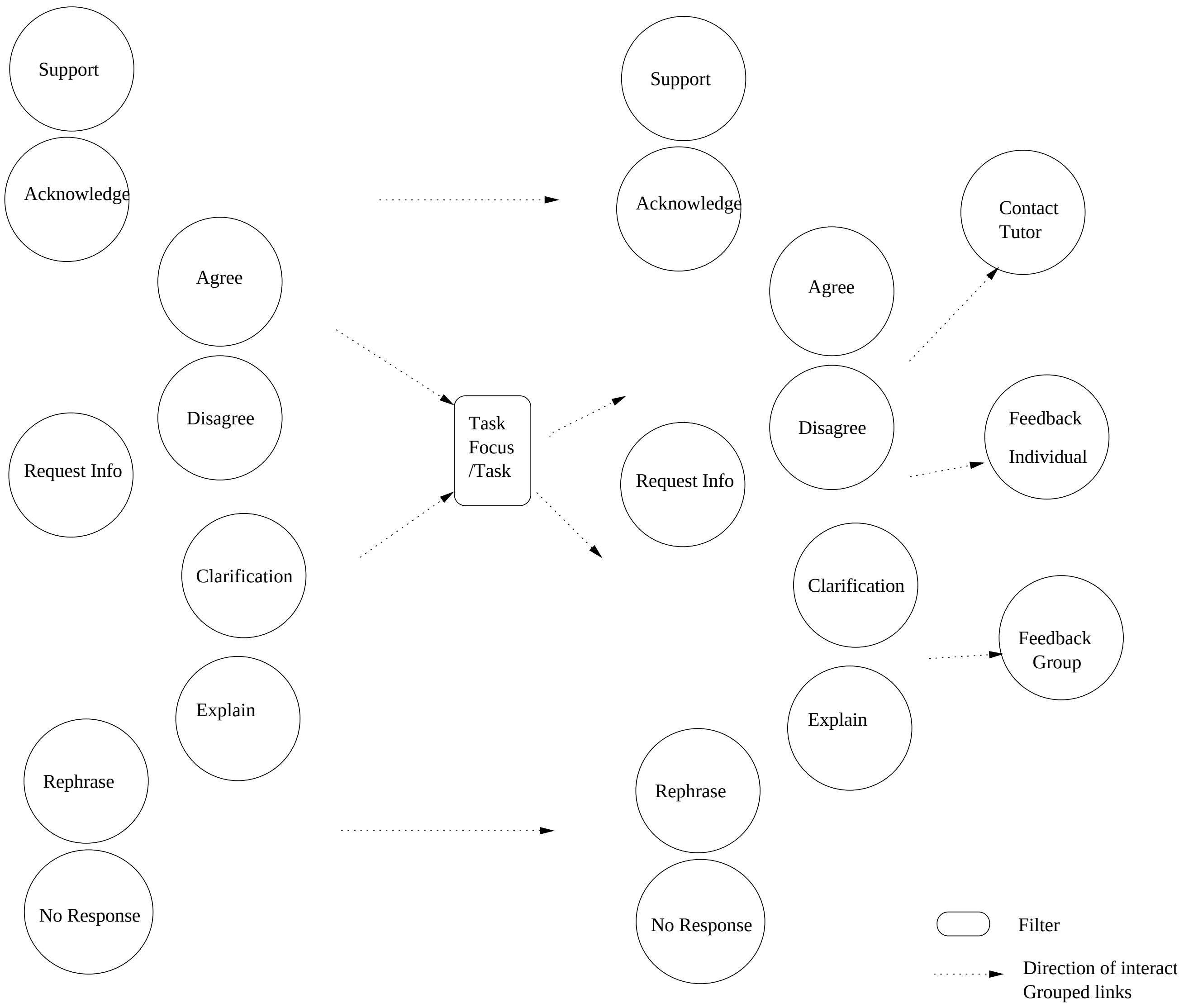


Figure 7.15: Pattern — Generic Interaction

This is a generic pattern (see Figure 7.15), so we have included examples of this pattern which are:

1. User Participation Pattern which checks if a student is involved in dialogue (via User Chat Participation Pattern) and opens their own files for group discussion.
2. User Dominance Pattern which checks if student's contribution to dialogue and document editing or opening is about average.
3. Co-operation Pattern which looks at how the group works on documents and diagrams, either editing separate tools at once, or one editing and others chatting with links to the visible part of the document or diagram made in the Chat window.
4. Sub-group Patterns where students elect to work in a smaller group either in a new session or new Chat window.

7.9.3 Learning Activity Pattern

Learning Activity Patterns are those that relate to the depth of activity in learning of a concept or an approach to learning. At present we use the stage the group is at in the course timetable to assess the present depth of activity, or to assess the knowledge that is available to the students to date. Hence, this is an assessment of the knowledge that is possibly available for synthesis into the students' own understanding.

This is a very simplistic approach and this research involved a more comprehensive analysis of patterns of the depth of approach. In accordance with the findings of Booth [45] we found that:

Depth of conception is usually attained through exposure to a greater variety of uses of the concept; and

Depth of approach usually involves the ability to develop an overview of the learning.

Hence we have developed Implementation Patterns that provide increasing complex representations or experiences of conceptions (Concept Extension Patterns) and ones that monitor bottom-up designs (Complexity Patterns in DFDs) or encourage top-down document development (View Patterns on Documents).

7.10 Conclusion

This chapter provided a basic overview of the patterns types that are extracted from the analysis of students actions presented in Chapter 6. By taking a pattern approach this research is able to cover a much broader area than any previous research into learner support on group ware. It also provides a broad basis on which the design agents which can deliver support in the multitude of process and approaches which learners use on-line.

These patterns need to now be implemented as agents in a manner which takes into account the generic aspects of each type of agent, while providing detailed analysis of student activities and their effectiveness for interaction or learning. That is, rather than simply develop agents for each pattern or sub pattern, we also have extracted common features of agents to provide a generic framework for their design.

Chapter 8

Implementation

8.1 Introduction

The pattern structure developed in the previous chapter merely provides the range of patterns which can be analysed in the groupware system. For the patterns to be implemented as scaffolding agents, they need to contain specific conditionals that specify the rules of the particular implementation of that pattern.

The development of many of these rule would be the role of the lecturer or instructor for the domain in which the student is learning. The design is novel in that it relies on the lecturer to construct rules relating to the specific focus of the course. However, some of this workload can be reduced by providing reusable rules for learning activities which are common to any domain, such as interaction in Chat, use of concept keywords, and so on. We look now at examples of providing implementation rules for the patterns for Chat Analysis.

This example is used as it mirrors the findings from previous studies by Soller and Lesgold [345] in their development of Hidden Markov Models (HMM). While this work cannot verify all the patterns, it is possible here to verify the Chat Analysis pattern against this previous work.

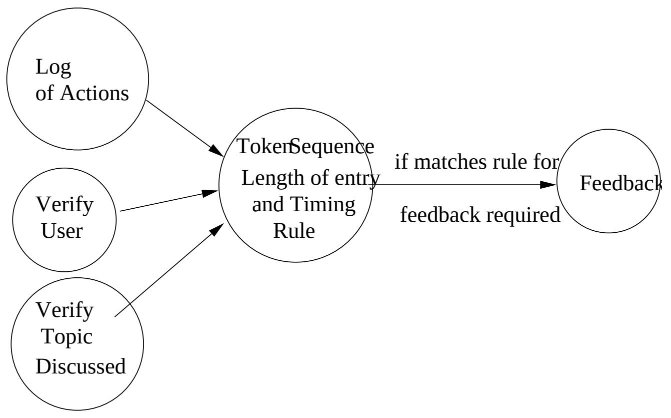


Figure 8.1: User Chat Analysis Patterns are combined through the Pattern Language

8.1.1 Chat Analysis

One pattern most commonly used in groupware analysis is the scaffolding of learning through analysis of dialogue or Chat interactions. These previous studies aim to provide feedback on the basis of the pattern of interchange in Chat. Conversation Acts that support learning which been covered in previous research are feedback, explanation effect versus acknowledgement, argumentation, questioning and participation.

In this research such patterns are included as part of the Chat Analysis Pattern, using either a subset of the tokens or all tokens in the conditional, either looking at all the group members or one User, and possibly linked to other patterns, such as the length of contributions in User Chat Explanation.

For example, the Chat Analysis Pattern can be implemented for participation analysis. The User Chat (Participation) Pattern (see Figure 8.1) is a Chat Analysis Pattern that looks for a single student and how much they contribute, by length and frequency.

The most basic User Chat Analysis pattern is an analysis of a single student's contribution to the dialogue, and the response. First we look at a simple system of six tokens for a single response (see Figure 8.2). However, we are now considering the implementation of the rule for analysis of Token sequence.

In this representation of the patterns, the arcs are numbered with the numerical value that the path will contribute (either positively or negatively) to a specific outcome state requiring a specific scaffolding. In this analysis we only know that a series of events is more likely to lead to an outcome state (in this case the state is one where the user is requiring feedback), not the exact state at any stage. To analyse the level reached towards the final state we again look at the case studies. In this example the outcome state is that a suitable answer to a question has not been achieved so help is required by the User.

This value given to each arc is the ratio of successful to unsuccessful transversals of that path. The sequence of tokens used in response to a question are extracted from the data. They form all the possible paths on the network. The sequences are then divided into those that lead to a suitable answer, and those that did not, in terms of the researchers assessment or the user's later use of the knowledge on-line.

The number of times a path is followed in an unsuccessful sequence is a positive arc. The particular sequences that lead to a satisfactory answer to the user are negative arcs(since the unsatisfactory answer is the state is sought). The unsuccessful traversals of the paths are divided by the total traversals and this value is placed on that path as the mean for that path. Any sequence of token has a value equal to the sum of the individual paths. Extra paths can be added to acknowledge the effect of poor answer mediated by later successful answer and thus the directional paths have negative value in reverse.

Note that in this case a path traversal of high value is in fact an unsuccessful learning sequence, as the outcome of the implementation is the need for scaffolding. That is, certain combinations of events in the Chat are found to be more likely to lead to students considering that they can benefit from feedback or some form of scaffolding. By finding the number of combinations of chat sequences that followed that path and lead to feedback needed, divided by the total number of chat sequences that followed that path, gives the value shown on each arc.

8.2 Data Analysis

The analysis of four chat tokens in sequence was carried out on the Chat files of five groups of student, from the third study group. They were Group 1,2,4,5,and 6. These were the groups that used Chat tokens and had a range of success in their learning. Also the learning during the particular discussion sequence had to be analysed either by interview after the session, or by reading the chat file and verifying that the text showed the students asking the question understood the concept/skill afterwards. Further analysis could be done on whether the question was about 'how to do something' rather than 'what something is', and if this effects the patterns of the response.

The files were searched for a 'Request Info' token and then the subsequent tokens collected in sequence if they were the tokens under analysis (see Figure 8.2). In this pattern we are only looking at one user, hence what is their learning level following the subsequent tokenised contributions by other students, after they request some information. There were about 40 such interchanges. 'No Response' was selected if the subsequent tokens were not one of the tokens studied.

Analysis of a sequence of tokens stopped when the topic changed (visual estimate if topic not selected), another question was asked, or five tokens had passed that did not relate to those being selected ('No Response'). Note that this level of analysis is not possible to automate by computer, but provides a rule by with such incomplete answers can be identified through tokens alone.

Soller and Lesgold found it was optimal to consider up to five subsequent contributions [345]. In this method we did not limit the number of chat contributions, and any intermediate tokens which were off the topic were ignored. This enabled us to understand the complexity of the interchange, for instance an acknowledgement may be followed by a better explanation from another user, so the urgency for feedback is reduced.

Also in the prior work there were only two involved in the interchange, one having been given the knowledge and another seeking it. This was not the set up in this work. There may be many users providing feedback, and it was assumed that the computer would have to analyse such situations where incomplete and incorrect explanation are given by different users. Hence where a path was not used more than 1/4 of the time in that particular analysis (either looking at six token in Figure 8.2, or all tokens in Figure 8.3 and Figure 8.4 as a response to a Request Information) then that path was ignored as not typical. Hence the token 'Statement' was not found to be valid in the Question Response analysis.

8.3 Chat Analysis Pattern with six tokens

The Pattern for User Chat Analysis is presented in the following table 8.1. The tables do not provide the explicit enactment rule for the conditional, this is only shown in the graph, which is a particular implementation of this pattern. Also the work in analysing the particular pattern implementations is very tedious, and simplified versions is used in the first implementation in the agent software. However some patterns are easy to implement directly from the learning goals of the course. This has been one aim of the research, to enable goals to be translated into feedback as in closed-answer tutorial systems.

For User patterns the pattern outcome in terms of requiring feedback, or value of the level of need for feedback, is retained for each student on the assumption that groups will treat different individuals differently. This Chat Analysis pattern can be

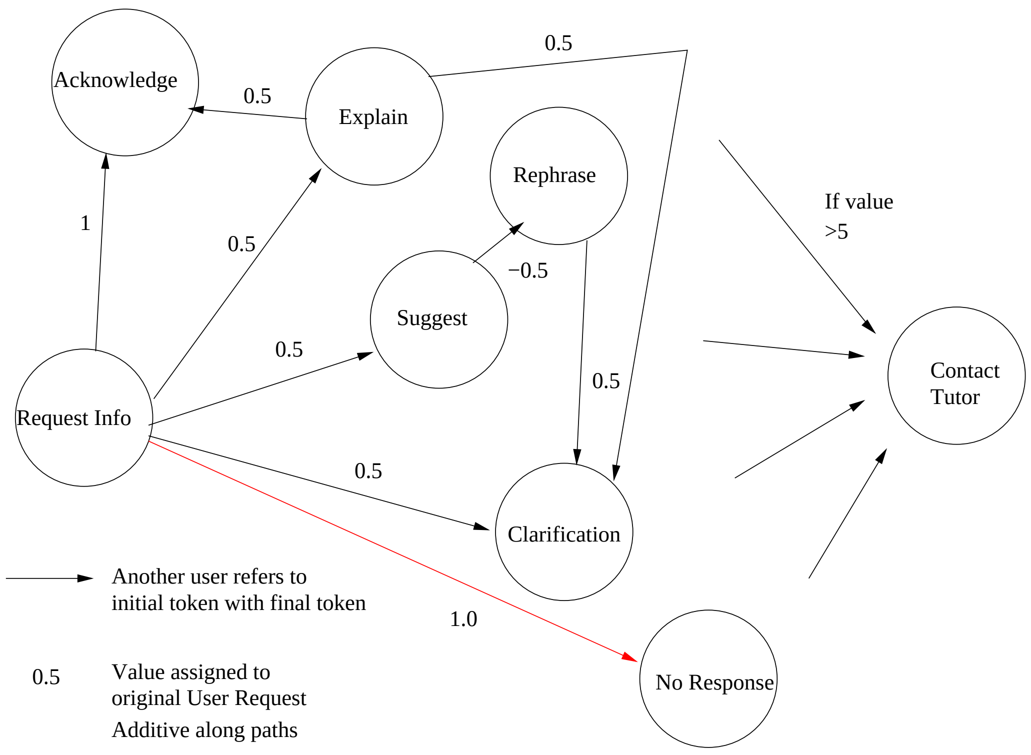


Figure 8.2: Pattern — User Chat Analysis (Six Token)

Name	User Chat Analysis
Context	Chat Tool/Single User
Objective	Alert Group to problems in their feedback to User
Subjects	Single User
Content	Groups provide the opportunity for immediate feedback to student queries.
Type	Reflective
Initiate	Permanent
Tool	Tokens and sentence openers
Conditional	If a student's questions are not being answered.
Action	Conditional: Ask the student if they are concerned they are not being answered. Action: If so offer to email summary to tutor.
	Conditional: Ask the group is they are aware that particular student is not being answered. Action: Check if answer occurs.
	Conditional: If student is answered. Action: Reward group member who answers. Give Concept Extension.
Changes	The student is aware they are not being answered.
	The group is aware they are not answering questions.
	The group member links giving explanation with learning.
Example	When the student is asking many questions and receiving no answers this may be a learning problem for that particular student. It may be that they are asking rhetorical questions. Leave the option to the student to answer. If the student selects it as a problem, then notify the group if it continues.
Weight	3.6
Role	May be linked to User Chat Participation of this student.

Table 8.1: User Chat Six Token Analysis (Question)

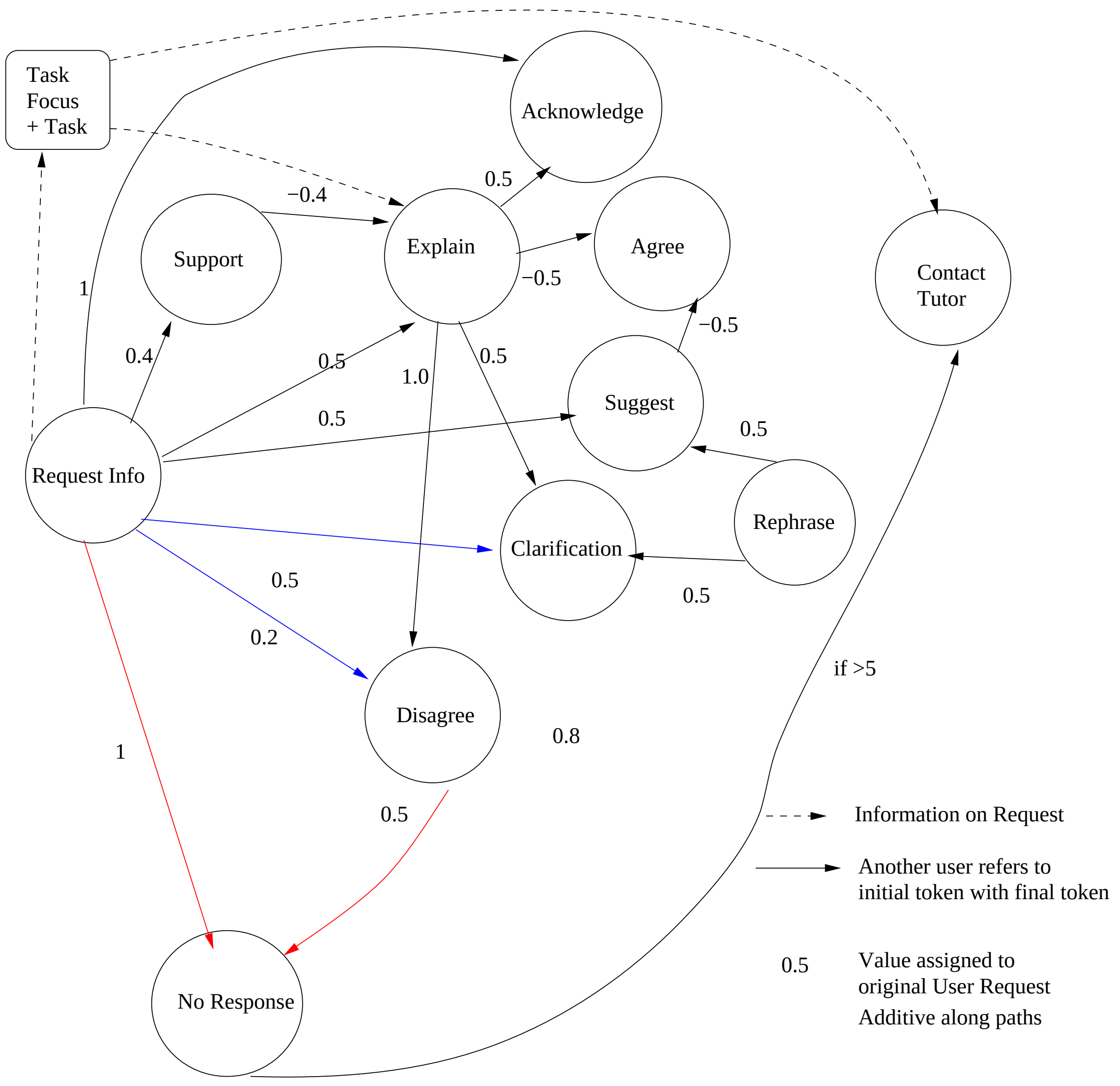


Figure 8.3: Pattern — User Chat Analysis (Question)

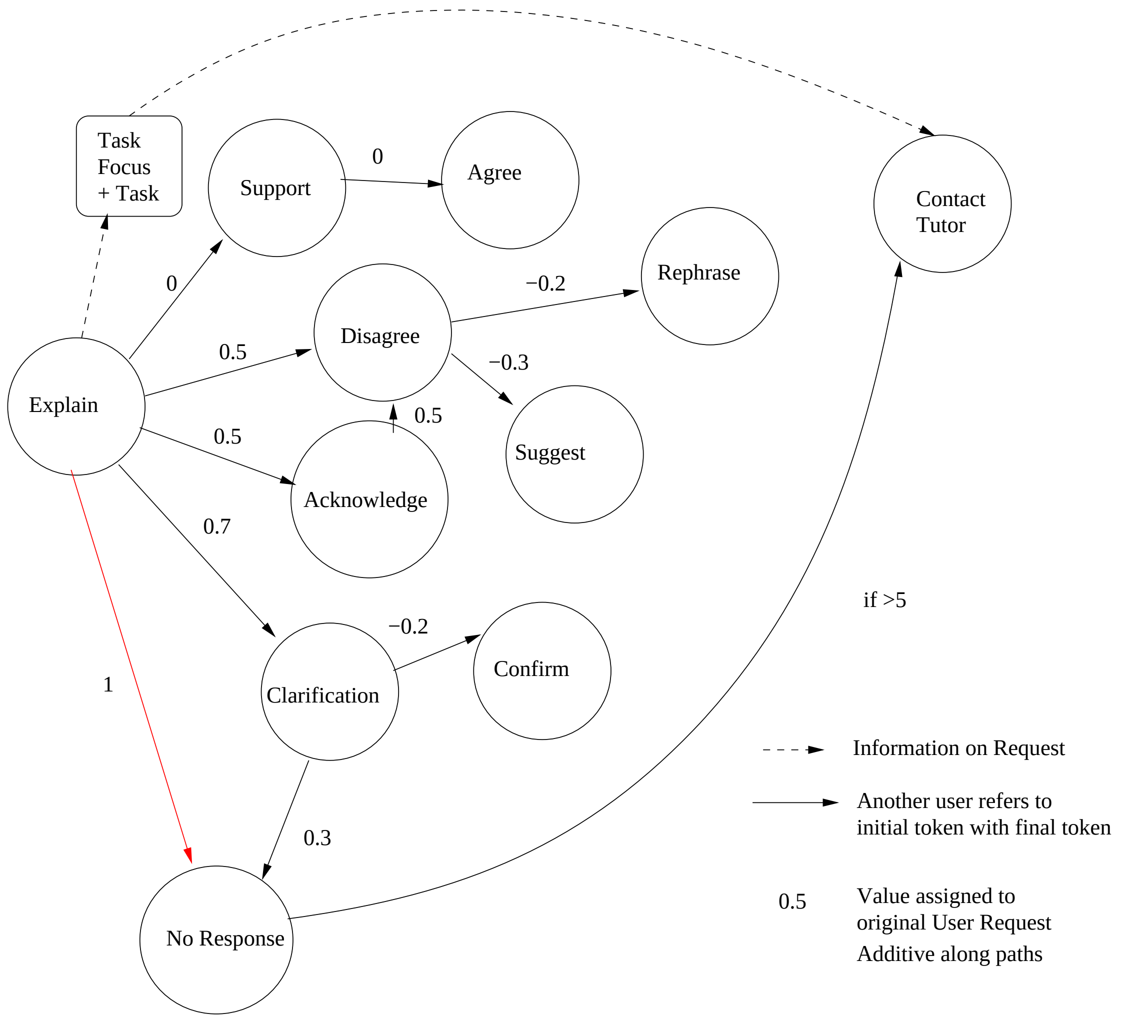


Figure 8.4: Pattern — User Chat Analysis (Explanation)

expanded to include all the tokens as part of the analysis and the feedback can be enhanced by analysing the task focus as well as any task that is being undertaken (e.g. Editing) in Figure 8.3.

The Chat Analysis Pattern (Question) (see Figure 8.3) is analysing learning outcomes which are based on effective and ineffective knowledge sharing patterns as a response to asked questions. A similar results from dialogue analysis are reported by Soller and Lesgold [345] in their development of Hidden Markov Models (HMM). In their work knowledge breakdown (requiring tutor support in the present pattern) are marked by *inaccurate or incomplete explanations* (when automated on-line this can only be analysed by length of receiver response to request for explanation), *doubt* (not used as a token in this work but comparable to disagree in how users perceived the token meaning) and *acknowledgement* of the explanation.

Since their work also showned the effective learning that arose from user initiaed explanations, we also analysed this pattern in Figure 8.4. This shows some similar sequences to those analysed by Soller and Lesgold, and listed below.

The list of sequence patterns extracted by this prior study also rely on knowledge about who is the person informed and who is requiring information in the interchange and involves only two users. This analysis was not used in this work as there were many contributors and sometimes more than one would answer. It was not always possible to verify if the person giving the explanation actually knew what they were explaining, or just supposing.. The sequences requiring feedback are:

- Sharer Propose; Sharer Explain; Receiver Acknowledge
- Sharer Propose; Sharer Explain; Receiver Confirm
- Sharer Explain; Receiver Acknowledge
- Sharer Propose; Receiver Doubt

- Receiver Request Explanation; Sharer explain poorly; (discussion stops)

The following paths did not require feedback as learning was effective:

- Receiver Request Information; Sharer Explain; Receiver Agree
- Receiver Request information; Sharer Explain; Receiver Request Clarification; Sharer Clarify
- Sharer Explain/Illustrate; Receiver Motivate/Encourage

In this previous research probabilities are not assigned to the events and this makes it hard to plan feedback based on learner's need. While the token used in the present work are not the same as used in this previous research, the general findings are the same, that acknowledgement and doubting are indications of ineffective learning and requires feedback. Also an explanation is not sufficient in itself, but may require more discussion to indicate the learning is successful. Therefore the values attached to different paths in the Executable Network are close in significance to those obtained from Soller and Lesgold's method, providing re-enforcement for their approach. In particular their method of analysis was semi-automatic, whereas this research used a manual method, which suggests an improvement for this work.

The aim in this work was to link the feedback to known outcomes at different levels, which had to be entered manually in this complex domain, however in more closed domains the HMM approach is useful. In fact by using patterns the HMM analysis can be extended to implementing agents for many different interaction types.

Alternative research by Constantino-González and Suthers [86] developed scaffolding in the form of advice on a groupware system based on sequences of Chat tokens and variation in the length of contributions between students in the group. In this prior research the advice given was then analysed by experts for suitability of feedback in the situation. Since we do not have data as to the sort of feedback provided, we do not know if this compares to our pattern. However given that the pattern analyses similar levels of learning effectiveness, the action initiating feedback would be similar.

The participation which was studied by Constantino-González and Suthers was similar to the User Explanation Pattern in this work. However this pattern extraction process was more generic and found that the number of contributions was not significant if the length was not included in the analysis also. Hence both factors are analysed together in the pattern (see Table 8.2) looking at the mean and standard deviation (std dev) of the contribution length/contribution to verify if that particular user's contributions are at variance with the group. The contribution length also consists of text length of chat or document edits and time length of diagrammatical edits.

There is no analysis here of the group as a whole, rather the group mean and variance is used as the standard by which each user is compared. The prior work mentioned a fixed value was used for all groups variance, however this would be an arbitrary figure and may vary in different groups for valid reasons. This was found in our initial study of groups on-line where some effective groups used short and succinct comments about their edits to explain them to other users, and learnt effectively this way.

In the prior work the participation balance was used to generate 48% of the advice judged by experts to be reasonable and Time on Task generated 40%. The User Explanation pattern would generate similar response to the combination of these, however there is no way to compare the balance given in this analysis to the one presented by Constantino-González and Suthers as they give no information on their Advice Selection module. Further work on the implementation patterns presented here should include a similar test of their implementation as agents.

Name	User Explanation	
Context	All Tools/Single User	
Objective	Alert User to problems in their participation in group	
Subjects	Single User	
Content	Users contribute equally to ideas and editing	
Type	Reflective	
Initiate	Permanent	
Tool	Mean Length of chat contribution and editing time	
Conditional	If Absolute(mean of group - student mean) less than Absolute(Mean of group - Group std dev)	
Action	Conditional	If std dev of user greater than group std dev
	Action	Suggest user is not contributing evenly
	Conditional	If std dev user less than group std dev
	Action	Suggest is user is contributing too much/little
Changes	The student is aware they are not contributing.	
Weight	7.0	
Role	Extension of User Chat Participation	

Table 8.2: User Explanation

8.3.1 Implementation Rules

The rules required for implementation are the conditionals on the pattern for action. The types of rules are:

Set Variable Limits e.g. participation length or frequency.

On Timing e.g. concepts linked to lecture schedule for expected weeks when they would be used and discussed

On Action e.g. when a file is opened or closed some support is provided

On Non-action e.g. when a tool is not used such as document view.

Combination of above rules e.g. rapid editing actions when deadline is nearly due.

Also the types of scaffolding tools are:

Show each User including coloured text and connection information

Change the view to show summary including minutes from chat dialogue and document template

Change view to show linkages including Text to diagram

The full formal language for the implementation rules is described in a paper submitted to the Journal of Artificial Intelligence in Education. It deals with the Activity Model application to an event based languages, which differs to state based the Situation Theory model used by Akhras and Self [11]and [8] previously.

8.4 Conclusion

This work re-iterates previous research in learning support in groupware by Constantino-González and Suthers [86], Soller and Lesgold [345], and Fjeld et al. [122], as well as learning skills that have not been approached before as possible areas for the development of agent support, such as work by Webb [387] de Jong and Mooney [96], Introne and Alterman [179], Traum and Dillenbourg [364], Wulf [396] and Entwistle [116].

The patterns provide a complete array of the possible methods of analysis of group work on an open system, and provide a means of integrating the various activities involved in document development with chat and diagram editing, into a coherent analysis system. Future work will be testing the advice given for its suitability against expert advice on the student work products.

Chapter 9

Conclusion

9.1 Summary

This thesis provides a framework for analysing student learning and interaction activity into patterns and encoding these patterns as analytical agents. By basing the research in an open flexible domain we not only tackle a new area of research, we provide a framework which is applicable to all domains. This process of translating learning needs into patterns for analysis then agents to implement this analysis, aims to provide intelligent support for students working online in groups.

9.2 Research Objectives

Constructivist learning environments encourage flexibility and discourage attempts to prescribe interactions between students. Most of the work in creating this type of learning environment is based on designing the tools required [183]. This work was done for the domain under study, that is software engineering design workshops for third year students at UNSW. The software design was based on an Activity Theory framework.

By a judicious choice of formats, it has been found by Rich and Sidner [306] that students can be encouraged to question and expand their understanding from interventions by simple intelligent agents. These agents have been developed in various domains, mainly in the area of analysing interchanges in Chat windows in distributed group mode (see Constantino [86] and Soller and Lesgold [345]) and obtaining common views of design objects in collocated groups (see Fjeld et al. [122]).

No standardised format has been developed for these agents, or a method for deriving these agents from the learning needs. Yet there is a wide range of agents that could be considered useful in many domains, and hence a re-usable part of any groupware system. In particular any learning domain involves either work patterns, rules of design or simply communication patterns.

This research presented a unified approach to all such patterns, in terms of Implementation Patterns. These patterns extend the concept of Interaction Patterns and Learning Patterns that are usually developed when studying students on-line, to look at their implementation in agents that analyse and provide feedback to students.

These patterns of learning can be developed into Patterns of analysis which provide the basis for coding the agents. Others have also provided a formal framework for such learning support (see Akhras and Self [10] and Dillenbourg and Self [98]), however this is the first such formalism that has been provided with an implementation.

In developing an implementation, the thesis first provided an outline of the agent design. The thesis showed that learning in the domain studies can be presented in terms of learning units, where the specific goals can be translated into agents that analyse learning actions and then give appropriate feedback.

Also the implementation of the formal language showed that the learning units have a pattern or structure and can be inter-related in the form of a Pattern Language. This enabled the design of a Learner Model for groups or individuals that augments the simple agent support to form a multi-agent system that responds to the learner's needs.

9.3 Research Process

The research in this thesis involved five steps:

1. Analysis of the conceptions and approaches to learning in the domain of study. These formed the learning units that were to be supported in the software. This research was based on Phenomenography and extended similar work by Booth [45] in the area of programming, and Laurillard [216] in the area of intelligent tutors, but provides new information for this research in software design.

2. Analysis of students actions in groups using a software system, to identify patterns that exemplified the comprehension of learning concepts or use of learning skills, or their lack. This was the initial stage of the Action Research into improving distance learning in groups on groupware systems. The research was based on the patterns approach to design developed by Alexander et al [12], and takes a different approach to patterns in programming developed by Gamma et al [129], in that the patterns are for implementation by agents rather than simple analysis of coding or design practises.

3. Derivation of a generic pattern structure and formal representation of the aspects of the patterns to be analysed by computer. This takes a different approach to previous research into developing groupware, in that it covers a broader scope than research into single patterns or single agent support. Similar work was done by Constantino and Suthers [87] who looked at a variety of agents, as they acknowledged the connectivity of the different aspects of learning. However the domain of study in this present research is more flexible and

hence more difficult to analyse. This restricted the support applicable to agent analysis. However the agents thus derived are generic and would assist learning in any domain, even more restrictive ones.

4. Development of an agent language to encode the agents rules derived from the patterns, into XML files for use by Java agents in the Intertac system. At present there are many systems of Java agents which have been developed for different domains and to provide different support for systems. This is another such application, however its domain is unique, in that there exists no other such agent support. Also the architecture is not common, in that the agents class is stable for each tool, but provides each different agent support through rule files.

5. Enhancement of the agents support with Learner Models and a Pattern language implemented as BDI agents. This research arose from both the patterns research and the formal language research and provide a completeness to these, developing a formal approach to the language. This is implemented using Learner Models, which have been developed for many domains (see for example the papers presented at AIED'03 Workshop on Learner Modelling for Reflection [60]).

In addition to the theoretical development, the software learning system and a selection of the agent types have been developed in a Java-based system. This will enable further testing and verification of the agents themselves, and hence the verification of the process of their development.

9.4 Research Limitations

By selecting such a large domain of learning and application, rather than focusing on Chat, or File mark up, as is done in most other research, we need to take an overview approach that assists the development of agents in any domain. This may cause some specific aspects of learning very atypical concepts or skills to be overlooked. However, a general design for agents support has been modelled and implemented.

Also because of the large scope of this work, to actually run tests on the agents, beyond their functionality as specified in their corresponding Implementation pattern, is not feasible (see Chapter 10). In other domains, such tests were completed by Soller and Lesgold [345] to verify that the agents do correctly analyse Chat interactions as successful for learning a single concept, or not, and Constantino and Suthers [87] verify that the agents do provide feedback that would be similar to that offered by a human tutor.

In considering the first approach, the pattern analysis already formally selects the chat interaction type, so it is the patterns which would have to be verified. Also the patterns analysed are not complex, as we have relied on the Learner Modelling (see Kutay and Ho [207]) to build up more complex models from the simpler agents.

In considering the second approach, there group sessions that are being conducted through Intertac are not usually monitored by tutors or mentors in collocated groups, hence there is no expertise in the department in terms of intervening in group interactions and in-time learning feedback. Hence it was proposed (Chapter 10) that the future Action Research focus more on the students experience of their learning through Intertac, than requiring comparison between the software and experts.

Finally the scope of the work only allowed an outline of the approach to be taken in agent development to be covered in the time and funding allowed. While many examples were derived and implemented, these are only a small part of the patterns of this type which may be generated for different learning domains.

Similarly the Learner Modelling is also derived as solely an outline of the appropriate approach, due to time limitations. To develop and implement the entire system is beyond the scope of the work, but the detail of the study of learning experience has enabled the researcher to provide an understanding of what is feasible in this domain, and what approach needs to be taken in providing support that is modified automatically for the learner.

9.5 Review

This research has been conducted within a large tradition of learning and software development. It has taken a Constructivist approach to learning in the development of the learning environment, and used Action research to design this environment. This approach to learning have been taken by most of the researchers in the area of groupware. In fact the main emphasis of, in particular, Social Constructivist is on providing learning opportunities in peer groups, which is the reason for the development of such groupware.

The approach taken in this thesis to understanding the students experience has been through Phenomenography and grounded research. This has provide the user requirements for the design of the agents in the form of Implementation patterns. Phenomenography has been used in many field to ascertain the learning approach students are using, the closest to this work being that by Booth [45], and Laurillard [216] and has been used by many to provide improvement in developing teaching materials for different learning needs and the teaching process (see Prosser and Trigwell [293]).

This thesis shows support for Booth's claim that depth in approach may be merely the use of an extended range of approaches rather than the acquisition of a more insightful approach.

The Pattern approach has been used in providing support for coding [129] and interface design [328]. It is a useful tool to provide a structure for a diverse range of designs, in this case agents implementation, for a range of user requirements, in this case students.

By using a patterns approach, this research finds a new approach to the analysis of students activity on CSCL (similar to the investigation of cliches suggested by Zapata-Rivera and Greer [398]) which provides a structure for designing support for the students. The patterns are used to generate automatic assistance through agents which analyse the students' activity.

To provide more strength to the link between user requirements and the agents, a formal approach was also taken to the development of learning support. Similar work has been done by Akhras and Self [7] – [11], but the present research took an Activity Theory approach rather than the Situation Theory approach used previously, and thus put more emphasis on planning and the history of the process, which can be supported by computer logging of user actions on the system and Learner Modelling.

The agent design and the multi-agent communication was designed using standard models of Rule-based agents with a Blackboard style of communication through the Group Model. This was enhanced with the Co-ordination agent with can be used to analyse and update the Group Model. This form of agent support also fits the standard of Belief–Desire–Intention or BDI agents. The depth of analysis available to such an agent is here limited to providing a summation or averaging of individuals and a logging of steps taken in more complex processes such as Decision Making. However this agent has a greater scope of applicability as the ability of analyse the depth of learning approach increases in more restricted, or less flexible, learning domains.

Chapter 10

Future Work

10.1 Introduction

10.2 Providing Scaffolding

The aim of this research is to develop a process of designing and providing scaffolding in on-line groupware. Agent support is designed to augment the basic interface and the scaffolding of the process steps and skills required by the students, so they can practise software design in an authentic but manageable environment.

The methodology for the design of this support in Intertac-II focused mainly on the architecture and the teaching aims of the software. The architectural approach used is a component-based system where agents can be plugged in. The agents can be expanded and altered as appropriate to the specific domain of learning for which Intertac is being used in future. From data collected from the use of Intertac-I and interviews with students, the researcher gained ideas of how the groups thought of their interactions and learning. These discussions are compared with how the group worked with Intertac, so their activities could be analysed for patterns in the data.

The focus of software development is always on what the student does. Teaching is about considering (see Biggs [40]):

1. what it means to understand the concepts and principles of the workshop.

2. what sort of activities are required to reach these kinds of understanding.

The first step is to develop an appropriate way to provide certain types of support. If the group model could be assumed at any time, what sort of feedback or support would be given to that group? What other factors affect the feedback besides the model? Since the learner modelling is perceived to be difficult in this unstructured domain, it is removed until the action agents, or agents that would initiate the actions of the final intelligent modelling and planning agents, are developed with inbuilt assumptions made at any instance as to the level achieved by the group.

Basic scaffolding can be provided by an advice system. To generate a simplified advice system of agents, the software analyses the student's present actions and generates the scaffolding, originally independently of the group model or detailed historical data of the group. These agents in fact implement the action part of the patterns developed from the research discussed in this work. The software must learn how to set up the required learning environment for a concept or skill, before the software can decide what to interpret the level of understanding held by the group with respect to the domain or problem at hand.

This advice system has been implemented in the form of knowledge-based agents in a prototype version of Interac-III. The research involved developing examples of such agents for the existing learning concerns raised above, and used this experience to develop a process for converting learning goals into agent support.

The process itself involves collecting the results of the Phenomenographical study of some learning activities described above to ascertain the goal for a particular learning approach. When the agents are implemented, action research can be used to verify the success of the implementation.

The advice system developed is necessarily domain-specific, except where it applies to the group's work processes. Certainly, some concern remains as to how much a learner or group modelling system can offer support for generating appropriate advice given the lack of language analysis and the large domain of learning (even if restricted to group work alone). In an open or flexible learning environment, the nature of the system ensues that it is difficult to obtain reliable information on the learner's behaviour. Thus the **bandwidth** or the amount and quality of the information available to the learner model is low, as there is limited explicit information on user traits and behaviour that the system can access.

10.3 Future Extensions

To develop agents that will provide support in various learning domains, we have provided an outline for the agent design. The first step is to develop a language and a formal approach is taken for this.

The main risk in the development of any software is that after much activity-based research into the user and technical requirements of the system, the final product may not resemble what the user and programmer decided to achieve. This is itself one of the issues that the students are confronted with in their projects, hence it is also one that the designer of the workshop or course, or the software agent support, must deal with.

The use of formal and semi-formal methods of software specification have been used to try and overcome the uncertainty in translating the users' language into code. In this context the patterns are the users' language, and the agents are the code.

The present work aims to provide a formal methods approach to developing group projects within Intertac-II and providing an appropriate superstructure to their learning. This process aims to provide a link between the instructor/lecturer and the student's learning environment that will ensure that the agents and information resources meet the requirements of the person doing the teaching.

As mentioned above, the process of moving from learning objectives to agent support is developed in this research and the present work led to the development of a formal language that unified the patterns and the agents.

An implementation language for the agents is also developed and has been implemented in Intertac-III which is not discussed here. The language used the ontology of the formal learning language developed for the agents. An interface has yet to be developed to enable lecturers to encode learning patterns into agent rules for automatic support.

10.4 Verification of System

As with the development of any software process, it is important provide verification. This thesis has presented a process to be used to develop agent support in a software environment, based on an analysis of the learning domain, including an in depth analysis of the approach and conceptions of the learners in that domain.

The next stage in this research is to verify the validity of the design and the concept of the patterns that are implemented. This verification should consider the aspects of Constructive Learning Environments that were adopted as the goals of the present work:

Scaffolding By developing Group Models that enable the tracking of students overt response to advice and the actions that follow any advice (Kutay and Ho [207]) some analysis of the scaffolding effect can be examined. While students will have immediate responses to the feedback of such agents, it is important that the patterns used by the agents to provide feedback be used to re-analysed the effect of the feedback, if any, on the students' 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 (Kutay and Ho [206]).

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. This would require self analysis. For instance the weight of agent advice should be varied on the basis of activities carried out by students after the advice has been given and if they follow the pattern the advice was assuming to be enacted.

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.

An alternative approach has been tried on the Decision Making network graph. Since the data on group decision making is hard to model as there are so many variables, some work has been done to develop network models for a decision making group. The current distribution of JUNG (or Java Universal Network/Graph Framework [190]) provides a language in which to code these models. JUNG includes implementations of a number of algorithms from graph theory, data mining, and social network analysis, such as routines for clustering, decomposition, optimisation, random graph generation, statistical analysis, and calculation of network distances, flows, and importance measures

JUNG also provides a visualisation framework that makes it easy to construct tools for the interactive exploration of network data, and model the effect of varying parameters. While this does not verify the agent support to learning, it does verify the model which we have developed of the decision making process, and therefore support, or refute, the framework for developing agents.

Finally, another approach for the development of the system would be to provide a proof mechanism for the formal language from the results of the verification experiments. However this is not the focus of this thesis.

10.5 Conclusion

The extent of the system developed and the fact that the patterns and agents are interdependent, prevents testing in the conventional sense of either software testing or testing learning materials. Given the variety of groups that will use the software, it would not be feasible to either isolate agent support and measure its sole effect on learning, or to consider the learning effect of using Intertac-I versus Intertac-II. The latter approach encounters problems of the paucity of communication through the interface, and the need for support when moving learning to distance mode.

The methodology for this verification of the work will be further Action Research, particularly in monitoring student use of the system and interviewing the users. In particular the research should consider the learning benefits for all students, whether taking a deep or surface approach, and derive these from observing their actions, interviews and their final report product, for occurrences of the use of the skills that have been encoded into the agent support.

Chapter 11

Glossary

Affordances — Originally invented by the perceptual psychologist J. J. Gibson to refer to the actionable properties between the world and an actor (a person or animal). In this work used to refer to a learning situation that can afford or provide the opportunity for certain types of learning, but not other types. This does not guarantee a student will achieve that learning when working in the environment.

Agent — a software agent is a program which is linked to a software system and runs in the background, communicating with the main software (compare human agent).

Accept used in SharedPlans to specify an act where the user agrees to a plan (see Propose).

Action — the conscious steps of an activity.

Active — refers to the particular window, in a software system with many windows open, that has been selected by the user for use at the present. All keyboard input goes to the active window.

Actor — Active subject of an action.

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.

Advice — Refers tot he feedback given to the students during their work, usually to be generated by software agents.

Animate — Term used in the B coding environment to an interface which allows you to simulate the methods of a B specification in C code before the system is written as an implementation that can be translated fully into a C program.

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).

API — Application

Approach — refers to the way in which students can see their learning and the process they use to achieve learning.

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.

Bandwidth — Used in Modelling to refer to the amount of available information for providing certainty to the the model. Used in information theory where low bandwith signals are easier to corrupt and slower to transfer a set package of information.

Billboard — Refers to agent communication where each agent writes information to a central repository from which other agents can read to obtain shared information.

Broadcaster — Part of software in Synchronous Internet or intranet systems that broadcasts changes or latest views to all clients.

BToolkit — An environment for writing, verifying and animating B code. See also Animate.

Buggy — Refers to thinking processes that are not successful for problem solving. Also can refer to rules that look for the manifestation of such processes of thinking.

Categories — General themes unifying and discriminating between conceptions of interviewees extracted from grounded research and also used to provide an outline to the interventions by Action Research.

Chat — see chat channel.

Chat Channel — a tool either in a groupware system, or stand-alone, 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.

Concept — Usually a word used in the course design to describe a term that is important for understanding the domain. Also used in Grounded Theory to describe the various areas of common concern to interviewees.

Conception — The idea that a student may have in their head of a concept, how they visage it (see Concept).

Consciousness — The level or depth of understanding of a students about the domain or one aspect of the domain of learning.

Concurrency — Describes the property of a groupware system where all users see the changes to the entities that appear on the interface more or less as they occur and in the same order (see Synchronous).

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

Constants — Used in formalism to expressed fixed relationships between variables, in this formalism they are used to represent the equation for evaluating the attributes of the Learner Model.

Context — The environment in which the learning is taking place, including prior knowledge, linked knowledge and assumed knowledge.

Conversation — The interchange of ideas that is involved in learning a new concept. Best example is the act of repeating a new idea in *your own words* to clarify if this is acceptable to others.

Completion — The end of an Activity.

Collobject A collobject is used in Habanero to describe a data type, an internal pointer to one of the Habanero tools, or an Intertac Tool.

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

CVS — Concurrent Versioning System.

Database — The agent rules form a data base of rules, some of which can be re-used in new domains. Similarly the student's files provide a database of alternative approaches, within a single domain.

Deep — See versions of depth

Depth of approach — used by Marton and Booth to classify 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.

Depth of Activity — the depth of student activity as reflected in the outcome of their learning, which is the text and graphics on the computer.

Depth of Outcome — Used by Biggs to classify the learning outcomes of students as achieving greater (or less) understanding of the domain.

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

Document — In this thesis this term refers exclusively to objects that can be represented as the Java class Document. These are text documents. Diagrams are regarded as being separate to documents, except where they are inserted as part of a document as a link.

Entities — The object that are manipulated in Intertac.

ER — ER or Entity Relationship diagrams are used in system requirements documents to describe the entities or objects (such as data) that the system will have to deal with, and how they relate to each other.

Expert — Used as the opposite to novice, one who is experienced in the skills or use of the concepts of the domain of learning.

Flaming — Abusive language used on-line, usually the result of an ill-controlled burst of temper.

Floor Control — Used in Co-operative Work software to enable one user only to lock the access to editing or commenting at any time. This remove the problem of synchronising updates.

Flow — The smooth sequencing of events in an action.

Focus — Used by agents to refer to the selected or *active* window (Participation Agent), or the topic and token of discussion (Interaction Agent), or the present step in the group plan (Planning Agent), or the process context being undertaken (BDI agent). The meaning will depend on the agent being discussed.

Frame — a structure for representing the domain knowledge or context of a knowledge-based system. Very similar to patterns used in architecture and programming.

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 thematic, or expressly explained, in the instruction (see Technical Constituents).

Gantt Chart – Visual representation of tasks and milestones for planning projects. Includes information about dependencies between tasks and timing of tasks.

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

Grounding — In conversation grounding is the process by which all participants come to a common understanding or reach a *common ground*

Group Modelling — Refers to modelling of individual users or students, in their interaction with the group, or of the group as a whole.

GUI — Graphical User Interface.

HCH — Human-Computer-Human interaction. A view of HCI for groupwork that removes the computer to the medium rather than an agent.

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).

ICQ — ICQ is a messaging service that can send asynchronous messages, check if the receiver is on-line and if they are, can be used for synchronous chats.

Ill-structured — Used in reference to problem set for students to solve where the solution is not a direct process or unique set of steps, and in fact there is not even a single solution.

Intelligent Tutoring Systems — Software systems that are designed for single users or pairs of users to step through set processes and can monitor their keyboard input for answer, errors, etc.

Interface — Refers to the screen images that are generated by software to provide an interface to the user, as in GUI and HCI.

Intertac — Generic name of the groupware developed for this thesis, in various stages.

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

Intertac-II — Augmentation of Intertac-I with agents to support various aspects of learning in the software engineering domain.

Intertac-III — Augmentation of Intertac-II with agent communication system linked by a group and user models.

Integration — One difficult aspect of reports written co-operatively is that the different parts need to maintain a consistent approach to the problem they are solving. In the case of the work being researched in this thesis, the issues is consistency in the design and how the different approaches to presenting the design in the report are integrated.

Implementation Patterns — a pattern language developed in this thesis that provides a link between concepts and skills the instructor may wish the user to learn in a particular domain, and the implementation of an agent to support or suggest these concepts and skills.

IRC — Internet Relay Chat. An instant, synchronous communication tool that works through the internet.

Language Act — a sentence in the language of communication that forms an expressive act.

Learner — user who is using a learning system program to learn (see User and Student).

Learning hierarchy — Gagné's hierarchy of learning processes, categorised in terms of learning approaches.

Learner Model — Model of user attributes in a learner system.

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)

Mal Rules — buggy rules or rules defining common errors.

Memory overload — Used for computers to refer to running out of disk space allocated to store steps in the process, but also used to refer to humans in that their brain is treated like a computer that is trying to remember all the various intermediate stages in formulating a solution to a problem, and start to forget steps or part formed results.

Meta-knowledge — Knowledge about knowledge or the acquiring or knowledge, such as the process of learning.

Microworld — Computers can form a small world in themselves, that the user becomes immersed in. This may be a virtual reality or simply the focus of the use becomes the screen and what is shown.

Mind tools — Software tools or programs that assist learnings to develop various mental skills, such as visualising a problem in a graphics tool.

Multi-agents system — When many agents are working in the same system they have to communicate. The research into multi-agent communication is used to develop the user/group models for Intertac-III

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

Objects — The thing that is learnt about or manipulated in the learning process.

Outcome — In Activity Theory, 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.

Open Source — Refers to programs written and distributed on the public domain, with often many programmers contributing to the final product.

Open Student Model — Computer data kept on student and their actions to enable an analysis of their learning from this history. If this model is open, the student may view it to reflect on their learning and possibly edit it to enable the system to provide a more accurate model.

Pattern — A structure for representing the relation between the user and an artefact, such as a building or program user.

Persistent — Refers to sessions on groupware that may be interrupted by loss of network linkage or by people going offline, yet the session remains *alive* in some form so that users can rejoin and regain the common screen information.

Phenomena — This term is used by Booth [45] to refer to all aspects of the course that is learnt, such as skills and concepts.

Plans — Used to describe segments of programs, where programming becomes the combining of these knowledge chunks or plans into a problem solution.

Pre-Design — Used to describe the stage before designing software which can include aspects of requirements elicitation and verification, developing initial conceptions of the design, etc.

Private/Privately — when working in a shared space such as groupware, the users will be sharing ideas and work, while at the same time their thoughts will remain private, and not available for analysis by the computer.

Profile — A representation of the state of a system to be initiated at start up. For instance Intertac-I can be set to start with certain windows open.

Propose — used in SharedPlans to specify an act where a plan is proposed by the user.

Records — the entities contained in the Intertac files, which are searched by the agents to verify if they satisfy the conditionals of the rules.

Return — used in Intertac agents to signify the returned list from a conditional, usually list of those entities that satisfy the conditional.

Scaffolding — used in reference to learning support in the form of templates or instructions that assist novices to learn how to approach a skill or concept. Then as the learner becomes more expert, the scaffolding may be removed, and the learning continue without it.

Single Mode or Single Student Mode — single user mode is used in reference to groupware software when it is used by one user, rather than many users at once. In this case the users are students (see Student).

Situated — used in reference to learning to describe the dependence of learning on various features of that particular learning situation, such as context, the activity undertaken and the culture and domain of learning.

Server — software running independently of groupware tools, usually on a different computer, which handles communication issues 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.

Specification Document — The report produced by students in this research is a document that specifies a commercial piece of software that in theory the students could write from the specification.

Stand-Alone — refers to applications etc. that may be used in groupware, but are also used alone as a separate program by a single user.

Student — learner studying a course, usually refers to the specific students of the Software Engineering Workshops being studied, who may or may not be learning through groupware (see Learner and User).

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.

Tagged — Information added to a file that may not be normally readable, as embedded in code that tells the editor not to display this information.

Task — Sub step of an activity.

Team Work — Refers to body of knowledge relating to people working in teams, including project management and synergy.

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

Think Aloud — Many experiments are carried out by recording humans undertaking activities while thinking aloud. Used to analyse planning in action.

Threads — generally used to refer to the collection of mail messages on one topic in email systems, but used here to also refer to parts of a document that refer to the same idea and hence the idea or thread forms a logical rather than sequential link between separate sections of the document.

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

Triangulation — A method used to check and establish validity in qualitative research studies such as evaluative research where data from many sources is used. For instance the stakeholders in the research domain should be grouped. Data from a comparable number of people from each stakeholder group then should be collected. Alternatively separate investigators can be used in a team and they compare their subjective evaluations of the data across the team.

User — generally used to refer to a single person who is using a computer program, which may or may not be a learning system, and who may or may not be a student. In the present research users are students so this latter term is used in preference, except where talking about generic research into uses of computer software and generic User Modelling (see Student).

User Pattern — refers to patterns that look at a single user's, in this case a student's, contribution to the group work.

UML Unified Modelling Language which is used to specify, visualise, design, construct and document software artifacts.

Vector — XML format used to describe a tag with its attributes.

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.

View — in software a document is stored as an electronic file. However the user can select many different ways that document can be viewed. For instance it can be viewed as plain text, or formatted text. *[*Widget] — small software tool to perform a specific task through a **GUI** interface, such as a display or a text entry screen.

Wizard — a small window produced by a software process that requires users to enter information, or select information from a list, that is used to set up a specific secondary process or interface according to user requirements. For instance a wizard for connecting to the Internet will ask the user for the data required to set up a secondary process that can then later be selected by the user to automatically connect to the Internet.

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.

XML — XML is a metalanguage used to design markup languages. The eXtensible Markup Language is an extension of HTML that uses the same format for encoding information in a document but enables information be embedded in web document that is not visible on HTML viewers yet can be read by text editors. For instance the history or versions of a document can be included in a such format (see Vector).

Bibliography

[1] Abrial J. R. The B-book: '*Assigning programs to meaning*, Cambridge University Press, 1996.

[2] Abrial J.R. Event-Based sequential program development: Application to constructing a pointer program, in *FME 2003:Formal Methods*, September 2003, 51–74, Springer.

[3] Abrial J.R. A formal (proved) approach to System Engineering: presentations and case studies. Seminar run NICTA Formal Methods Program Educational Activities. Retrieved September 2003 from http://www.cse.unsw.edu.au/ carrollm/Abrial.

[4] Achten H. An Approach for Agent-Systems in Design Support, in F. Brazier and N. Wijngaards, editors, *Workshop 7 Notes: Intelligent Agents in Design AIED–2002* July 15–17.

[5] Aiken M.W. Using a group decision support system as a teaching tool, *Journal of Computer-based Instruction*, Summer, **19**, 2, 1992, 82–85.

[6] Aiken M.W. Using a group decision support system as an instructional aim: An exploratory study, *International Journal of Instructional Media*, **19**, 4, 1992, 321–328.

[7] Akhras F.N, Self J. A process-oriented perspective on analysing learner-environment interactions in constructivist learning, *AAI/AI-ED Technical Report No.126*, *Proceedings of the 6th Brazilian Symposium on Computing in Education (SBIE'95)*, 1995, Florianopolis, Brazil, SBC.

[8] Akhras F.N, Self J. A process-sensitive learning environment architecture, in *Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96)*, Montreal, 1996, 430–438.

[9] Akhras F.N, Self J. Modelling learning as a process, *Proceedings of the 8th World Conference on Artificial Intelligence in Education (AIED'97)*, Kobe, Japan, 1997, 418–425.

[10] Akhras F.N, Self J. Modelling the Process not the Product of Learning. Unpublished manuscript, Computer Based Learning Unit, University of Leeds, 1997.

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

[12] Alexander C, Ishikawa S, Silverstein M. *A Pattern Language: towns, buildings construction*, Oxford University Press, Oxford, 1977.

[13] Alexander C. *The Timeless Way of Building*, Oxford University Press, 1979.

[14] Allen D. Bringing Problem-Based Learning to the Introductory Biology Classroom, in A. McNeal and C. D'Avanzo , editors, *Students Active Science*, 1998, Chapter 15. Retrieved January 10,2002 from http://www.saunderscollege.com/lifesci/studact/chapters/ch15.html.

[15] Ambury G. Beginning to Tutor Problem-Based Learning: A Qualitative Investigation of Andragogy in Medical Curriculum Innovation. Retrieved August 28,2002 from http://educ.queensu.ca/ amburyg/pbl-c.html.

[16] Anand P.G. *Gagne's Eclectic Behaviourism.* Class handout, Slippery Rock University College of Education, 1998. Retrieved February 1, 2002 from http://www.sru.edu/depts/educatio/psycholo/panand/gagne.htm

[17] Anderson J.R. *The Architecture of Cognition*, Cambridge, Mass. Harvard University Press, 1983.

[18] Anderson J.R, Farrell R, Sauers R. Learning to program in LISP, *Cognitive Science*, **8**, 1984, 87-129.

[19] Anderson J.R, Pirolli P, Farrell R. 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.

[20] Azouaou F, Desmoulins C, Mille D. Formalisms for an annotation-based training memory: Connecting implicit and explicit semantics, in U. Hoppe, F. Verdejo and J. Kay, editors, *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies (AIED'03)*, 2003, 374–376.

[21] The B Method for Formal Specification. Description retrieved September 1998 from http://www.afm.lsbu.ac.uk/b/

[22] Bagozzi R. The self-regulation of attitudes, intentions and behaviour, *Social Psychology Quarterly*, **27**, 4, 1992, 336–346.

[23] Baker M.J, Lund K. Flexibility Structuring in a CSCL environment, in P. Brna, A. Paiva and J.Self (editors) *Proceedings of the European Conference on Artificial Intelligence in Education Euro-AIED'96*, Lisbon, October, 1996, 401–407.

[24] Baker M, de Vries E, Lund K, Quignard, M. Computer mediated epistemic interactions for co-constructing scientific notions: Lessons learnt from a five year research programme. *Proceeding of Euro CSCL, 2001*, Retrieved 10 January 2002 from http://www.mmi.unimaas.nl/euro-cscl/Papers/13.doc.

[25] Barba R.H. Events of Instruction (Robert Gagne). Lecture Notes, San Jose State University, School of Education, 1997. Retrieved February 1,2002 from http://www.sjsu.edu/depts/it/edit186/gagne.html

[26] Barge J.K, Hirokawa R. Towards a communication competency model of group leadership, *Small Group Behaviour*, **20**, 1989, 167–189.

[27] Barnes D, Todd F. *Communications and Learning in small groups*, London, 1997, Routledge and Kegan Paul.

[28] Barros B, Verdejo M. Analysing student interaction processes in order to improve collaboration. The DEGREE approach. *IJAIED*, **11**, 2000.

[29] Bayle E, Bellamy R, Casady G, Erickson T. Finicher S, Grinter B, Gross B, Lehder D, Marmolin H, Moore B, Potts C, Skousen G, Thomas J. Putting it all together: towards a pattern language for interaction design: A CHI97 Workshop, *SIGCHI Bulletin*, **30**, 1, 1998, 17–23.

[30] Bentley R, Busbach U, Sikkel K. The architecture of the BSCW shared workspace system. *Proceedings of the ERCIM workshop on CSCW and the Web, Sankt Augustin*, February 7–9, 1996.

[31] Berelson B. *Content analysis in communications research*, 1952, Illinois, Free Press.

[32] Bermell-Garcia P, Mulet E, Vidal R. Garcia-Fernadez L, Fan I. A Multi-Agent System to Support Engineering Design, accepted paper at the *Workshop in Intelligent Agents in Design*, AID–2002, July 15–17.

[33] Berzeny C.A. Teaching interlocutor relationships in electronic classrooms, *Computers and Composition*, **16**, 1999, 229–246.

[34] Biggs, J.B. The relationship between development levels and the quality of student learning. In S. Modgil, and C. Modgil, editors, *Towards a Theory of Psychological Development*, Slough, Bucks: NFER, 1980.

[35] Biggs J.B. *Students' Approaches to Learning and Studying.* Melbourne, Australia, Australian Council for Educational Research, 1987.

[36] Biggs J.B. *Study Process Questionnaire.* Melbourne, Victoria, Australian Council for Educational Research, 1987.

[37] Biggs J.B. Approaches to learning and essay writing, in R.R. Schmeck, editor, *Learning Strategies and Learning Styles.* New York, Plenum, 1988, 185–228.

[38] Biggs J.B. What do inventories of students' learning process really measure? A theoretical review and clarification, *Brit. J. Ed. Psych* **83**, 1993, 3–19.

[39] Biggs J.B. *Student Approaches to Learning and Studying* Hawthorn, Vic: Australian Council for Educational Research 1997.

[40] Biggs J.B. What the Student Does: teaching for enhanced learning. *Higher Education Research & development*, **18**, 1, 1999, 57–75.

[41] Biggs J.B, Collis K. Evaluating the Quality of Student Learning: The SOLO Taxonomy. New York, Academic Press, 1982.

[42] Blackboard. Retrieved May 23, 2002 from http://blackboard.com.

[43] Bloom B, editor, *Taxonomy of Educational Objectives. Handbook I: Cognitive Domain.* London, Longman Group, 1956.

[44] Bogia D.P, Kaplan S.M. Flexibility and Control for Dynamic Workflows in the worlds environment, *Proceedings of the Conference on Organisation Computing Systems*, ACM Press, Milpitas, CA, Nov. 1995.

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

[46] Boud D.J. *Implementing Student Self-Assessment.* Second Edition. Sydney, HERDSA, 1991.

[47] Boud, D. Experience as the Base for Learning. *Higher Education Development & Research*, **12**, 1, 1993, 33-44.

[48] Boud D, Feletti G, editors, The Challenge of Problem-Based Learning. London: Kogan Page. Second Edition, 1997.

[49] Boulay B du, Lukin R. Modelling human teaching tactics and strategies for tutoring systems, *International Journal of Artificial Intelligence in Education*, **12**, 2001, 235-256.

[50] Bouwers A. ArgueTrack: Computer support for educational argumentation, *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 July 18-19, 1999.

[51] Bowden J.*Curriculum development for conceptual change.* (Occasional Paper 90.3). Melbourne, Educational Research and Development Unit, Royal Melbourne Institute of Technology, 1990.

[52] Bowden J, Dall'Alba G, Laurillard D, Martin E, Marton F, Masters G, Stephanou A, Walsh E. Displacement, velocity and frames of reference: Phenomenographic studies of students' understanding and some implications for teaching. *American Journal of Physics*, **60**, 1992, 262–269.

[53] Brachman R.J. On the epistemological status of semantic networks, in N.V. Findler, editor, *Associative Net-works: Representation and use of knowledge by computers*, 1979, 3–50.

[54] Bratman M. *Intention, plans and practical reason*, Harvard University Press, 1987.

[55] Brooks R. Comparative Task analysis: An alternative direction for human-computer interaction science, in J.Carroll, editor, *Designing Interaction: Psychology at the Human Computer Interface.* Cambridge: Cambridge University Press, 1991.

[56] Brown A. Metacognition, executive control, self-regulation and other more mysterious mechanisms. F. E. Weinert and R. H. Kluwe, editors, Metacognition, Motivation and Understanding, Hillsdale, Lawrence Erlbaum Associates, 1987.

[57] Brown J.S, Burton R.R. Multiple representations of knowledge for tutoring reasoning, in D.G. Bobrow and A. Collins, editors, *Representation and Understanding*, Academic Press, New York 1975, 311–349.

[58] Brown J.S, Collins A, Duguid P. Situated Cognition and the Culture of Learning. *Educational Researcher*,**18**, Jan–Feb, 1989, 32–42.

[59] BSCW. Retrieved January 15, 2000 from http://bscw.gmd.de/DownloadServer.html

[60] Bull S, Brna P, Dimitrova V, editors, AIED'03 Workshop on Learner Modelling for Reflection, in V. Aleven, U. Hoppe, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo and K. Yacef *Supplementary Proceedings (AIED'03)*, Sydney, 2003, 193–298.

[61] Bull S, McEvoy A, Reid E. Learner Models to Promote Reflection in Combined Desktop PC/Mobile Intelligent learning Environments, in V. Aleven, U. Hoppe, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo and K. Yacef *Supplementary Proceedings (AIED'03)*, Sydney, 2003, 199–208.

[62] Bundy A, Silver B, Plummer D. An analytical comparison of some rule-learning paradigms, *Artificial Intelligence* **27**, 1985, 137–181.

[63] Bunt A, Conati C. Probabilistic Student Modelling to Improve Exploratory Behaviour. *User Modelling and User-Adapted Interaction*, **13** 200, 269–309.

[64] Burton R.R. Diagnosing bugs in a simple procedural skill, in D. Sleeman, J.S. Brown, editors, *Intelligent Tutoring Systems*, Academic Press, 1982, 157–183.

[65] Burton M, Brna P, Pilkington R. Splitting the collaborative atom: How to support learning about collaboration, in B. du Boulay and R. Mizoguchi (editors), *Artificial Intelligence in Education: Knowledge and Media in Learning Systems*, Amsterdam, The Netherlands: IOS Press, 1997, 135–142.

[66] Burton R.R, Brown J.S. Toward a natural language capability for computer assisted instruction, In H, O'Neill , editor, Procedures for Instructional System Development, Academic Press, New York, 1979.

[67] Burton R.R, Brown J.S. An investigation of computer coaching for informal learning activities, in D.H. Sleeman and J.S. Brown, editors, Intelligent Tutoring Systems, London, Academic Press, 1982, 79–98.

[68] Cahn J, Brennan S. A psychological model of grounding and repair in dialog, *Proceedings of the 1999 AAAI Fall Symposium: Psychological Models of Communication in Collaborative Systems*, Cape Code, MA, 1999, 25–33.

[69] Cañas J.J, Bajo M.T, Gonzalvo G. Mental models and computer programming. International Journal of Human-Computer Studies, 40, 1994, 795–811.

[70] Carroll J, Swatman P.A. Structured-Case: A methodological framework for building theory in Information Systems research, *European Journal of Information Systems*,**9**, 4, 2000, 235—342..

[71] Chaib-Draa B, Moulin B, Mandiau R, Millot P. Trends in Distributed Artificial Intelligence. Artificial Intelligence Review, 6, 1992, 35–66.

[72] Charney D, Reder L, Kusbit G. Goal setting and procedural selection in acquiring computer skills: A comparison of tutorials, problem-solving and learner exploration. Cognitive Science, 7, 1990, 323–342.

[73] Chaib_Draa B, Moulin B, Mandiau R, Millot P. Trends in Distributed Artificial Intelligence, *Artificial Intelligence Review*, **6**, 1992, 35–66.

[74] Chen M.S, Chau C.C, Kabat W.C. Decision Support Systems: A rule-based approach, *Artificial Intelligence Information Systems*, ACM, 1985.

[75] Chi M, Van Lehn K. The content of physics self-explanations. The Journal of the Learning Sciences, 1, 1991, 69–106.

[76] Chi M.T.H, Bassok M, Lewis M.W, Reimann P, Glaser R. Self-explanations: How students study and use examples in learning to solve problems. Cognitive Science, **13**, 1989, 145–182.

[77] Churcher N, Cockburn A. An Immersion Model for Software Engineering Projects, in ACM Australasian Computer Science Conference'97, Melbourne, 2-4 July, 1997, 163–169.

[78] Cimilino L, Kay J, Miller A. Incremental student modelling and reflection by verified concept mapping. V. Aleven, U. Hoppe, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo and K. Yacef *Supplementary Proceedings (AIED'03)*, Sydney, 2003, 219–227.

[79] Clarke A.A, Smyth M.G.G. A co-operative computer based on the principles of human co-operation. International Journal of Man-Machine Studies, **38**, 1993, 3–22.

[80] Collins A, Warnock E.H, Aiello N, Miller M.L. Reasoning from incomplete knowledge, in D.G. Bobrow and A. Collins , editors,*Representation and Understanding*, Academic Press, New York, 1975, 383–415.

[81] Collins A, Brown J.S. The computer as a tool for learning through reflection, in H. Mandl and A. Lesgold (editors), Learning Issues for Intelligent Tutoring Systems, 1–18, Springer Verlag, 1988.

[82] Collins A, Brown J.S, 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.

[83] Collins A, Brown J.S. The computer as a tool for learning through reflection, in H. Mandle, and A. Lesgold, editors, Learning issues for intelligent tutoring systems. New York: Springer Verlag, 1988.

[84] Constantino-González M, Suthers D.D. A coached collaborative learning environment for Entity-Relationship Modelling, in G. Gauthier, C. Frasson and K. Van Lehn, editors, *Intelligent Tutoring Systems, Proceedings of the 5th International Conference (ITS 2000)*, Berlin:Springer–Verlag, 2000, 325–333.

[85] Constantino-González M, 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 22–24, 2001, 173–180.

[86] Constantino-González M, Suthers D.D, Icaza J.I. Coaching web-based collaborative learning based on problem solution differences and participation, in J.D.Moore, C.L. Redfield and W.L. Johnson, editors, *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future (Proceedings AI&ED 2001)*, IOS Press, 2001, 176–187.

[87] Constantino-González M, Suthers D, Santos J. Coaching Web-based Collaborative Learning based on Problem Solution Differences and Participation, *International Journal of Artificial Intelligence in Education*, 2002, **13**, to appear, Retrieved March 15 from http://www.cogs.susx.ac.uk/ijaied/.

[88] Corey S. Action research to improve school practice. New York: Teachers College, Columbia University, 1953.

[89] Cunningham D.J, Duffy T, Knuth, R. The textbook of the future, in C. McKnight, A. Dillon and J. Richardson, editors, *Hypertext — A psychological perspective*. Chichester: Ellis Horwood, 19–50, 1993.

[90] Cunningham D, Duffy T.M, Knuth R. Textbook of the future. C. McKnight, editor, Hypertext: A psychological perspective. London: Ellis Horwood Publishing, 1993, 19–50.

[91] Csikszentmihalyi M. *Flow: the psychology of optimal experience*, 1990, NY, Harper Row.

[92] Concurrent Versioning System (CVS). Retrieved August 2001, for Unix from http://www.cvshome.org, for Windows from http://www.cvshome.org/cyclic/cvs/windows.html, and as a web based versioning jCVS for Java from http://www.cvshome.org/cyclic/jcvs/index.html.

[93] Dalgarno B. Choosing Learner Activities for Specific Outcomes: A Tool for Constructivist Computer Assisted Learning Design, in C. McBeath and R. Atkinson, editors, *Planning for Progress, Partnership and Profit. Proceedings EdTech'98.* Perth: Australian Society for Educational Technology, 1998. Retrieved June 21 1998 from http://cleo.murdoch.edu.au/gen/aset/confs/edtech98/pubs/articles/abcd /dalgarno.html.

[94] Davis L. *Handbook of genetic algorithms*, New York: Van Nostrand Reinhold, 1991.

[95] De Franco-Tommarello J, Deek F. Collaborative software development: A discussion of problem solving models and groupware technologies, *Proc. 35th Hawaii International Conference on System Sciences*, 2002. Retrieved December 11 2002 from http://www.hicss.hawaii.edu/HICSS_35/HICSSpapers/PDFdocuments /CLUSR13.pdf.

[96] De Jong G, Mooney R. Explanation-based learning: An alternative view. Machine Learning, **2**, 1, 1986, 145–176.

[97] Dillenbourg P. The language shift: a mechanism for triggering metacognitive activities. M. Jones and P. H. Winnie, editors, Adaptive Learning Environments, 1992, 287–315. NATO ASI Series, **F85**, 287–315.

[98] Dillenbourg P, Self J.A. A Computational Approach to Socially Distributed Cognition.*European Journal of Psychology in Education*, **VII**, 4, 1992, 353–374.

[99] Dillenbourg P, Self J.A. A framework for learner modelling, *Interactive learning Environments*, **2**, 1992, 111–137.

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

[101] Dillenbourg P, Baker M, Blaye A, O'Malley C. The evolution of research on collaborative learning, in E. Spada and P. Reiman, editors, Learning in Humans and Machines: Towards an interdisciplinary learning science. Oxford, Elsevier, 1995.

[102] Dixon, A.S., Can Hong Kong students cope with problem-based, small-group learning?, presented at the *3rd Asia Pacific Conference on Problem Based Learning*, Yeppoon , Australia, December, 2001.

[103] Deutsch M. A theory of co-operation and competition, *Human Relations*, **2**, 1949, 129–152.

[104] Deutsch M. Co-operation and trust: some theoretical notes. *Nebraska Symposium on Motivation*, 1962, 275–230.

[105] Deutsch M. The effects of co-operation and competition upon group processes, in D Cartwright and A. Zander, editors, Group Dynamics, 461–482. New York: Harper and Row, 1968.

[106] Davis J.H. *Group Performance.* Wokingham, Addison-Wesley, 1969.

[107] Dewey J. *How we Think*, Boston, Heath, 1910.

[108] Dewey J. *Experience and education.* New York, McMillan, 1938.

[109] Dewey J. *Logic, the theory of inquiry.* New York, Holt and Co, 1938.

[110] Doise W, Mugny G. The social development of the intellect, *International Series in Experimental Social Psychology* **10** Pergamon Press, 1984.

[111] DOORS. Retrieved on July 10 2000, from http://www.qssinc.com.

[112] Eklund, J. *Cognitive Models for structuring hypermedia and implications for learning from the world-wide web*, 1995. Retrieved January 10 2000 from http://www.girardin.org/luc/cgv/report/report-7.html.

[113] Elio R, Scharf P.B. Modelling novice-to-expert shifts in problem solving strategy and knowledge organisation, *Cognitive Science*, **14**, 1990, 579–639.

[114] Ellis C, Gibbs S, Rein G. Groupware: Some issues and experiences *Communications of the ACM*, **34**, 1, 1991, 39–58.

[115] Emacs. Retrieved March 10, 2000 from http://www.gnu.org/software/emacs/.

[116] Entwistle N. *Styles of Learning and Teaching; an integrated outline of educational psychology for students, teachers and lecturers* Chichester, 1981.

[117] Entwistle N. Contrasting Perspectives on Learning, in F. Marton, D. Hounsell and N. Entwistle, editors, *The Experience of Learning.* Edinburgh, Scottish Academic Press, 1984, 1–18.

[118] Erickson J.D. Beyond Systems: Better Understanding the User's World, *Computer Language Magazine*, **10**, 3, March, 1993, 51–66.

[119] Learning Environment Software. Retrieved March 2002 from http://www.fdlearning.com/fdlearning/html/applicatio
_environment.html.

[120] Farshchian B. Shared workspace applications for collaboration in the large: A product-centered approach.

[121] Fischer K, Muller J.P, Pischel M. A programmatic BDI architecture, in *Proceedings of the IJCAI Workshop on Intelligent Agents II: Agent Theories, Architectures and languages* **1037** LNAI, 19-20 August 1996, 203–218, Berlin: Springer Verlag.

[122] Fjeld M, Lauche K, Bichsel M, Voorhorst F, Krueger H, Rauterberg M. : 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*, 2000.

[123] Flores F, Graves M, Hartfield B, Winograd T. Computer systems and the design of organisational interaction. *ACM Transactions on Office Information Systems*, **6**, 2, 1988, 153–172.

[124] Flores-Méndez R.A. Java concept maps for the learning web, *Proceeding EDMedia 97*, 1997. Retrieved January 1998 from http://pages.cpsc.ucalgary.ca/ robertof/publications/edmedia97/

[125] Flower L.S, Hayes J.R. Identifying the Organisation of Writing Processes, in Gregg, L.W, Steinberg, E.R. editors, *Cognitive Processes in Writing* . Hillsdale, NJ: Lawrence Erlbaum Associates, 1980.

[126] Foley R.P, Levy J, Russinof H.J, Lemon M.R. Planning and implementing a problem-based learning rotation for residents. Teaching and Learning in Medicine, **5**, 2, 1993, 102–106.

[127] Foss C.L. Learning from errors in Algebraland. IRL Tech. Report 3, Institute for Research on Learning, Xerox Palo Alto Research Centre, 1987.

[128] Gagné R.M. Learning Hierarchies, *Educational Psychologist*, **6**,1, 1968, 1–9.

[129] Gamma E, Johnson R, Helm R, Vlissides J. *Design patterns: Elements of reusable object-orientated software*, Addison-Wesley, 1995.

[130] Gaudioso E, Boticario J.G. Supporting personalization in virtual communities in distance education, in L.C. Jain and R.J. Howlett, editors, *Virtual Environments for Teaching and Learning*, 2002, 327–364. World Scientific Publishing Company Pty Ltd.

[131] Gaudioso E, Boticario J.G. Towards web-based adaptive learning communities, in U. Hoppe, F. Verdejo and J. Kay, editors, *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies (AIED'03)*, 2003, 237–244.

[132] Gavish B, Kalvenes J. An economic model of group decision support systems, *Proceedings of the 4th International Conference on Telecommunication Systems - Modelling and Analysis*, 1996, 319–329.

[133] Gaines B.R, Shaw M.L.G. Collaboration through concept maps. Proceedings of *Computer Supported Cooperative Learning*, Bloomington, October, 1995. Retrieved March 12, 2000 from http://ksi.cpsc.ucalgary.ca/articles/CSCL95CM.

[134] Genesereth M.R. An agent-based framework for inter-operability, In J.M. Bradshaw, ed. *Software Agents*, Menlo Park, California: AAAI Press/MIT Press, 1997, 317–345.

[135] Gianoutsos S, Grundy J.C. Qualitative Evaluation of a Collaborative Web Browser, *Proceedings of the OZCHI'96 Workshop on the Next Generation of CSCW Systems*, Hamilton, New Zealand, November 25, 1996, 40–44.

[136] Gianoutsos S, Grundy J. Collaborative work with the World Wide Web: adding CSCW support to a Web browser, *Proceedings Oz-CSCW'96*, University of Queensland, Australia, August 1996, 14–21.

[137] Gibb J.R. Climate for Trust Formation, in T-Group Theory and Laboratory Method. Wiley. New York, 1964, 279–309.

[138] Gibson J.J. The theory of affordances, in R. Shaw and J. Bransford (editors), *Perceiving, Acting and Knowing*, Hillsdale, NJ, John Wiley, 1977, 67–82.

[139] Gilmore D, Self J. The application of machine learning to intelligent tutoring systems, in J. Self, ed. *Artificial intelligence and human learning: intelligent computer-aided instruction*, Chapman and Hall, London, 1988.

[140] Glaser R, Bassok M. Learning theory and the study of instruction, *Annual Review of Psychology*, **40**, 1989, pp. 631–666.

[141] Glaser B, Strauss A. *The discovery of grounded theory*, 1967, Chicago, Illinois, Aldine.

[142] Glaserfeld E. von. Cognition, construction of knowledge, and teaching, *Synthese*, **80**, 1989, 121–140.

[143] Glaserfeld E. von. *Mind in Society: The Development of Higher Psychological Processes*, 1989, Cambridge, Mass: Harvard University Press, 1989.

[144] Glaserfeld E. von. *Radical Constructivism: A Way of Knowing and Learning*, London: The Palmer Press, 1989.

[145] Goodman B, Geier M, Haverty L, Linton F, McCready R. A framework for asynchronous collaborative learning and problem solving, in J.D. Moore et. al. editors,*Proceedings of the 10th International Conference on Artificial Intelligence in Education (AIED)*, 2001, IOS Press.

[146] Goldberg D.E. Genetic algorithms in search, optimization and machine learning. Reading, Mass: Addison-Wesley, 1989.

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

[148] Graesser A.C, Person N.K, Harter D. The Tutoring Research Group, Teaching tactics and dialog in AutoTutor, *International Journal of Artificial Intelligence in Education*, **12**, 2001, 257–279.

[149] Grosz B, Kraus S. Collaborative Plans for Complex Group Action, *Artificial Intelligence* **86**, 2, 1996, pp. 269–357.

[150] Grosz B, Kraus S. The Evolution of SharedPlans, in A. Rao and M. Wooldridge, eds. *Foundations and Theories of Rational Agencies*, pp. 227–262, 1999.

[151] Grosz B, Sidner C. Plans for discourse, in P.R. Cohen, J. Morgan and M.E. Pollack, editors, *Intentions in Communication*, 417–444, Bradford Books, Cambridge, MA.

[152] Greeno J. Trends in the Theory of Knowledge for Problem Solving. D.T. Tuma and F. Reif, editors, *Problem Solving and Education: Issues in Teaching and Research*, John Wiley and Sons, New Jersey, 1987.

[153] Grundy J.C, Mugridge W.B, Hosking J.G. Static and Dynamic Visualization of Software Architectures for Component-based Systems. *Proceedings of the 10th International Conference on Software Engineering and Knowledge Engineering*, San Francisco, June 18–20, KSI Press, 1998, 426–433.

[154] Grundy J.C, Apperley M.D, Hosking J.G, Mugridge W.B. A decentralized architecture for software process modelling and enactment, *IEEE Internet Computing*, Sept–Oct 1998, 53–58.

[155] Grundy J.C. Human interaction issues for collaborative editing components, *Proceedings of the 3rd Asia-Pacific Conference on Human-Computer Interaction*, Tokyo, Japan, July 14–17, IEEE CS Press, 1998, 145–150.

[156] Grundy J.C. Engineering component-based, user-configurable collaborative editing systems, in S. Chatty and P. Dewan, editors, *Engineering for Human-Computer Interaction*, Kluwer Academic Publishers, 1999.

[157] Guinn C. An analysis of initiative selection in collaborative task-oriented discourse, *User Modelling and User-adapted Interaction*, **8**, 3–4, 1998, 255–314.

[158] Gygi K. Recognizing the symptoms of hypertext... and what to do about it, in B. Laurel, editor, *The art of human-computer interface design*, MA: Addison-Wesley, 1990.

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

[160] Hall E, Gott S.P, Pokorny R.A. A procedural guide to cognitive task analysis: The PARI methodology. Technical Report (AL/HR-TR-1995-0108) Brooks Air Force Base, TX: Human Resources Directorate, Armstrong Laboratory, 1994.

[161] Heath M.J. Instructional Design Models for Emerging Technologies, in J. Willis, et al, editors, Technology and Teacher Education Annual, Charlottesville, VA: Association for the Advancement of Computing in Education, 1997, 459–462.

[162] Heidegger M. *Being and Time*, A translation of *Sein Und Zeit* by J. Macquarie and E. Robinson, Harper San Francisco, Revised Edition 1962.

[163] Heuelen A. van. Learning to think like a Physicist: A review of research-based instructional strategies, in textitAmerican Journal of Physics, **59**, 24, 10. 1991, 891–897.

[164] Hiltz S, Turoff M. Structuring computer-mediated communication systems to avoid information overload. *Communications of the ACM*, July, 1985, 680–689.

[165] Hiltz S. Productivity enhancements from computer-mediated communication: A system contingency approach, *Communications of the ACM*, **31**, 12, 1988, 1438–1454.

[166] Hirokawa R.Y. Discussion procedures and decision-making performance: A test of a functional perspective *Human Communications Research*, **12**, 1985, 203–224.

[167] Hirokawa R, Sheehorn D. Communication in faulty group decision-making. R. Hirokawa and M. Poole, , editors, Communication and Group Decision-Making, Sage Publication, Beverly Hills, 1986, 63–80.

[168] Hirokawa R.Y. Group communication and Decision-making performance: A continued test of the functional perspective,*Human Communication Research*, **14**, 487-515.

[169] Ho P.S, Whale, G. A Multimedia System for Reinforcing Computer Programming Skills, in Proceedings of Ed-Media, AACE, Univ. of Calgary, 14–19 June, 1997, 1239–1242.

[170] Holt P, Dubs S, Jones M, Greer J. The state of student modelling, in J. Greer and G. McCalla, editors, *Student Modelling: The Key to Individualised Knowledge-based Instruction*, Springer-Verlag, 1994, 3–35.

[171] Hoppe H, Gaßner K, Műhlenbrock M, Tewissen F. Distributed visual language environments for cooperation and learning: applications and intelligent support. *Group Decision and Negotiation*, **9**, 3, May, 2000, 205–220. Retrieved August 30, 2001 from http://collide.informatik.uni-duisburg.de/publications/gdc00.pdf.

[172] Hounsell D. Learning and essay-writing. In *The experience of learning.* Edinburgh: Scottish Academic Press, 1984.

[173] Humes A. Research on the Composing Process, *Review of Educational Research*, **53**, 2, 1983, 201–216.

[174] Hundhausen C, Douglas S. Using Visualizations to Learn Algorithms: Should Students Construct Their Own, or View and Expert's?, in 2000 IEEE Symposium on Visual Languages. Los Alamitos, CA: IEEE Computer Society Press, 2000, 21–28.

[175] Hunsberger L, Zancanaro M. A mechanism for group decision making in collaborative activity, *Proceedings of AAAI-2000*, 2000, 30–35.

[176] Husband R.W. cooperation versus solitary problem solution. Journal of Social Psychology, 11, 1940, 405–509.

[177] Inaba A. Ikeda M, Mizoguchi R. What learning patterns are effective for a learner's growth? In U. Hoppe, F. Verdejo and J. Kay, editors, *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies (AIED'03)*, 2003, 219–226.

[178] Inaba M. Internet Consultant: An Integrated Conversational Agent for Internet Exploration, 1997, Master Thesis Mitsuyuki Inaba. Retrieved April 5, 2000 from http://www.ritsumei.ac.jp/kic/ps/seminar/inabam/papers/thesis.pdf.

[179] Introne J, Alterman R. Extracting procedural knowledge from a groupware for planning system, *Proceeding of the AAAI Fall Symposium 2000*, November 3–5, 2000, Cape Cod, Mass. Retrieved March 12, 2001 from http://www.dfki.de/ bauer/fs2000/Proceedings/introne.pdf.

[180] Jack BDI agent software system written in Java, Retrieved January 17, 2002 from http://www.agent-software.com.

[181] JViews software written by J. Grundy and J. Hoskings, Retrieved March 13, 2000 from http://www.cs.aukland.ac.nz/ john-g/jviews.html.

[182] Jarke M. Knowledge sharing and negotiation support in multi-person decision support systems, *Decision Support Systems*, **2**, 1986, 93–102.

[183] Jonassen D.H *Computers in Schools: Mindtools for Critical Thinking*, Pennsylvania State University Press, 1994.

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

[185] Jonassen D. Invited Speaker, Ascilite '98. Wollongong University, 1998.

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

[187] Jonassen D.H. Tessmer M, Hannum W.H. *Task Analysis Models for Instructional Design*, New York, Erlbaum, 1999.

[188] Johnson D.W, Johnson R.T. *Learning together and alone.* Prentice Hall, Englewood Cliffs, NJ, 1991.

[189] Johnson-Laird P. *Mental models: towards a cognitive science of language, inference and consciousness*, Cambridge, MA: Harvard University Press, 1983.

[190] Java Universal Network/Graph Framework, Retrieved October 10, 2003 from http://jung.sourceforge.net/index.html.

[191] Kantschik W, Banzhaf W.F. Linear Tree GP and its comparison with other GP structures, *Proceedings of 4th EuroGP Conference, Como 2001*, Springer, Berlin, 2001, 302–312

[192] Katz S, Aronis J, Creitz C. Modelling pedagogical interactions with machine learning. *Proceedings of the Ninth International Conference on Artificial Intelligence in Education*, LeMans, France, 1999, 543–550.

[193] Katz S, O'Donell G, Kay H. An approach to coding educational dialogues for descriptive and prescriptive purposes, *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.

[194] Kay J. The UM toolkit for collaborative user modelling. Early draft of paper for UMUAI, 4, 1995, 149–196.

[195] Kay J, Halin Z, Ottomann T, Razak Z. Learner Know Thyself: Student Models To Give Learner Control And Responsibility, International Conference on Computers in Education (ICCE'97), Malaysia, AACE, 1997. 17–24.

[196] Kay J. *A scrutable user modelling shell for user-adapted interaction*, PhD Thesis, 1999, Basser Department of Computer Science, University of Sydney, Australia. Retrieved January 9, 2001 from http://www.cs.usyd.edu.au/ judy/Homec/Pubs/thesis.pdf.

[197] Kemmis S, McTaggart R. *The action research planner.* Victoria, Australia: Deakin University Press, 1982.

[198] Kinny D, Georgeff M, Rao A. A methodology and modelling technique for systems of BDI agents, in *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'93)*, **1038** of LNAI, 22–25 January 1996, Berlin:Springer Verlag, 56–71.

[199] Kolb D.A. *Experiential learning: Experience as the source of learning and development.* Englewood Cliffs, NJ. Prentice Hall, 1984.

[200] Koedinger K.R, Herson J.R, Hadley W.H, Mark M.A. Intelligent tutoring goes to school in the big city, *International Journal of Artificial Intelligence in Education*, **8**, 1997, 30–43.

[201] Kraemer K, King J. Computer-based systems for co-operative work and group decision making, *ACM Computing Surveys*, **20**, 2, 1988, 115–146.

[202] Kreifelts T. Hindrichs E, Woetzel G. BSCW–Flow: Workflow in web-based shared workspaces. Workshop on Cross-Organisational Workflow Management and Co-ordination, San Francisco, 1990.

[203] Kutay C, Ho P, Whale G. Internet Based Groups in Computer Science: Helping Groups Work. Proceedings of ENABLE '99, Helsinki:HIT, 56–68, 1999.

[204] Kutay C. Intertac Software Architecture, Technical Report No. 0318, Computer Science and Engineering, University of New South Wales, ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/0318.pdf, 2003.

[205] Kutay C, Ho P, Whale G. Achieving Learning Outcomes in HCI for Computing — An Experiential Testbed, Enable'99, Helsinki, Finland, 1999, 626–631

[206] Kutay C, Ho P. Using intelligent agents for the analysis of students' interaction and learning, *Proceeding of AIED'03, Workshop on technologies for Electronic Documents for Supporting Learning*, 2003.

[207] Kutay C, Ho P. Assisting students by analysing their interactions and learning, *Proceeding of AIED'03, Workshop on Learner Modelling for Reflection*, 2003.

[208] Lajoie S, Derry S, editors, *Computers as Cognitive Tools*, Lawrence Erlbaum Associates, New Jersey, 1993.

[209] Lajoie S.P, Faremo S, Wiseman J. Identifying human tutoring strategies for effective instruction in Internal Medicine, *International Journal of Artificial Intelligence in Education*, **12**, 2001, 293–309.

[210] Lampert M. Knowing, doing and teaching multiplication, *Cognition and Instruction*, **3**, 4, 1986, 305–342.

[211] Lander S.E, Staley S.M, Corkill D.D. *Concurrent Engineering: Research and Applications* Special Issue on the Applications of Multi-Agent Systems to Concurrent Engineering, **4**,1, March 1996, 59–72.

[212] Lahteenamki, M-L., Problem Based Learning during the first academic year, in P. Little and P. Kandlbinder, *The Power of Problem Based Learning*, Australian Problem Based Learning Network, 2001.

[213] Laughlin P.R, McGlynn R.P, Anderson J.A, Jacobson E.S. Concept attainment by individuals versus co-operative pairs as a function of memory, sex and concept rule. *Journal of Personality and Social Psychology*, **8**, 1968, 410–417.

[214] Laurillard D. The processes of student learning, *Higher Education*, John Wiley and Sons, 1979.

[215] Laurillard D. Computers and the emancipation of students: Giving control to the learner, in Paul Ramsden, editor, Improving Learning: New Perspectives, Kogan Page, London, 1988.

[216] Laurillard D. Phenomenographic Research and the Design of Diagnostic Strategies for Adaptive Tutoring Systems. M. Jones and P. Winne, editors, *Adaptive Learning Environments*, NATO ASI Series, **F85**, 1992, 233–248.

[217] Laurillard D. Rethinking university teaching : a framework for the effective use of educational technology, Routledge, New York, 1993.

[218] Lave J, Wenger E. Situated Learning: Legitimate Peripheral Participation. Cambridge, UK. Cambridge University Press, 1990.

[219] Lester J.C, Towns S.G, Fitzgerald P.J. Achieving affective impact: Visual emotive communication in lifelike pedagogical agents, *International Journal of Artificial Intelligence in Education*, **10**, 3–4, 1999, 278–291.

[220] Lester J.C, Voerman J.L, Towns S.G, Callaway C.B. Cosmo, A life-like animated pedagogical agent with deistic believability, *Applied Artifical Intelligence*, **13**, 4–5, 1999, 383–414.

[221] Lepper M.R, Wooverton M, Mumme D.L, Gurtner J.L. Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors, in S.P. Lajoie and S.J. Derry, editors, *Computers as Cognitive Tools*, Lawrence Erlbaum, Hillsdale, New Jersey, 1993, 75–105.

[222] Levin L.A, Moore J.A. Dialogue-Games: metacommunication structures for natural language interaction, *Cognitive Science*, **1**,4, 1997, 395–420.

[223] Lewin K. *Field Theory in Social Science: Selected Theoretical Papers*, D. Cartwright, editor, New York: Harper Torchbooks. 1951.

[224] Lowry N, Johnson D.W. The effects of controversy on student construction and learning. Journal of Social Psychology, 115, 1981, 31–43.

[225] Luckin R, Boulay B. du. Ecolab: The development and evaluation of a Vygotskian design framework, *International Journal of Artificial Intelligence in Education*, **10**, 2, 1999, 198–220.

[226] Lund K, Baker M.J. Teachers' collaborative interpretations of students' computer-mediated collaborative problem-solving interactions, in S.P. Lajoie and M. Vivte, editors, *Artificial Intelligence in Education, Open Learning Environments: New technologies to support learning, exploration and collaboration*, Amsterdam, IOS, 1999.

[227] McCarthy J, Hayes P.J. Some philosophical problems from the standpoint of artificial intelligence, in *Proceedings of the European Conference on Artifical Intelligence in Education (EuroAIED)*, Lisbon, Portugal, 1969, 408–414.

[228] McGregor D. *The professional manager* edited by C. McGregor and W.G. Bennis, New York:McGraw-Hill, 1967.

[229] McKnight C. Dillon A, Richardson J. *Hypertext in context*, Cambridge University Press, 1991.

[230] McLeod P.L. New Communication Technologies for Group Decision Making, in R. Hirokawa and S. Poole, editors, Communication and Group Decision-Making, Sage Publication, Beverly Hills. 1986, 426–261.

[231] McManus M.M, Aiken R.M. Monitoring computer based collaborative problem solving, *Journal of Artificial Intelligence in Education*, **6**, 4, 1995, 307–336.

[232] McManus M.M, 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.

[233] Maheshwari P. Understanding Students' Conceptions of Learning Programming. *Research report CIT-94-23*, School of Computing and Information Technology, Griffith University, 1994.

[234] Maiden N.A.M, Cisse M, Perez H, Manuel D. CREWS Validation Frames: Patterns for validating Systems Requirements, *Fourth International Workshop on Requirements Engineering: Foundation for Software Quality (RESFQ)*, Pisa, Italy, June 8th-9th, 1998.

[235] Maiden N.A.M. CREWS-SAVRE:Scenarios for acquiring and validating requirements. *Journal of Automated Software Engineering*, **5**, 4, October, 1998, 419–446.

[236] Malone T, Lepper M.R. Making Learning fun, in R. Snow and M. Farr, editors, *Aptitude, Learning and Instruction: Conative and Affective Process Analyses*, Lawrence Erlbaum, 1987.

[237] Maheshwari P. Understanding Students' Conceptions of Learning Programming. Research report CIT-94-23, School of Computing and Information Technology, Griffith University, 1994.

[238] Mandle H, Lesgold A. Learning issues for intelligent tutoring systems Springer-Verlag, 1988.

[239] Mansfield T, Kaplan S, Fitzpatrick G, Phelps T, Fitzpatrick M, Taylor R Evolving orbit: a progress report on building locales. ACM SIGGROUP, GROUP 97, Phoenix, Arizona, 1997, 241–250.

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

[241] Martin D, Rodden T, Rouncefield M, Sommerville I, Viller S. Finding Patterns in the Fieldwork, *Proceedings of ECSCW 2001*, Bonn, Germany, 2001, 39–58.

[242] Marton F, Booth S. *Learning and Awareness* in Robert Sternberg, editor, The Educational Psychology Series. Mahwah, New Jersey, Lawrence Erlbaum Associates,1 997.

[243] Marton R, Såljő R. On Qualitative Differences in Learning — 1: Outcome and Process, in textitBrit. J. Educ. Psych. **46**, 4–11, 1976.

[244] Marton R, Såljő R. On Qualitative Differences in Learning — 2: Outcome as a function of the learner's conception of the task, in textitBrit. J. Educ. Psych. **46**, 115–27, 1976.

[245] Marton F, Såljő R. Approaches to learning, in F. Marton D. Hounsell, N. Entwistle, editors, The Experience of Learning. Edinburgh: Scottish Academic Press, 1984.

[246] Marton F. Tang C, Watkins D. Discontinuities and continuities in the experience of learning: an interview study of high school students in Hong Kong. *Learning and Instruction*, **7**, 1997, 21–28.

[247] Marttunen M. Electronic mail as a pedagogical delivery system. *Research in Higher Education*,**38**(3), 1997.

[248] Marttunen M. Learning of argumentation in face-to-face and e-mail environments. *ERIC Document Reproduction Service, ED 422 791*, 1998.

[249] Marwell G, Schmitt D. Cooperation — an Experimental Analysis. London, Academic Press, 1975.

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

[251] Mayer R.E. A psychology of how novices learn computer programming, *Computer Surveys*, **1**, 1981, 121–141.

[252] Medina-Mora R, Winograd T, Flores R, Flores F. The action workflow approach to workflow management technology, *CSCW 92 Proceedings*, November, 1992.

[253] Minsky M. *The Society of Mind*, Simon and Schuster, 1988.

[254] Miflin, B., Mediating a leap of faith: lessons from the experience of preparing teachers for Problem Based Learning, in P. Little and P. Kandlbinder, *The Power of Problem Based Learning*, Australian Problem Based Learning Network, 2001.

[255] Morales R, Pain H, Bull S, Kay J, editors, Open, interactive and other overt approaches to learner modelling. International Journal of Artifical Intelligence in Education, **10**, 1999, 1070:1079.

[256] Mower D. A content analysis of student/instructor communication via computer conferencing, *Higher Education*, **32**, 1996, 217–241.

[257] Műlenbrock M, Tewissen F, Hoppe U. 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*, Amsterdam, The Netherlands: IOS Press. 1997, 191–198.

[258] Műhlenbrock M, Tewissen F, Hoppe U. A Framework System for Intelligent Support in Open Distributed Learning Environments. International Journal of Artificial Intelligence in Education, **9**, 1998, 256–274.

[259] Műlenbrock M, Hoppe U. Computer supported interaction analysis of group problem solving, in C. Hoadley and J. Roschelle, editors, *Proceedings of the Computer Support for Collaborative Learning (CSCL) Conference*, Dec. 12–15, Stanford University, Palo Alto, California. Mahweh, NJ: Lawrence Erlbaum Associates, 1999.

[260] Mullens, G. The Evaluation of Teaching in a Problem-Based Learning Context, in S.E. Chen, R.M. Cowdroy, A.J. Kingsland and M.J. Ostwald *Reflections on Problem Based Learning*, Australian Problem Based Learning Network, 1994.

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

[262] Nardi B.A. Studying Context: A comparison of activity theory, situated action models, and distributed cognition in Bonnie A. Nardi, editor, *Contexts and Consciousness: Activity Theory and Human-Computer Interaction*, MIT Press, Cambridge, Mass. 1996, 69–102.

[263] Nardi B.A. Activity Theory and Human-Computer Interaction, in Bonnie A. Nardi, editor, *Contexts and Consciousness: Activity Theory and Human-Computer Interaction*, MIT Press, Cambridge, Mass. 1996, 69–102.

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

[265] *Microsoft-NetMeeting 2.1*,(Registered Trademark), 1998, Retrieved March 12, 2002 from http://www.microsoft.com/netmeeting.

[266] Ngwenyama O.K, Bryson N, Mobolurin A. Supporting facilitation in group support systems: Techniques for analysing consensus relevant data, *Decision Support Systems*, **16**, 2, 1996, 155–168.

[267] Nisbett R, Wilson T. Telling more than we know: verbal reports on mental processes. *Psychological Review*, **84**, 1977, 231–259.

[268] Nonas E, Poulovassilis A. Optimisation of active rule agents using a genetic algorithm approach, *Proceedings DEXA '98*, Vienna, August 1998, 332–341.

[269] Norman *Learning and Memory*, San Francisco: W.H. Freeman, 1982.

[270] Nud*ist Software. Retrieved May 10 2002 from http://www.gsrinternational.com

[271] NVivo Software. Retrieved March 20, 2003 from http://www.qsrinternational.com.

[272] O'Malley C. Designing computer support for collaborative learning, in Claire O'Malley, editor, *Computer Supported Collaborative Learning*, Springer-Verlag, Berlin, 1995.

[273] O'Donnell A.M, Dansereau D.F, Hythecker V.I, Hall R.H, Skaggs L.P, Lambiotte J.G, Young M.D. Cooperative procedural learning: Effects of prompting and pre — versus distributed planning activities, *Journal of Educational Psychology*, **80**, 1988, 167–171.

[274] Ohlsson S. Some principles of intelligent tutoring, in R.W. Lawler and M. Yazdani, editors, *Artificial Intelligence and Education - Learning Environments and Tutoring Systems*, **1**, NJ: Ablex, 1987, 203–237.

[275] Ohlsson S. Learning to do and learning to understand: A lesson and a challenge for cognitive modelling, in P. Reiman and H. Spada, editors, *Learning in Humans and Machines*, Elsevier, Oxford, 1995, 37–62.

[276] Oriogun, P.K, Ferguson, A.W, Ludmann, M, Maisuria, N. and Swhe Yu, M., Using the Problem Based Learning grid in the management of a software engineering module. In P. Little and P. Kandlbinder, *The Power of Problem Based Learning*, Australian Problem Based Learning Network, 2001.

[277] Orlikowski O. CASE Tools as organisational change: Investigating incremental and radical changes in systems development. *Management Information Systems Quarterly (MISQ)* **17**, 3, September 1993.

[278] Oliver R, Herrington J. *Teaching and learning online: a guide for the beginning teacher and learner.* Mt Lawley, Perth: Edith Cowan University, 2002.

[279] Palinscar A.S, Brown A.L. Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities, *Cognition and Instruction*, **1**, 1984, 117–175.

[280] Parkes A.P. A study of problem solving activities in a Hypermedia representation, *AAI/AI-ED Technical Report No.123*, *Journal of Educational Multimedia and Hypermedia*, **3**, 2, 1994, 197–223.

[281] Pask G. *Conversation theory: Applications in education and epistemology*, Amsterdam:Elsevier, 1976.

[282] Passardiere B. de La, Dufresne A. Computer Assisted Learning, in I. Tomek, editor, *Proceedings of the 4th International Conference (ICCAL'92)*, Berlin: Springer, 1992, 555–567.

[283] Perry W.G. Forms of Intellectual and Ethical Development in the College Years: a Scheme. New York: Holt, Rinehart and Winston, 1970.

[284] Pazzani M, Billsus D. Learning and Revising User Profiles: The identification of interesting web sites. Machine Learning, 27, 1997, 313–331.

[285] Pérez-Quinõnes A, Silbert J.L. A Collaborative Model of Feedback. Human-Computer Interaction Proceedings of CHI 96, Vancouver, BC Canada, April 13–18 1996.

[286] Perkins D.N. What Constructivism Demands of the learner, *Educational Technology*, September, 1991.

[287] Piaget J. *The development of thought: Equilibrium of cognitive structures.* New York, Viking Press, 1977.

[288] Pilkington R.M. Analysing educational discourse: The discount scheme. Technical Report No. 99/2, January. Computer Based Learning Unit, University of Leeds, 1999.

[289] Pilkington R.M. Analysing educational dialogue interaction: towards models that support learning, introduction to *IJAIED Special Issue on Analysing Educational Dialogue Interaction*, **12**, 2001, 1–7.

[290] Pilkington R.M, Parker_Jones C. Interacting with computer-based simulation, *Computers and Education*, **127**,1, 1996, 1–14.

[291] Pinkwart N, Hoppe H.U, Bollen L, Fuhlrott E. Group-orientated modelling tools with heterogeneous semantics, *Intelligent Tutoring Systems*, 2002, 21–30.

[292] Porayska-Pomsta K, Mellish C, Pain H. Aspects of speech act categorisation: Towards generating teachers' language, *International Journal of Artificial Intelligence in Education*, **11**, 2000, 254–272.

[293] Prosser M, Trigwell K. Using Phenomenography in the Design of Programs for Teachers in Higher Education. *Higher Education Research & Development*, **16**, 1, 1997, 41–54.

[294] Putnam L.L. Conflict in Group Decision-Making. Randy Y. Hirokawa and Marshall Scott Poole, editors, Communication and group decision-making, Sage, Beverley Hills, 1986, 175–196.

[295] Quillian M.R. Semantic Memory, in M. Minsky, editor, *Semantic Information Processing*, Cambridge, Mass. MIT Press, 1968, 216–270.

[296] Ramsden P, editor, *Improving learning — new perspective*, London: Kegan Paul, 1988.

[297] Ramsden P. Learning to Teach in Higher Education London: Routledge, 1992.

[298] Rao A, Georgeff M. BDI agents: from theory to practice, in *Proceedings of the First International Conference on Multi–Agent Systems*,San Fransisco, CA: MIT Press, 1995, 312–319,

[299] Rational Rose Software, from IBM. Retrieved March 2001 from http://www.rational.com.

[300] Ravenscroft A, Pilkington R. Investigation by design: developing dialogue models to support reasoning and conceptual change, *International Journal of Artificial Intelligence in Education*, **11**, 2000, 273–298.

[301] Rees J, Koehler G. Brainstorming, negotiating and learning in Group Decision Support System: An evolutionary approach, *Proceedings of the 32nd Hawaii International Conference on System Science*, 1999.

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

[303] Reiter R. The frame problem in situation calculus, in V. Lifshitz (editor), *Artificial Intelligence and Mathematical Theory of Computation: Paper in Honour of John McCarthy*, Academic Press, 1991, 359–380.

[304] Restificar A.C, Ali S.S, McRoy S.W. ARGUER: Using argument schemas for argumentation detection and rebuttal in dialogs. *Proceedings of the 7th international Conference on User Modelling'99*, Banff, June 1999. Retrieved June 10, 2000 from $http://www.cs.usask.ca/UM99/Proc/short/restificar_0 30609.pdf$.

[305] Reusser K. Tutoring Systems and Pedagogical Theory: Representational Tools for Understanding, Planning and Reflection in Problem Solving, in Susanne P. Lajoie, Sharon J. Derry, editors, *Computers as Cognitive Tools*, London:Erlbaum Associates, 1993, 143–177.

[306] Rich C, Sidner C.L. COLLAGEN: When agents collaborate with people. *Automonous Agents*, 1997, 284–291.

[307] Ridgway J. Of course ICAI is impossible. Worse though it might be seditious, in J. Self, editor, *Artificial Intelligence and Human Learning*, 1988, 28–48, Chapman and Hall Computing, London.

[308] Ritchie D, Norris P, Chestnutt, G. Incorporating Technology into Problem-Based Learning. In Technology and Teacher Education Annual. Association for the Advancement of Computers in Education. 1995, 429–433.

[309] Ritter S, Koedinger K. An Architecture for Plug-In Tutor Agents. *Journal of Artificial Intelligence in Education*, **7**, 3/4, 1996, 315–347.

[310] Robertson J, Good J, Pain H. BetterBlether: A computer based educational communication tool, *International Journal of Artificial Intelligence in Education*, **9**, 1998, 219–236.

[311] Resnick L.B. Learning in School and Out. Educational Researcher, **16**, 9, 1987, 13–20.

[312] Roschelle J. What should collaborative technology be? A perspective from Dewey and situated learning, *SIGCUE Outlook*, **21**, **3**, 1992, 39–42.

[313] Roschelle J, Behrend S.D. The construction of shared knowledge in collaborative problem solving, in C. O'Malley , editor, *Computer Supported Collaborative Learning*, 1994, Berlin, Springer-Verlag.

[314] Roschelle J, Teasley S. The construction of shared knowledge in collaborative problem solving, Claire O'Malley, editor, Collaborative Problem Solving, Springer-Verlag, Berlin, 1995.

[315] Rosatelli M.C, Self J.A. A Collaborative Case Study System for Distance Learning, in *International Journal of Artificial Intelligence in Education*, **13**, 2002.

[316] Roseman M, Greenberg S. Teamrooms: Network places for collaboration. M.S. Ackerman, editor, Proceedings of the ACM 1996 Conference on Computer Supported cooperative Work, CSCW'96, ACM Press, Cambridge, Mass, 1996, 325–333.

[317] Rourke L, Person T, Garrison D.R, Archer W. Methodological issues in the content analysis of computer conference transcripts, *International Journal of Artificial Intelligence in Education*, **12**, 2001, 8–22.

[318] Sắljő R. Learning in the Learner's Perspective: 1: some commonplace misconceptions, in *Reports from the Institute of Education*, University of Gothenburg, **76**, 1979,

[319] Savery J.R, Duffy T.M. Problem based learning: an instructional model and its Constructivist framework. *Educational Technology*, **35**5, September–October, 1995, 31–38.

[320] Scardamalia M, Bereiter C, McLean R.S, Swallow J, Woodruff E. Computer supported intentional learning environments. *Journal of Educational Computing Research*, **5**, 1989, 51–68.

[321] Scardamalia M, Bereiter C, Lamon L. The CSILE Project: Trying to bring the classroom into World 3. In K. McGilly, editor, *Classroom lessons: Integrating cognitive theory and classroom practice*. Cambridge MA, MIT Press. 1994, 201–210.

[322] Schneider D. Tools for collective learning in higher education: Flow, creativity and learning, *Future of Learning Workshop*, Seville, April, 28–30, 2003.

[323] Schoderbek P.P, Schoderbek C.G, Kefalas A.G. *Management systems: Conceptual considerations* . Plano, Texas: Business Publications, 1985.

[324] Schoenfeld A.H. *Mathematical Problem Solving.* New York, Academic Press, 1985.

[325] Schőn D.A. *The Reflective Practitioner. How professionals think in action.* London, Kogan Page, 1983.

[326] Schőn D.A. *Educating the reflective practitioner: Towards a new design for teaching and learning in the professions.* San Francisco: Jossey-Bass, 1987.

[327] Schultz B, Ketrow S.M, Urban D.M. Improving Decision quality in the small group: The role of the reminder, in *Small Group Research*, November, 1995.

[328] Schűmmer T. Patterns for Groupware Interface, Retrieved January 14, 2002 from http://www.groupware-patterns.org.

[329] Self J.A. Student Models in computer-aided instruction, *International Journal of Man-Machine Studies*, **6** , 1974, 159–166.

[330] Self J.A. Concept Teaching, *Artificial Intelligence*, **9**, 1977, 202–221.

[331] Self J. Dormobile: a vehicle for metacognition, in P. Maes and D. Nardi, editors, *Meta-Level Architectures and Reflection*, Amsterdam, North Holland, 1988.

[332] Self J. The role of students models in learning environments. *Transactions of the Institute of Electronics, Information and Communication Engineers*, **E77-D**, 1, 1994, 3–8.

[333] Self J. Deconstructionist student models in the computer-based learning of science, in A. D. Ilarraza Sánchez and L.F. Castro, editors, *Computer Aided Learning and Instruction in Science and Engineering*, Third International Conference, CALISCE '96, Springer, 1996.

[334] Shaw M.E, Ackerman B, McCown N.E, Worsham A.P, Haugh L.D, Gebhardt B.M, Small P.A. Interaction patterns and facilitation of peer learning, *Small Group Behaviour*, **10**, 2, May, 1979, 214–223.

[335] Shelly A.C. and Shelly, R.K. The relationship of cognitive development to interaction inequality. *Current research in Social Psychology*, October 2004, Available December, 2004 at http://www.uiowa.edu/ grpproc/crisp/crisp.html

[336] Shuell T.J. Designing instructional computing systems for meaningful learning, in M. Jones and P. Winne, editors, *Adaptive Learning Environments*, **F85**, NY:Springer-Verlag, 1992, 19–54.

[337] Simone C. Supporting collaborative dialogues in distance learning. M. Felisa Verdejo, Stefano A. Cerri, editors, Collaborative dialogue technologies in distance learning, NATO ASI Series, **F133**, Springer-Verlag, New York, 1994, 156–169.

[338] Slavin R. E. When and why does co-operative learning increase achievement? Theoretical and empirical perspective, in *Interaction in Cooperative Groups*, R. Hertz-Lazarowitz, and N. Miller, editors, Harvard University Press, Cambridge, 1992, 145–173.

[339] Soldato T. del, Boulay B. du. Implementation of Motivational Tactics in Tutoring Systems, in Journal of Artificial Intelligence in Education, **6**, 4, 1995, 337–378.

[340] Soler J, Julian V, Rebollo M, Carrascosa C, Botti V. Towards a real-time multi-agent system architecture, *Proceeding of Autonomous Agents and Multi-Agent Systems AAMAS 2002, Workshop: Challenges in Open Agent Systems*, 2002. Retrieved 10 January 2003, from http:///www.agentcities.org/Challenge02/Proc/Papers/ch02_22_soler.pdf.

[341] Soller A, Goodman B, Linton F, Gaimari R. Promoting effective peer interactions in an intelligent collaborative learning system, *Proceedings of the 4th International Conference on Intelligent Tutoring Systems (ITS98)*, San Antonio, Texas, 1998, 186–95.

[342] Soller A, Lesgold A. Analyzing peer dialogue from an active learning perspective, *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.

[343] Soller A, Lesgold A, Linton F, Goodman B. What makes Peer Interaction Effective? Modelling Effective Communication in an Intelligent CSCL. Proceedings of the 1999 AAAI Fall Symposium: Psychological Models of Communication in Collaborative Systems, Cape Cod, MA, 1999.

[344] Soller A. Supporting social interaction in an intelligent collaborative learning system, *International Journal of Artifical Intelligence in Education*, **12**, 2001, 40–62.

[345] Soller A, Lesgold A. A computational approach to analyzing online knowledge sharing interaction. In U. Hoppe, F. Verdejo and J. Kay, editors, *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies (AIED'03)*, 2003, 253–260.

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

[347] Sommerville I, Rodden T, Sawyer P, Bentlel R, Twidale M. Integrating Ethnography into the Requirements Engineering Process, *Proceedings 1st IEEE Symposium on Requirements Engineering, IEEE Computer Society Press*, 1993, 165–173.

[348] Sommerville I. *Software Engineering.* 5th Ed, Addison-Wesley, Reading, Mass, 1996.

[349] Spiro R. J. Feltovich P. J. Jacobson, M. J, Coulson R. L. Cognitive flexibility, Constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. Educational Technology **31** 5, 1991, 22–25.

[350] Spoher J, Solowoy E. Novice mistakes: Are the folk wisdoms correct? Communications of the ACM, **29**, 7, 1986, 624–632.

[351] Stefik M, Foster G, Bobrow D, Kahn K, Lanning S, Suchman L. Beyond the Chalkboard: Computer support for collaborative and problem solving in meetings, *Communications of the ACM*, **30**, 1, 1987, 32–47.

[352] Suchman L. *Plans and Situated Actions: the Problem of Human-Machine Communication*, New York: Cambridge University Press, 1987.

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

[354] Suthers D. Analyzing learner discourse effects 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.

[355] Suthers D.D, Weiner A, Connelly J, Paolucci M. Belvedere: Engaging students in critical discussion of science and public policy issues, *Proceedings of the World Conference on Artificial Intelligence in Education (AIED'95)*, August 16–19, Washington DC: AACE, 1995, 266–273.

[356] Svensson L. Theoretical Foundations of Phenomenology. *Higher Education Research & Development*, **16**,2, 1997, 159–171.

[357] Swigger K.M, Brazile R, Shin, D. Teaching computer science students how to work together, *Proceedings CSCL '95*, Bloomington, Indiana, 1995, 356–361.

[358] Swigger K, Brazile R, Shin D, Dicksworth L. Intelligent agents for teaching people how to co-operate, Proc. ED-MEDIA/ED-TELECOM 97 World Conferences on Educational Multimedia and Hypermedia and on Educational Telecommunications [CD-ROM], Charlottesville,VA: AACE, 1997.

[359] Thorndike R.L. The effects of discussion on the correctness of group decisions when the factor of majority influence is allowed for, *Journal of Social Psychology*, **9**, 1938, 343–362.

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

[361] Teasley S. D, Rochelle J. Constructing a Joint Problem Space: The Computer as Tool for Sharing Knowledge. Susanne P. Lajoie, Sharon J. Derry, editors, *Computers as cognitive tools*, London: L. Erlbaum Associates, 1993, 229–257.

[362] Tewissen F, Baloian N, Hoppe U, Reinberg E. Matchmaker: Synchronising Objects in Replicated Software Architectures. Proceedings of 6th International Workshop on Groupware, CRIWG 2000 (to appear). Portugal: Madiera, 2000.

[363] Toulmin S. E. *The Uses of Argument*, Cambrige: Cambridge UP, 1958.

[364] Traum D.R, Dillenbourg P. Miscommunication in multi-model collaboration, *Presented at AAAI workshop on Detecting, Repairing and Preventing Human-Machine Miscommunication*, August, 1996.

[365] Trigwell K, Prosser M. Changing approaches to teaching: A relational perspective, *Studies in Higher Education*, **21**, 1996, 275–284.

[366] Twidale M.B. Knowledge acquisition for intelligent tutoring systems, in F.L. Engel, D.G. Bouwhuis, T. Bősser and G. d'Ydewalle, editors, *Cognitive modelling and interactive environments in language learning*, Springer-Verlag, Berlin, 1992, 62–71.

[367] Twidale M.B, Rodden T, Sommerville I. The Designer's Notepad: Supporting and understanding cooperative design. *Proceedings of ECSCW'93*, Milan, 1993, 93–108.

[368] Twidale M.B, Rodden T, Sommerville I. The use of a computational tool to support the refinement of ideas, *Computer and Education*, **22**, 1994, 107–118.

[369] Twidale M.B, Rodden T, Sommerville I. Developing a tool to support collaborative dialogues and graphical representation of ideas, in F. Verdejo, editor, *Collaborative Dialogue Technologies in Distance Learning*, Berlin, Springer, 1994, 219–235.

[370] Van Lehn K. Student Modelling, in M.C. Polson and J.J. Righardson, editors, *Foundations of intelligent Tutoring Systems* Hillsale, NJ:Erlbaum, 55–78.

[371] Van Lehn K. *Mind Bugs: The origins of procedural misconceptions*, Cambridge, MA, MIT Press, 1990.

[372] Van Rossum E.J, Schenck S.M. The relationship between learning conception, study strategy and learning outcome. *British Journal of Educational Psychology*, **54**, 1984, 73–83.

[373] Veerman A.L, Andriessen J.E.B, Kanselaar G. Learning through synchronous electronic discussion. *Computers and Education*, **34**, 2000, 269–290.

[374] Verdejo M.F, Barros B, Rodríguez-Artacho M. A proposal to support the design of experimental learning activities, in U. Hoppe, M. Ikeda, and H. Ogata, editors, *New technologies for Collaborative Learning*, Kluwer, 2002.

[375] Verdejo M.F, Barros B, Mayorga J.I, Read T. Including collaborative learning designs in a Learning Object Repository, in U. Hoppe, F. Verdejo and J. Kay, editors, *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies (AIED'03)*, 2003, 509–511.

[376] Virtual Network Computing (VNC) Developed by AT&T Research Cambridge University Retrieved 10/8/03 from http://www.uk.reseach.att.com/vnc/.

[377] Volet S.E, Lund C.P. Metacognitive Instruction in Introductory Computer Programming: A better explanatory construct for performance than traditional factors. *Journal of Educational Computing Research*, **10**, 4, 1994, 297–328.

[378] Voss J.F, Post T.A. On the solving of ill-structured problems, in M.T.H. Chi, R. Glaser, and M.J. Farr, editors, The nature of expertise, Lawrence Erlbaum, 1988, 261–285.

[379] Vroom V, Grant L, Cotton T. The consequences of social interaction in group problem solving, *Organisational Behaviour and Human Performance*, **4**, 1, 1969, 77–95

[380] Vygotsky L.S *Thought and Language*. Cambridge, MA:MIT Press. 1962.

[381] Vygotsky L. *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA:Harvard University Press, 1978.

[382] Waite W. verbal communication at seminar at University of Macquarie, Abstract available January 2005 http://www.comp.mq.edu.au/newsevents/seminars/2003/waite.html

[383] Wang H. Research Associate, CCMB-USC. On AERA listserve on-line discussion, 1998.

[384] Watkins D. Depth of processing the quality of learning outcomes. *Instructional Science*, **12**, 1983, 49–58.

[385] Watson R.T, Ho T.H, Raman K. S. Culture: A Fourth Dimension of Group Support Systems. Communications of the ACM, **37**, 10, October, 1994, 45-55.

[386] Webb N. Student Interaction and Small-Group Learning, in R. E. Slavin, S. Sharan, S. Kagan, R. Hertz-Lazarowitz, C. Webb and R. Schmuck, editors, *Learning to co-operate, co-operating to learn*, 1985, 147–172.

[387] Webb N.M. A model of student interaction and learning. R Hertz-Lazarowitz and N Miller, *Interaction in cooperative Groups*, Cambridge University Press, 1992.

[388] Webb G. Deconstructing deep and surface: Towards a critique of phenomenography. *Higher Education*, **33**, 1997, 195–212.

[389] WebCT software. Retrieved September 10, 2002 from http://webct.com.

[390] Wenger E. *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*, Morgan Kaufmann, California, 1987.

[391] Willis J. Alternative Instructional Design Paradigms: What's Worth Discussing and What Isn't. Ed Tech **38**, 3, 1988, 5–16.

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

[393] Wood D. Scaffolding, contingent tutoring and computer supported learning, *International Journal of Artificial Intelligence in Education*, **12**, 2001, 280–292.

[394] Woods P.J, Warren J.R. Rapid Prototyping of an Intelligent Tutorial System. Proceedings of ASCILITE'95, Melbourne, 1995.

[395] Woolf B, McDonald D.D. Context-dependent transitions in tutoring discourse. *Proceedings of AAAI'84*, 1984, 355–361.

[396] Wulf V. Why did that happen? Building appropriate mental models on groupware functions, in H.J. Bullinger and J. Zeilgler, editors, *Human-Computer Interaction: Communcaition, Cooperation and Application Design*, Lawrence Erlbaum, Mahweh, 1999, 338–342.

[397] Yin R. textitCase study research: Design and Methods, 2nd Edition, Beverley Hills, CA: Sage Publications, 1994.

[398] Zapata-Rivera J, Greer J.E. Analysing student reflection in *The Learning Game*, in *Supplementary Proceedings (AIED'03)*, 2003, 288–298.