Teachers' Concepts and Beliefs about Educational Software: A case study of teachers within a software development process.

Juan Enrique Hinostroza

Thesis submitted in fulfilment of the requirements for the Ph.D. degree of the University of London

Institute of Education, University of London

1999

ABSTRACT

Most present day educational software has been designed for use as a cognitive tool, aimed at fostering students' learning outcomes and without considering the teaching framework in which it will be used. A literature review demonstrated that there is a lack of evidence about teachers' concepts and beliefs concerning educational software. In order to address this issue a case study was designed in which teachers would need to think deeply and purposefully about the characteristics and features of software. The case chosen was a process of educational software development, in which two teachers, a software engineer, a psychologist and a graphic designer, were committed to develop a piece of software during a seven month period. In each session the teachers expressed ideas and conceptions about software and were continuously reflecting on its nature. The sessions were video-recorded and the tapes transcribed, these data were analysed using both qualitative and quantitative techniques using systemic networks to organise and give a structure to the categories of analysis.

Based on the discussion of the findings, the main implications of this study are represented as a model of understanding of educational software that considers teachers' actual concepts and beliefs about computers and software.

This model, firstly, shows, that these teachers conceived of the computer as a resource that could replace them in the role of managing students' rehearsal of materials, and, secondly, presents the characteristics of the educational software that these teachers designed and shows the dimensions of their teaching strategies (classroom atmosphere, pedagogy, and learning conceptions) that were embedded in these characteristics (human-computer interface, browsing, and interaction with the software respectively). This model demonstrates significant links between the study of Pedagogy and the study of Information Technology in Education and has implications for the relationship between these two areas of research and consequently for teacher training.

Table of Contents

Figures

Tables

I. INTRODUCTION

'Enlaces' (links in English) is a national project which forms part of the MECE programme of the Chilean Ministry of Education (for more information see: Hepp, Laval, Moënne, & Ripoll, 1996; Hepp, Rehbein, Hinostroza, Laval, Dreves, & Ripoll, 1994; Potashnik, 1996). This project has been in operation since 1993 at the Universidad de la Frontera (UFRO) in the region of La Araucanía in the South of Chile. This Program for Improvement of Educational Quality (MECE) of the Chilean Ministry of Education seeks to have a long term impact on the quality, equity, and decentralisation of Chilean education. Through the Enlaces Project, the MECE program is achieving the incorporation of modern computer technology in the country's poorest schools.

To this end, between 1993 and 1998, a computer network has been gradually installed at 3000 urban and rural primary and secondary schools. Furthermore, an Educational Computing Institute (Instituto de Informática Educativa) was established at the university (UFRO) to design educational activities, produce materials for the schools and administer the network.

Among the materials being offered to the schools on the network is educational software. This software is largely produced by the Educational Computing Institute itself due to the lack of computer software products on the market in Spanish and of software programs which meet the specific needs of these schools. The experience of introducing and using educational software in schools is calling into question, not only the use of educational software (whether available in the market or locally produced), but also the understanding of educational software and its role in schools.

Within this context, the aim of this research is to contribute to a better understanding of the concept of educational software through examining teachers' concepts and beliefs about this technology. In this sense, the present research proposes a model of teachers' understanding of educational software and computers that convey some new ideas and views about these elements. These ideas and views represent teachers' concepts and beliefs about educational software and do not correspond to their 'actions with' nor 'use of' existing educational software. This is the key difference compared with other reports (some examples are: Olson, 1988; Sandholtz, Ringstaff, & Dwyer, 1997; Schofield, 1995) that makes this piece of research a useful source for comparison and contrast of what is actually offered to teachers as 'information technology' with what they actually believe it should be.

In this general framework, the following paragraphs describe the structure and general contents of this thesis.

The literature survey presented in Chapter II, reviews the work in the areas of the design, development and evaluation of educational software; of the process of educational innovation and it also includes a brief overview of studies in the area of pedagogy, which will be used to interrogate the results of this study. The general claim presented in the literature survey is that there is a need to consider teachers' beliefs for the understanding and eventual development of educational software. The argument is based on the premise that educational software is still an arena for debate and controversy that has not yet clear design prescriptions that could ensure its effectiveness nor its use (Cuban, 1997; Johnson, Cox, & Watson, 1994; Lowther & Sullivan, 1994). If this premise is accepted, there are at least four possible explanations of this: (i) that the design of educational software is focused on learning and does not include the teaching dimension therefore teachers have difficulty using it, (ii) that the evaluation of software did not consider this issue before and is only recently moving to consider the teaching dimension, but some dimensions of it are still missing, (iii) that the incorporation of teachers into the development process has not helped to include this dimension because they were asked to contribute only in certain areas of the design. And finally (iv) that this situation is confirmed by the observations of the introduction and use of information technology in the classroom which has been generally reported as innovation processes. These processes usually demanded from the teacher that (s)he accommodate to the software being introduced because it implemented theoretical models of learning and therefore theoretical models of teaching. These four lines of argumentation suggest that there is a need for research in the area of teachers' concepts and beliefs about educational software.

Due to the lack of evidence concerning teachers' concepts and beliefs about educational software, a case study (Stake, 1994; Yin, 1994) was designed. The theoretical considerations about defining this research as case, the description of the case design process and the analysis methods used are presented in the chapter on the methodology of the study (Chapter III). In general terms, the case was a process of educational software development, in which two teachers, a software engineer, a psychologist and a graphic designer, were committed to develop a piece of educational software during a seven month time period. Each session of the development process was observed and recorded to obtain the raw research data. In each session teachers expressed ideas and conceptions about software and were continuously reflecting on educational software. The data gathered from the case were analysed from both, qualitative and quantitative stand points, using systemic networks

(Bliss, Monk, & Ogborn, 1983) to organise and give a structure to the categories of analysis and the software QSR NUD*IST was used to support the coding process.

Because of its relevance for this research, the systemic network developed is presented and discussed in Chapter IV, together with the definition process (the evolution of the network), the operational definition of its branches and leaves and the problems encountered during the analysis.

The three methods of analysis developed are presented in different chapters, these are:

- Participation analysis (Chapter V): Aimed at establishing individual participation profiles during the development process, based on the calculation of the frequencies and distributions of participation of each team member in each category.
- Sequences analysis (Chapter VI): Aimed at establishing the inter-relations among the different components of the software based on the analysis of the patterns of sequences of units in the data for the group and each team member.
- Contents analysis (Chapter VII): Aimed at looking for the meanings expressed by the teachers about the different dimensions of the piece of educational software.

These methods of analysis were used in a complementary way, enabling the researcher to focus on relevant issues and to present a triangulation of the implications drawn. The analysis is integrated in the 'Discussion and Implications' chapter (i.e. Chapter VIII) where these findings are discussed in terms of several different theoretical frameworks.

The discussion starts with a theoretical perspective that looks for understanding two dimensions of what the two teachers involved in this study believed. First, what they believed about the use of computers for teaching, and second, what they believed about the characteristics that a piece of educational software should have.

The former dimension shows that these teachers conceived of the computer as a resource that could take over some of the teacher's 'traditional' teaching routines. The latter presents the characteristics of the educational software that these teachers designed and shows the particular dimensions of their pedagogy that were embedded into these characteristics. These two dimensions together constitute what is called here a model of teachers' understanding of information technology.

In order to also analyse the way in which this model was elicited, the methods used during the research process are also discussed. This section starts by discussing the

definition of this piece of research as a case study and then it examines the way in which the data gathered were structured, that is, how the systemic network was used. Finally, the analyses of the data are discussed, encompassing the combination of both, qualitative and quantitative methods. The implications of this discussion are related to the process of construction of the systemic network and to the value added by the combination of quantitative and qualitative methods.

These three sections of the discussion converge in the last section of this chapter, which presents the implications of this study. The main implications are represented through the model of teachers' understanding of information technology that considers a strategy to use computers as an aid for the overall teaching process and provides a framework to design educational software that incorporates pedagogy related strategies into particular characteristics of the software.

The existence of this model demonstrates significant links between the study of 'Pedagogy' and the study of 'Information Technology in Education' and has implications for the relationship between these two areas of research and consequently for teacher training.

II LITERATURE SURVEY

2.1. INTRODUCTION

The use of educational software in schools is still an arena for debate and controversy. There is the software development industry with its growing market of new multimedia products evolving as fast as new hardware-technology (ABLA Learning Works, Broderbund, Microsoft Corp., Sanctuary Woods Corp, TAG Development, The Learning Company, Tom Snyder Productions, Unlimited, ZETA Multimedia, etc.)[1]. On the consumers' side, there is evidence that the role of information technology in schools is controversial (Lowther & Sullivan, 1994) or at least its effects not conclusive (Johnson, et al., 1994), and that software products that are most frequently used in school are based on drill and practice activities (Cuban, 1997; Evans-Andris, 1995). Then there are research groups producing a growing number of reports focusing on the teaching and learning processes using particular pieces of software (diSessa, Hoyles, & Noss, 1995; Laborde, 1995; Mellar, Bliss, Boohan, Ogborn, & Tompsett, 1994; Schwartz, 1996; Schwartz, Yerushalmy, & Wilson, 1993; Soloway & Pryor, 1996). These three groups use and offer different software products and have very different views as to how to assess their value.

This chapter analyses this apparent dissociation, presenting evidence and ideas that might help to identify the root of the problem. The life cycle of a piece of software is reviewed, that is, from its design and development, up to its use, evaluation and insertion in schools as an educational resource.

In order to analyse the design of software, the different classification methods that have been applied to educational software are presented, in order to see the different perspectives of analysis that could be applied to the design elements embedded in the software. Based on this a different classification is presented. The new classification is aimed at throwing light on some of the reported problems related to the lack of understanding of the role of the computer in schools and the activities of the teacher (in relation to computers). This constitutes the first line of arguments about the problems of educational software.

The development stage is analysed from two perspectives: software engineering and the reported experiences of research groups. The former presents a general view of different development methods and the latter presents some experiences of software developments reported by research groups, and a discussion of the main problems

[1] All names of companies are respective Trade Marks

identified in the literature follows. One approach to a solution to the identified problems would be to try to gain an understanding of the social and professional context in which the software will be used as part of the software development process itself. This constitutes the second line of argument.

In order to analyse the use of educational software, the way in which software evaluation has been approached is then presented. This analysis concentrates on the conception of situated actions and how this can be applied in the area of evaluation and use of educational software. So the third line of argument is the claim that research into the conceptions of educational software of teachers immersed in their practice is needed.

In the fourth section the process of the insertion of software into schools is presented within the wider perspective of educational innovation. Computers and software are often more or less consciously introduced into schools and classrooms as catalysts in order to produce a change (indeed Hawkridge, Joworosky, & McMohan, (1990) gives the catalytic rationale as one of the four most used rationales for the introduction of information technology into schools). In contrast to this view, computers are conceived here as supports for ongoing changes prompted by the needs of the educational system itself, rather than as causes of changes.

In summary, it is argued that there is a need for research into the concept of educational software from a situated perspective, and in particular for research into the understanding of the role of educational software that a teacher has and how (s)he conceptualises such a product.

2.2. EDUCATIONAL SOFTWARE DESIGN

The task of designing software can be analysed from different points of view:

- from the methodological perspective, that is tools and techniques to design software (object oriented design, structured design, top down design, bottom up design, etc.),
- from the design of different elements of software (human-computer interface, contents, functionality, etc.) and
- from the intentions of the software's author(s), that is, looking at the complete product as it has been conceived.

In this review the design of educational software is analysed from the perspective of the intentions of the author, that is from the underlying teaching and learning princi-

ples that can be found in the software. Understanding the difficulty of knowing the 'real' intentions of the author, it focuses on the explicit elements of the design.

2.2.1 Educational software classification

'Learning with Software' (Learning, 1995) presents an overview of the different attempts to classify software. The same organisation, including additional references, is used here.

- **By subject**:
  This type of classification is based on school subjects. For example, all those software programs that can be applied to, say, History. It is not a useful system when discussing principles of educational computing but it is helpful when simply describing resources appropriate to specific content areas taught in schools.

- **By software type**
  With the following categories:

  - **Computer as tutor**: To function as a tutor in a specific subject area, the computer must be programmed by an expert in order to provide a 'surrogate teacher' to the user. In the context of use, the computer (as if it was an expert) presents the student with some subject-matter content together with a set of questions or directions; the student responds and the computer completes the learning cycle by evaluating the response, and from the results of the evaluation determines what to present next.
  - **Computer as tool**: In order to function as a tool, the computer needs to run generic software applications, such as a: word processor, spreadsheet, data base software, etc.
  - **Computer as tutee**: To function as a tutee the computer provides an environment in which the user can 'teach' the computer through expressing their own ideas and solutions to problems. In order to teach the computer, the user must learn to program, to talk to the computer in a language it understands.

  This classification was initially proposed by Taylor (1980) and is mentioned by Squires & McDougall (1994). A similar classification using the criteria of 'type of software' was used by Laurillard (1990), she starts from two teaching and learning models (i.e. didactic and communication models) and does an analysis of tutorial, simulation and intelligent tutorial software based on the degree of control that the user (student) has over the following components of the software design:

- learning strategy built into the software,
- the manipulation of learning content and
- the description of learning content.

Another approach to classification of software fitting into this category is given by Chandler (1984):

- Tutorial
- Game
- Simulation games
- Experimentational simulation
- Content free tools
- Programming languages

- **By educational paradigm**
  This classification comprises four paradigms:

  - **Instructional**, e.g. drill and practice software, associated with a behavioural perspective.
  - **Revelatory**, e.g. simulations, associated with discovery or experiential learning.
  - **Conjectural**, e.g. programming, associated with the application of constructivism and other cognitive views of learning to software development and use.
  - **Emancipatory,** e.g. word processing, associated with reducing workload, so that teaching and learning can take place free of the time consuming processing of data.

  This classification was initially proposed by Kemmis, Atkin, & Wright (1977). It is also mentioned by Squires & McDougall (1994) and by Anderson, Tolmie, McAteer, & Demissie (1993) who propose a simplification of this classification based on the twin ideas of interaction around the computer and interaction with the computer. Crook (1987) in a similar vein proposes the two categories of closed (low in user control) and open (high in user control).

  Another classification that matches this group was proposed by Laurillard (1993). She starts from a definition of the learning process:

> The learning process must be constituted as a dialogue between teacher and student, operating at the level of descriptions of actions in the world, recognising the second-order character of academic knowledge.
>
> (Laurillard 1993, p. 94)

Within this framework she proposes a set of principles to define a teaching strategy that are then used to classify educational media, including software. She defines the categories as:

> ***Discursive***: both teacher's and student's conceptions are accessible to the other and both topic and task goals can be negotiable; students must be able to act on, generate and receive feedback on descriptions appropriate to the topic goal; the teacher must be able to reflect on student's actions and descriptions and adjust their own descriptions to be more meaningful to the student.
> ***Adaptive*** (by teacher): the teacher can use the relationship between their own and the student's conception to determine the task goals for the continuing dialogue, in the light of the topic goals and previous interactions.
> ***Interactive*** (at the level of actions): the student can act to achieve the task goal; they should receive meaningful intrinsic feedback on their actions that relate to the nature of the task goal; something in the 'world' must change observably as a result of their actions.
> ***Reflexive***: teachers must support the process by which students link the feedback on their actions to the topic goal, i.e. link experience to descriptions of experience; the pace of the learning process must be controllable by the students, so that they can take the time needed for reflection when it is appropriate.
>
> (Laurillard 1993, p. 100)

In this sense she is using a particular educational paradigm (her model of university teaching) to classify educational media and in particular, educational software.

This classification was criticised by Bostock (1996), who argues that it is too broad and therefore too simple to map neatly onto the more complex educational processes. He argues that:

> Laurillard's attempt may be as good as any is likely to be but this review found significant gaps in the fit [between the teaching and learning model and the media classification model]. DAIR [Discursive, Adaptive, ...] may be useful in the instructional design process as a checklist of the characteristics of the learning processes which need to be supported.
>
> (Bostock 1996, p. 82; our brackets)

Through his argument, Bostock (1996) emphasises the need of a pedagogical classification of media that could have a better 'fit' with a classification of learning activities, but he does not give further answers to this.

- **By use**

Regarding this category, Fatouros, Downes, & Blackwell, (1994) offer a classification of CAL based on software use, with specific attention given to the role of the teacher in determining how a software item might be deployed to engage learning, in young children in particular. They say:

> Whether the software is incorporated within a learning centre for children to explore freely or it is implemented as a planned activity within a unit of work, teacher decisions about implementation have a significant impact on the learning processes that are generated. For example, language and social skills may be developed through the use of a typical drill and practice program if the activity is implemented in such a way as to promote discussion and negotiation between a pair of children using the software, rather than having an individual child use the software as a means of knowledge testing
>
> (Fatouros, Downes, & Blackwell 1994, p. 186).

These authors offer a classification system based on the key domains or learning areas that teachers plan for young children to explore:

- images
- sounds
- text
- stories and ideas
- facts and figures
- consequences

These areas broadly define domains that span developmental areas and curriculum areas for early childhood education; and they emphasise the integrated nature of young children's learning.

This kind of classification was also used by Watson (1993), he focused on the 'Educational Activity' that is supported by the software, he uses different categories but the underlying principle is similar. He classifies the software in this way:

Educational Activity	The contribution of IT
Information Gathering	Electronic mail, Expert Systems, Data Bases, CD-ROM
Analysis and Evaluation	Spreadsheets, Databases, Modelling software
Presentation	Word Processing, Desk-top publishing, Desktop presentations, Multimedia, Teletext and viewdata, Graphics

Also Self (1985) classifies software based on the role it should have in the school. He includes the following categories:

- Engaging motivation
- Providing new stimuli
- Activating pupil response
- Giving information
- Encouraging practice
- Sequencing learning
- Providing resource

- **By impulses to learn**

  This category is based on a taxonomy proposed by Bruce (1997) where he provides an exhaustive characterisation of kinds of educational resources (including software). He uses the ways in which they support integrated, inquiry-based learning as the key for classification. He defines four broad categories:

  - Enquiry
  - Communication
  - Construction
  - Expression

Summarising the different ways in which a classification for software has been attempted, it was found that classifications were anchored on:

- The contents and subjects (i.e. by subject),
- The functionality build-in to the software (i.e. by type)
- The learning paradigm that is embedded in the software (i.e. by educational paradigm)
- The teaching strategy that could be triggered by the software or is embedded in the software design (i.e. by use)
- The relation a user can build-up with the software or the educational need the software is attempt to fulfil (i.e. by impulse of learn).

Each classification serves a purpose of analysis and comparison and each could well be used for one or another specific aim. For example, if the aim is to build a library of software to be consulted by teachers, the subject classification could be used, if the aim is to compare the effects of software on students' performance, then the educational paradigm could be used.

None of these classification schemes is really intended to address the issue of software design from a perspective that incorporates the classroom situation, including the teacher, the students and the school. Looking at the design of software

from such a perspective, an alternative classification is proposed in this study, which comprises three groups:

- The first group includes educational software which has been designed focusing on the end-user as a stand alone student (or group of students), examples of these kinds of products are Intelligent Tutoring Systems, Computer Aided Learning, Problem Solvers, Modelling Software, Authoring Environments. It is possible to call the software included in this group **learning centred software**. The common characteristics of this group are that it is assumed that the software will be used by students either in individual or group based activities; and that the software has a learning theory built into it, together with a set of propositions about how it should be used for learning derived from that theory.

- The second group corresponds to the software which has been designed focusing on classroom activities. That is software that includes a specific teaching method in its design and therefore it has been conceived as a **teaching centred software**. Examples of software of this group are discussion organisers, group activity based software, presentation tools, etc. This group has a didactic proposition embedded in the software, that has been used as a paradigm for the design rather than as a set of guidelines for using the software.

- The third group of software corresponds to software that has been designed as a general tool or resource and can be used in different ways. This kind of software does not have explicit pedagogical assumptions embedded, it has been conceived as **'teaching material/resources provider'**. Examples of software of this group are word processors, encyclopaedias, content-rich CD-ROMs, spreadsheets, e-mail software.

Understanding that this classification has exemptions and therefore that is not exhaustive, it will never the less serve for the purpose of analysis. These three groups reflect different tendencies of conceptualising the role of software at the design stage within the framework of the teaching/learning activity.

The analysis that follows will expand some ideas about each of the defined groups, addressing the mayor issues that are expressed in the literature.

2.2.2 Learning centred software

In this group the software design is viewed as an activity whose goal is to produce a tool that is expected to have an effect, or impact, at a cognitive level. In other words,

the design is grounded in some learning theory which gives the framework for the software design. These theories include behaviourism, constructivism, and others but all have a clear element of self-determined learning and therefore have common assumptions in the design. This group also includes software that incorporates some teaching strategies, but generally those are embedded in a learning framework which is explicit in the software. From the designer's perspective, the arena of learning is in the interaction of the student with the computer. Software of this group can be represented with the following figure:

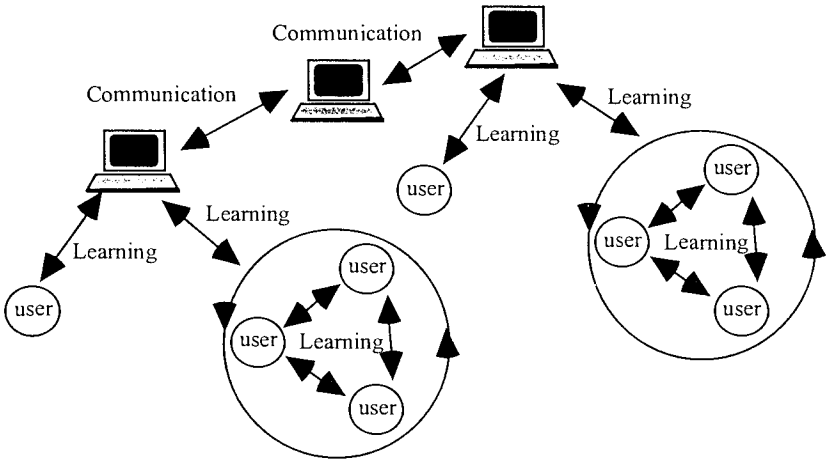

Figure 2.1. Model of learning centred software

In this scheme students interact with the software and while this interaction occurs they 'learn'. The locus of learning is in the interaction with and around the computer.

Reusser (1993) has software of this group in mind when he describes educational software as:

> Computer environments should be seen as mind-extending or catalysing tools for intelligent and volitional learners and virtually autonomous problem solvers. They should provide stimulating and facilitating structures in order to promote meaning construction activities, such as planning, representation and reflection.
>
> (Reusser 1993, p. 146)

Another example of this sort of design is given by Laurillard (1990), she describes two models for teaching and learning: the didactic model and communication model. In the former she speaks about a 'preceptual knowledge' that is transmitted by the teacher to the student. In the latter she describes knowledge as a 'negotiable commodity' between teacher and pupil.

Based on this latter model she specifies the following software requirements:

- the student should have direct access to the object domain
- the software should have operational knowledge of the domain
- the software should be able to give intrinsic feedback
- the software should make the goals of the exercise explicit.

As in the previous one, this definition excludes the teacher in the learning process, which is not the case in all the software included in this group. Other definitions correspond to the categories mentioned before and include the classification of the computer as tool, as tutee and by some of the educational paradigms (instructional, revelatory and conjectural).

From the perspective of the present research, the main problem of this kind of software is that the assumptions made during the design are well grounded in learning theories but it is assumed that the computer will be used in a specific way in order to produce the designed effect. This assumption demands of the teachers that they act in a particular way in order to create the situation in which the student can interact with the software in the manner intended by the designer so that learning can take place[2].

2.2.3 Teaching centred software

The design of software in this group has its origins in particular teaching methods, that is it has been conceived as an organisational aid (i.e. management) for the teacher in the classroom. The essential difference between this group of software and the pervious one, is that the software design has one (or several) explicit assumption(s) of how to use the computer in the classroom.

This alternative way of designing educational software integrates the computer into a certain teaching strategy, giving the teacher a special role in the activities. This role is made explicit in the design of the software. The locus of learning is in the classroom activity, not in the interaction with the computer, in fact, the software could be

[2] It is possible to have a second level of classification within this category. This level would be defined using the different theories of learning, instruction, and cognition. For example:

- Learning theories: Constructivism, Behaviourism
- Instructional theories: Bruner, Gagne, Derived from Vygotsky
- Cognitive theories: Derived from Vygotsky, Derived from Piaget
- Communication Models: Students control of learning

Wilson & Cole, (1991) present a somewhat similar classification of cognitive teaching models·

designed in such a way that the student does not need to have direct interaction with it. Software in this group can be represented with the following figure:

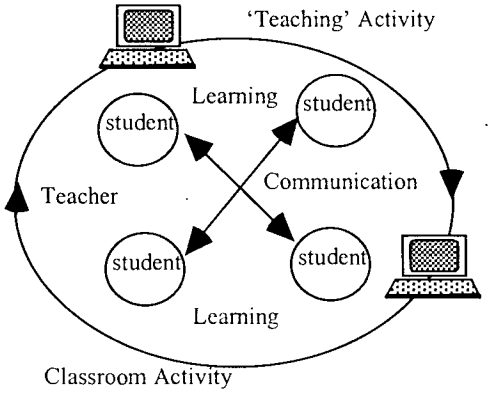

Figure 2.2. Model of teaching centred software.

In this scheme students are part of a learning activity and the computers take the role of supporting the activity. The locus of learning is in the classroom activity.

Fraser, Burkhardt, Coupland, Philips, Pimm, & Ridgway (1991) describe the different classroom roles that the teacher, pupil, or computer adopt when using software in a classroom situation. The different roles described are:

- Manager (tactical), corrector, marker, computer operator
- Task Setter, questioner, example setter, strategy setter
- Explainer, demonstrator, scene setter, image builder, focuser, imitator, rule giver, coach
- Counsellor, adviser, helper, devil's advocate, encourager, stimulator, listener/supporter, observer, receiver, diagnostician, problem solver.
- Fellow pupil, rule applier, hypothesizer, problem solver
- Resource, system to explore, giver of information.

(partial transcript of Fraser, et al. 1991, p. 212):

These roles reflect the behaviour of the teacher and/or students while using a piece of software in a classroom lesson, and they could form a useful starting point for thinking about the design of software in this group. An example of this kind of approach is reported by Dockterman, (1991) in a description of producing software to be used in a one-computer classroom situation.

Mercer, (1993), describing the implications of the context in learning, writes:

1 It implies that the process of learning about, or through, computers is not primarily to do with the relationship between a learner/user and the machine - the 'interface' - or even the software being used. It is instead very much to do with the contextual framework within which the learner/user is doing things with the computer.

2 It implies that what is learnt by particular children through the use of computers may only be understandable in terms of the history of the teaching-and-learning relationship in which that learning took place.
(Mercer 1993, pp. 31-32)

With this definition, Mercer (1993) changes the focus of the software design from the student-machine interaction to the context of use. But he still assumes a certain amount of interaction between the student and the computer, which is not required in this description of software in this group.

In a similar vein, Laurillard (1993), starting from a definition of the process of teaching at University level, examines how different media could support different stages of this process. Further on, she presents a classification of existing teaching media which is structured using her model of university teaching (the classification was presented in section 2.2.1). In doing so, she is advocating for conceptualising software as an aid for teaching, and in this sense, she is giving additional support to this category of software.

On the other hand, looking at her proposition in more detail, it could be argued that her definition of the university teaching model is based on the learning processes that should be happening during teacher-student interaction. If so, her classification of educational media would also be based on the learning processes that the software would support and therefore it would be useful for classifying learning centred software rather than teaching centred software. However, her argument is still consequent with the ideas that support this category.

Further support for the idea that much educational software has a role in the whole classroom situation rather than in the individual-computer interaction is to be seen in the work of Olson, (1988), who found two different ways in which teachers use the computer:

- Trojan horse: The computer is used as an aid to innovate in the teaching strategy.
- Expression tool: The computer is used as an instrument to express how they want to be seen as teachers.

These findings show two different roles of the computer as a teaching resource in the sense that it provides the teacher an aid to perform his(her) job. In this case the aid is not directly related to the specific teaching activity, but to the professional performance of the teacher.

2.2.4 Teaching material/resource provider

In this group of software those packages that could be seen as multipurpose software are included, which serve basically as a resource to carry on a specific task. The software here does not include an explicit learning or teaching strategy, but it helps to perform learning and/or teaching processes.

The focus of learning could be in the student-software interaction or in the activity organised by the teacher. The computer is conceptualised as a special tool for performing some activity or as a powerful book-like resource.

There are many experiences that report the use of 'traditional' software in schools, such as word processor, data bases, spreadsheets, encyclopedias, etc. This is software that, generally, has been designed to be used in other environments (industry, administration, library, home, etc.) and is being 'introduced' into classrooms settings and therefore is unlikely to emphasise cognitive and pedagogical aspects (Squires, 1996). Teachers who use this kind of software are often advocating 'vocational' arguments for its use, arguing that students need to be prepared to use this kind of software when they enter into the job market (Squires, 1996).

2.2.5 Discussion

Leaving aside the use of software as teaching material/resource provider it was argued that educational software design follows one of two tendencies:

- Designing software for learning: Here the authors are trying to build software that implements some learning, cognitive or instructional theory. In doing so they give the computer a high degree of responsibility for the learning outcomes.

- Designing software for teaching: Here the authors are trying to discover ways in which a computer could be used as part of the teaching process. They include a more systemic view of the process of teaching and learning, rather than simply a particular conception of learning with or around the computer.

As regards the first tendency, despite the designer's intentions, research shows that the software most frequently used in school is based on drill and practice activities (Cuban, 1997; Evans-Andris, 1995). In respect of the second tendency the key question remains: what is the role of the computer in teaching?

Despite high expectations about the use of computers in education, research has shown that in this field the role of the technology is controversial (Lowther & Sullivan, 1994) and its effects not conclusive (Johnson, et al., 1994). In trying to explain this situation there are two main arguments, the first is that the teacher should be more technology literate in order to master the technology, and the second refers to the lack of understanding of the software designers about the teaching/learning process.

Supporting the first argument, Handler (1993) and Winship (1989) complain about the lack of appropriate training and support for teachers that want to use this technology. Other problems with software reported by Winship (1989) that reflect a rather intermediate position are:

- Teachers find it very difficult to identify software that they believe will be useful in their own teaching.
- Much of the existing software is difficult to integrate into teaching because it is either too easy, too hard or it takes too long before useful results are produced.
- Often the teachers must put in a great deal of preparation time before the software can be used in the classroom.

There seems to be a dissociation between what is being offered today as good educational software design, what teachers really do with software in the schools and what teachers' expectations are of what could be done with it.

Supporting the second argument, one general critique of the design of educational software to date is that there is a lack of understanding of what is happening in the classroom and of the discourse of the teacher and his/her reality. The different arguments for this general claim can be summarised as follows:

- While studying what teachers do with computers, Mercer & Scrimshaw (1993) and Olson (1988) claim that too little is known about the activities in the classroom with the computers. They argue in favour of the observation and analysis of what teachers actually do with computers and, starting from these results, they propose to revise the design of existing software.

- While observing what pupils do with computers, Crook (1987; 1991) argues that technology has a potential for catalysing socially organised learning. He outlines the theoretical basis for developing computers in such a way. His focus is on the social interactions that occur around the computer, rather than with the computer.

- Winograd & Flores (1986) talk about the general issue of understanding the domain of action of the user. They analyse the structures of communication that industry workers use in their daily work. Based on these structures, they propose a set of guidelines for designing software. In the case of the present study, the users would be both, the teachers and the pupils. In a similar vein, Crook (1998) proposes three features of pupils' social interaction that should be considered while designing educational software for collaborative work. In doing this, he is integrating features of this particular social context into the software design. Following a similar tendency, Mantovani (1996) proposes a model for integrating the social context into the design of software. His model of the social context considers three levels: construction of context (social context), interpretation of situation (situations) and local interaction with environment (artefacts).

- Koedinger & Anderson (1993) argue that while designing educational software, it is important to understand the instructional context and, in particular, to understand the role of the teacher. In a similar vein, Reusser (1993) speaks about the pedagogical and didactic philosophy that a software design should incorporate and the importance of the learning and teaching activities that take place in the 'behavioural setting' of schooling. From a different perspective, Chen (1995) proposes a methodology for characterising computer-based learning environments that includes three dimensions of analysis: (i) the types of knowledge presented, (ii) the pedagogical strategies to communicate the knowledge and (iii) the form and functions of interaction elicited. In doing so, Chen is indicating the need for considering these dimensions for the design of educational software.

These arguments coincide in stressing the necessity of having a more situated view of the software design and they propose different focuses to implement such view. Some argue in favour of considering the actual experience of the teachers and the pupils; others highlight the importance of defining a model of the social context in which computers will be used; and lastly, some point out the importance of the pedagogical and didactic philosophy (or theory) that the software should incorporate. The main implications for this study are that the design of educational software needs to be examined and that, while doing so, all these arguments should be brought into consideration.

For the aims of the present study, it will be assumed that school software should be designed to be used in the school, for purposes and needs that are present in the school. The implication of this assumption is that to design a piece of school software, first it is necessary to know the needs of the school and from this starting point, to design a piece of software to help to satisfy these needs. In other words,

understanding the role that the software can play in this solution within this context. It implies not imposing preconceived designs of software to improve teachers professional activities, but designing, based upon their actual practices, pieces of software that enable them to do more effectively what they do.

2.3. EDUCATIONAL SOFTWARE DEVELOPMENT

In order to analyse the process of software development, this section starts with a brief look at the general issues of software engineering and the methods for software development proposed in this area. Then it presents some reported experiences about the software development process.

2.3.1 Software engineering

This section includes a short description of some of the main software development models proposed in the literature. There is still a strong debate about the utility of formal methods to develop software, it is not the intention here to recreate such debate. The idea is to present the main tendencies that are being used and outline the stages of some software development methods. The purpose of this section is to provide evidence from a different theoretical framework which will support the ideas about educational software design presented in the previous section.

The concept of the software 'life cycle', defined by Sommerville (1989), was first used to define a model of the software development process. Based on this model there have been several propositions, including the waterfall model and the revised waterfall model. Both of them are based on the following sequence of activities:

- Requirements analysis and definition
- Design
- Implementation and units tests
- Installation
- Evolution

The waterfall model conceives the software development process as a linear sequence of activities that are clearly marked and that can be controlled. The revised waterfall model has similar activities but incorporates feedback between the different activities, transforming the model from a pure sequential one to an iterative model as shown in figure 2.3. In each stage there are activities of validation and control incorporated that give the required feedback.

There are other models of software development proposed, such as the prototype evolution model and the spiral model of software development proposed by Boehm, (1988). These models propose a sequence of stages that allow the developers to reduce the risk involved in the development, by introducing reiterative tests of a prototype of the software which evolves within the project.

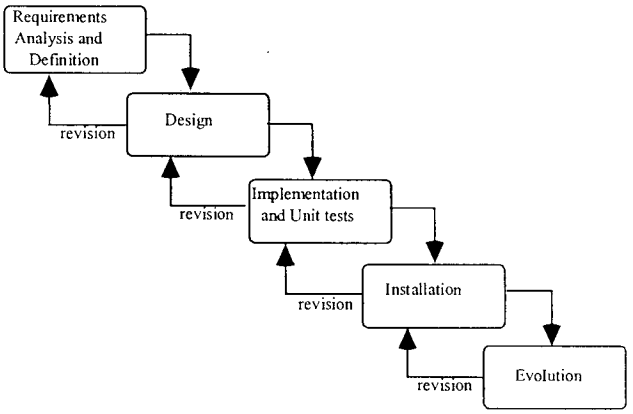

Figure 2.3. Revised waterfall model

The use of prototypes allows the software engineer to acquire information about the requirements, technical feasibility, and other risk elements, but it is not well integrated to the end product because of its partial and ill integrated step by step development. Boehm, (1988) proposed a new approach to this development method, which has been named the spiral model. It integrates the waterfall model with the evolution of prototypes.

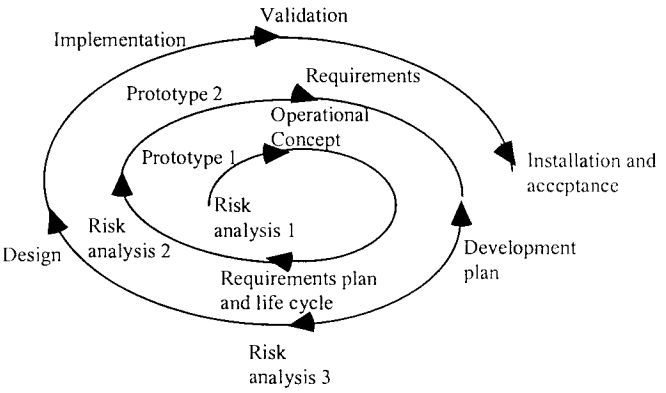

Figure 2.4. Spiral model of software development

Figure 2.4 shows an example of this model with three cycles. The accumulated cost of the project is represented by the radius of the spiral at each point.

Yet another model, which is similar to the prototype development model, is the Exploratory Programming, but it does not have a structured approach. Another model for software development can be found in Hinostroza, Hepp, & Straub, (1996), which is the result of adapting, modifying and expanding theoretical models to address real

needs (Potts, 1993). These needs arise from the interdisciplinary nature of the work, the incorporation of multimedia and the focus on education. This latter focus imposes additional requirements to be considered during the design stage, such as the need of having a more permanent effect on the user at a cognitive level (for example, the user should learn something).

This method consists of four principal activities, some of which are carried out in parallel. It is based on the incremental development of a prototype in which critical evaluation, modification and redefinition are carried out at all stages, seeking to culminate in a field-tested product. The four main activities are Project Definition, Design of the Application, Development of Prototypes and lastly, Product Manufacture.

In a similar vein Galvis, (1994) describes a method for educational software development that corresponds to an adaptation of the revised waterfall model (figure 2.3) for this particular type of software. His main contribution is the accuracy in the description of the activities to be carried out at each stage, particularly at specifying all the pedagogical and psychological considerations that should be taken into account in each step of the development process.

The use of fourth generation languages is also an alternative for software development, in which the problem to be solved is described using an abstract representation and then code is generated automatically. Generally these types of systems have been designed to develop information systems for commercial companies and therefore are not very suitable for developing educational software.

Apart from these models, there are other techniques that are used to produce software, such as the use of formal specifications that allows to specify software requirements using formal notations and the code results from the translation of the specifications into a programming language either automatically or manually. Among these, the most widely used are the Vienna Definition Method, Z Notation, Programming through transformations, and the Cleanroom Method (DeGrace & Hulet Stahl, 1990).

Despite of the different methods or techniques used for software development, it is still difficult to produce a piece of software. In 1987, Brooks published the paper 'No silver bullet: essence and accidents of software engineering', where he characterises the essential difficulties of software development. These are:

- Software products are very complex because they are built combining a large number of different entities (functions, subroutines, etc.) that need to work in a

co-ordinated and consistent way. This characteristic is rare in other comparable human constructions.

- Software products need to be adapted to the different contexts in which they will be used. There are no unifying principles in software since every organisation (industry) or human system is different and therefore has particular requirements.
- Software product are under a permanent pressure for change, they need to evolve into more powerful products and/or products that perform additional functions. Normally new versions of products replace the old ones, in the case of software, the new version is basically the same piece of software plus the additional features.
- Software products are invisible, they only exist in the domain of the ideas of the author and the user will never 'see' the product, (s)he will only use it and be aware of its 'effects'.

These characteristics are still part of the software being produced today and Winograd (1995) supports its 'essentiality', by saying:

> One of the key differences between software and most other kinds of artefacts that people design is the freedom of the designer to produce a world of objects, properties, and actions that exist entirely within the created domain.
>
> (Winograd 1995, p.70)

He adds that this special characteristic of software products make it very difficult to share the idea of the product to be developed, because the artefact belongs to the author and it is not possible to apprehend it in the computer, but simply to actualise it through every attempt to use it. These special conditions of software makes it very complex to produce. Such difficulty, added to the design problems mentioned in the previous section, constitutes a rather challenging scenario for educational software developers.

Nevertheless, the issue of software development methods will not be further pursued here, rather the main focus will be that of software design issues. From this perspective, the actual scenario can be summarised as:

> The design of a computer application, regardless of its specific details, is intertwined with the design of the organisational interactions that surround its use.
>
> (Winograd 1995, p. 73)

Given that educational software is supposed to be used in the classroom, the 'organisational interactions' that should be analysed in this case are the ones that happen between the pupils (or among them) and the teacher. This implies that the

users of the software are both, teachers and pupils, so far they are permanently interacting with each other in the classroom. Considering such perspective, it could be argued that in order to design a 'good' piece of software both type of users would need to specify their requirements to the software engineer and participate in the development process, as Squires (1996) describes that is the case of 'successful' production of educational software. The problem is that, although this simple idea could work, research in the Software Engineering area indicates that misunderstandings between software designers and software users are very common and that this is one of the main problems faced during the software development process (Gilb, 1988; Olsen, 1993).

Colin Potts in his article 'A Software Engineering Research Revisited' in 1993, suggests that an emerging trend exists today toward using industry as the laboratory for discovering new techniques of developing software and that this tendency emphasises the relevance of an empirical definition of the problems, the study of actual cases and its contextual aspects. Hughes, King, Rodden, & Andersen, (1995) in arguing for a role for ethnography in software development claim that it is vital for designers to understand the work setting as a socially organised setting prior to initiating the design stage. Similar arguments that there is a need to understand the context of use of the technology are given by Crook (1991), Koedinger & Anderson (1993), Mercer & Scrimshaw (1993), Olson (1988), Reusser (1993) Suchman (1987) and Winograd & Flores (1986).

2.3.2 Development of educational software

There are two main tendencies in which different software design teams have tried explicitly, or implicitly, to tackle the problem just mentioned. The first is transferring the software design responsibility to the teacher, and the second one is to incorporate teachers as part of the design team.

The first tendency means that a teacher, by him/her self, designs a piece of software that (s)he finds to be useful for his(her) activities (Fitzgerald, Bauder, & Werner, 1992). This has been tried in different empirical situations but the essential problem here is that the highly specialised technical knowledge required (software engineering, programming methods and techniques, human computer interface design, etc.) to produce a professional piece of software is far beyond the normal training of teachers (and other professionals as well). In this sense, it is easy to understand that resulting software can be very useful in bounded situations, but certainly not satisfactory in general, neither for external evaluators (for example, the pupils) nor for teacher themselves. In fact, Hoyles, Noss, & Sutherland (1991), in

their final report of the Microworlds project, found that the process of creation of microworlds was of much more value than the product itself, they say:

> We now see the creation of a satisfactory *product* (i.e. a microworld for general use in the classroom) to be both more complicated and of lesser importance. ...
> Achieving a satisfactory product requires a vastly different range of skills of production and sequencing than is normally required for teachers to carry out their work in the classroom.
>
> (Hoyles, Noss, & Sutherland 1991, Vol. I, p. 29)

Despite the fact that this experience was focused in other issues (i.e. "to develop, implement and evaluate a programme of in service teacher education concerned with the use of generic computer applications within the secondary school mathematics curriculum" Vol. 1, p.1), this study highlights the difficulties of transforming teachers into software developers.

The second tendency implies the involvement of one or more teachers in the software development process with defined roles and activities. There are some experiences reported in the literature about the incorporation of teachers in development of software (for example: Char & Hawkins, 1986; Hawkins & Kurland, 1986; Watson, 1987; Watson, 1993)

In Char and Hawkins (1986)'s report, teachers were part of the design process, advisors in curriculum issues and evaluators in different stages of the software development process. Hawkins & Kurland (1986) describe an experience of requirement specification for the design of information-managing tools in the schools. Both projects started from a prior conception of the piece of software to be developed but they didn't have a stage of 'requirements specification' in order to analyse what was really needed in the school(s).

Watson (1987) describes a similar project that included teachers, curriculum developers, programmers and system analysts (and as the result of this interaction there emerged what she defines as a "CAL developer" p. 44) in the development group. In this case requirements were defined by the curriculum (as the name of the project suggests 'Computers in the Curriculum'), so they implemented several units of software based on units of the curriculum. This emphasis on curriculum gave them the orientation and framework for the software design, not the school itself.

Watson (1993) reports about a study based on the analysis of what is required in the school from a piece of software. The analysis is based on comparing the general practices to be found in a school with existing software tools (educational activities

v/s the contribution of IT). Based on this argument, they left the teachers to build projects where the students could program pieces of software.

One study which incorporates a requirements specification is a study by Self (1985), where he describes the process of software design and includes the requirement specification stage. But when he explains it, he mentions aspects such as who will use the software, accommodation of the teacher, etc. He does not include a diagnosis of the school's problems and a justification of the piece of software.

There are a few good examples of approaches to the use of technology in education based on requirements analysis, one in particular is McConnel (1994)'s course design, which starts with an analysis of co-operation and learning, and from this designs strategies for using the technology to support the different ways of implementing computer supported collaborative learning.

However, there seems to be little (even if some) experience about development of software based on an analysis of the needs and activities found in the schools, almost all the software development projects described above were based in a preconceived idea of what kind (or specific piece) of software should be developed and teachers and students were incorporated in the process after this initial definition or conception.

2.4. SOFTWARE EVALUATION

There are many views of software evaluation, so the first issue is to define what is to be understood as evaluation. In the literature that defines meanings of evaluation there are a number of conceptions that range from conceiving evaluation as a pure quantitative procedure that will state the position of something in a predetermined scale, to conceiving evaluation as a qualitative analysis that will help to understand something. In this sense, evaluation is as complex as research and some authors claim that, in fact, evaluation is research. Accepting this last position it is possible to envision the type and amount of controversy simply by transferring all the discussion about research methods and techniques to evaluation. This review tries to keep apart from these discussions and focus on software evaluation only, hoping to be able to gain clarity in some points.

2.4.1 What is software evaluation?

Squires & McDougall (1994) make the distinction between evaluation, selection and review of software. These concepts are also found in Winship (1989)'s book, where

he gives an initial framework to focus the discussion. The different activities defined by Squires & McDougall (1994) are:

- **Selection**: we mean the assessment of software by teachers in anticipation of its use with groups of students in classrooms or with individual students.
- **Review** of educational software is the process of assessing it to write a summary of its features and characteristics for the information of others who are involved in software selection.
- **Evaluation** of software can take place during either the development of the software or the use of a complete package.
  - Evaluation during the development (formative) focuses on possible modifications to the software.
  - Summative evaluation after publication, is concerned with the quality and variety of experiences that the software can support."

(Squires & McDougall 1994, pp. 3-4)

Selection of software could also involve some kind of evaluation. When the teacher selects one piece of software among other pieces, it could be interpreted as the consequence of an evaluation, but the sense given here is that in the process of choosing there is not, necessarily, a formal process that ends in a judgement. It could be a matter of taste rather than a conclusion of a formal procedure.

The software review process where the evaluator is asked to review the software holistically (Reiser & Kegelman, 1996) is seen here as one technique that could lead to an evaluation, but it is still a process of observation which should have no judgement in it.

For the purposes of this research software evaluation is understood as a formal procedure that helps someone else to build up a judgement about the software, in the sense of its effectiveness in the areas or activities for which it has been designed or is been used. In this review, formative evaluation was excluded because it is part of software development methods, that is, software engineering which is not the focus here.

2.4.2 Types of software evaluation

The different evaluations techniques are classified here in three groups:

Experimental methods

In this approach the experimental method is used in order to assess the effectiveness of a piece of software. Examples of these methods are found in reports by Reiser & Dick, (1990) and Zahner, Reiser, Dick, & Gill, (1992), in which they propose a

specific method of carrying on an experiment to evaluate software. In general terms, they follow the model of pre-post tests using experimental and control groups.

Check-list approach

This groups of methods are based on applying a set of predetermined criteria to a piece of software and the result will enable the evaluator to find out if the software is suitable or not. Examples of this methods are found in Tolhurst (1992)'s study and Squires and McDougall (1994) does an extensive review of the different options.

Qualitative evaluation

Some evaluators argue for the evaluation of the software in a situated context and propose the application of qualitative methods to evaluate software. Examples of this are given by Crook (1991) where he changes the focus of analysis of software from the interaction with the computer to the interaction around the computer, and Squires and McDougall (1994) propose a method of analysis of software based on a 'perspectives interactions paradigm' that contribute to evaluate software, the perspectives are: teacher- student, teacher-designer and student designer (the designer is meant to be the software designer or programmer). Each perspective is presented with several different aspects to be analysed, for example:

- Teacher-Student perspective: analysis of the kind of classroom interactions and activities that might be sponsored by the package.
- Student-Designer: analysis of the learning process and assumptions which are designed into the package.
- Designer-Teacher: analysis of the curriculum issues, notably concerns with the curriculum process and content designed into the package.

These evaluation methods are still being examined (Baumgartner & Payr, 1997; Squires & Preece, 1996) and there is still no clear conclusion about their effectiveness (Reiser & Kegelman, 1996).

2.4.3 The issue of software evaluation

One of the main problems of software evaluation seems to be the nature of software itself, in that it is not readily accessible, unlike a book or other teaching material, it is not accessible to inspection, by skimming, scanning, browsing, etc. It must be run and explored in the computer and the teacher will need a certain expertise to be able to do it (Squires & McDougall, 1994). Winograd (1995) expressed a similar view about the nature of software, describing the particularity of software design as the creation of an independent artefact that exists and has sense in a defined domain. In a similar vein

Olson (1988) defines software as 'ideaware'. Another line of argument concerning the difficulty in using evaluation methods is the lack of time for the teachers to evaluate the software. There is evidence that the teachers do not spend much time preparing computer based lessons, they prefer to employ a piece of software that they can use without even knowing it (Evans-Andris, 1995).

These ideas lead to an understanding of software as an artefact that needs to be used and explored in order to make sense of it and that in this interaction the user will build up a personal mental representation of the software. Within this perspective, software evaluation is transferred to the domain of evaluating situations that are created within the interaction between the user(s) and the artefact.

Winograd and Flores (1986) base their analysis of software in a philosophical position derived from Heidegger who argues that the separation of subject and object denies the more fundamental unity of being-in-the-world (dasein) because:

- Our implicit beliefs and assumptions cannot all be made explicit.
- Practical understanding is more fundamental than detached theoretical understanding.
- We do not relate to things primarily through having representations of them.
- Meaning is fundamentally social and cannot be reduced to the meaning-giving activity of individual subjects.

(Winograd and Flores 1986, pp. 32-33)

This idea is also used by Suchman (1987) in developing the concept of 'situated actions' as a term that underscores the view that every course of action depends in essential ways upon its material and social circumstance. She describes the properties of the ethnomethodological view of purposeful action and shared understanding as:

1 plans are representations of situated actions;
2 in the course of situated action, representation occurs when otherwise transparent activity becomes in some way problematic;
3 the objectivity of the situations of our action is achieved rather than given;
4 a central resource for achieving the objectivity of situations is language, which stands in a generally indexical relationship to the circumstances that it presupposes, produces, and describes;
5 as a consequence of the indexicality[3] of language, mutual intelligibility is achieved on each occasion of interaction with reference to situation particulars, rather than being discharged once and for all by a stable body of shared meanings.

(Suchman 1987, pp. 50-51)

[3] The word indexicality is used in the technical sociological sense here.

In this definition, the second property refers to the concept of breakdown mentioned by Winograd & Flores (1986) and the third one refers to construction of common-sense from the actions. The fourth and fifth refers to the communication process using the language and specifies its context dependency and particular meaning.

Starting from such a similar theoretical framework these authors (Suchman, Winograd and Flores) use it to develop different conclusions. On the one hand, Winograd and Flores (1986) describe the different types of interactions among people that take place in a particular context (they develop the case of the communication for management in the context of an office). Their description uses linguistic categories (speech acts) to characterise each type of dialogue that occurs (or should occur) in this context for such purpose. Based on this example, they propose the design of what they call 'tools for conversation', which are pieces of software that implement the structures of the dialogues described and thereby provide a framework for interaction which helps to change the interactions themselves.

In this proposal, they are using the technology as a means to change the structure of the communication among people and thereby to change what is happening in a situation. Their argument is that "language does not describe a pre-existing world, but creates the world about which it speaks" (p. 174), and if they change the communication, they will be able to change the world in which such communication happened.

On the other hand, Suchman (1987) describes her argument as:

> My argument has been that as long as machine actions are determined by stipulated conditions, machine interaction with the world, and with people in particular, will be limited to the intentions of designers and their ability to anticipate and constrain the user's actions.
>
> (Suchman 1987, p. 189)

Considering that, in a way, a piece of software is a particular implementation of a 'plan' (set of actions to be implemented by a user), it seems reasonable to expand Suchman's argument as to include not only machines but also other artefacts (such as software). Then, it would be possible to say that software actions could also be limited to the designers' intentions and their ability to anticipate and constrain the user's actions. In this sense, it is possible to realise that Winograd and Flores (1986) use Suchman's argument for their convenience, this is, they want to constrain the user's actions in order to make their communication in the office more effective.

Whether this can be considered right or wrong, will not be discussed here in so far it is not the focus of this research, the important issue for this study is that these authors coincide in the importance of the context in which such artefacts are used and that they (the artefacts) embed the power of changing a given reality (accordingly to Winograd and Flores, 1986) and that a given reality changes the artefact in use (accordingly to Suchman, 1987).

Looking at a similar issue but from a different area (i.e. education), Crook (1987; 1991; 1994) adds another element to this discussion, he concentrates on the concept of interaction between students and teacher while they are using the computer. In doing so, he defines a social context in which this interaction occurs. His arguments are based on ideas proposed by Vygotsky and refer to the Zone of Proximal Development. As he defines it:

> The novice in the cognitive system, the pupil let us say, appropriates the goals and strategies that are manifest in the overt, jointly organised problem solving behaviour. Much of this will be carried by the language that is exchanged within the interaction. It is in this way that the learner is said to make internal ('intramental' in Vygotsky's terms) the events in which they have participated on an 'inter-mental' level.
>
> (Crook 1991, p. 83)

Considering the arguments presented until now, it is possible to say that a piece of software can be designed as to change the structures of conversation; that a user could change the plan embedded in the software and thereby change its initial design and finally, that the interactions in the social context in which it is being used could change what happens with the software.

In such framework, and in order to be able to evaluate a piece of software, it is possible to argue that it has no sense to evaluate the software as an isolated element, it needs to be evaluated in a specific social context and circumstance. The software as it is defined here needs to be internalised[4] in order to be understood and thereby evaluated.

In this sense, the proposition made by Squires and McDougall (1994) could be seen as the most suitable or appropriate, because it allows to structure the analysis focusing in different dimensions of use or interaction (teacher-student, student-designer, and designer-teacher). This way of structuring the analysis could lead to a meaningful evaluation. This proposition can be complemented with the perspective that the

[4] Understanding internalisation as a process wherein an internal plane of consciousness is formed and the external reality at issue is a social interactional one (Wertsch, 1985)

evaluation should take place in a real situation in order to be comprehensive, in this case an analysis of each interaction in a given scenario would be needed.

In doing so, the evaluation itself loses its predictive value because the software is already being used. Evaluation then could only explain a certain outcome or process that is happening or has happened.

2.5. EDUCATIONAL INNOVATION

The introduction and use of information technology in education is commonly associated to a process of 'educational innovation' (for example: McDonald & Ingvarson, 1997; Olson, 1988; Wright, 1987). This perspective provides a different framework for the analysis of educational software, in so far its adoption and use are considered to be part of a process in which the teacher and the school is engaged. From this perspective, this section presents an overview of the literature reporting such processes.

Hurst (1983) raised the question of whether educational change is different from change in other institutions. The arguments that support the difference are:

- Teachers spend less time with colleagues and more with students
- Teachers are not usually subject to detailed supervision and assessment of performance.
- In developing countries, at least, teachers are often inadequately trained, underpaid, and enjoy relatively low status.
- Educational systems are often highly centralised

But, he argues that it is a mistake to think that the organisational context determines the behaviour of its members, and concludes that teachers do not respond to new ideas in a way that is fundamentally different from anyone else, and just as with other people the results of an innovation process directed at teachers will differ depending on the institution, group or individual.

Fullan & Stiegelbauer (1991) give some conceptions about schools in terms of goals, power, decision making, external environment and teaching process that could help to analyse these organisations and their innovation possibilities. Fullan (1992; 1996) distinguishes between the adoption of an innovation and the implementation of the innovation, and list the themes and factors that influence these processes.

Some conclusions drawn by the authors that have been considered about innovation in education are that schools, like any other institution, are different from each other in particular, but share some common characteristics. And, the rejection, adoption or modification of an innovation by an institution is uncertain, but there are several conditions that predispose to success, but do not assure it. They mention two points that are important for success:

i) Mastering the innovation in the classroom and sharing it with colleagues (Fullan, 1992; Huberman, 1992). Huberman (1992) describes innovation it as a process of grafting the new on the old, and he comments that every 'old' is a distinctive, local context with its own history and configuration. Olson (1988) also defines the process of change as not one of substituting one practice for another, but one of subjecting existing practices to challenge posed by another well conceived practice. Both definitions have an evolutionary approach to change, rather than a revolutionary one. They suppose that 'the new' should be somehow grounded in 'the old' (the revolutionary approaches to change are not addressed here).
ii) The importance of understanding the teachers and teachers' self-understanding. Fullan and Stiegelbauer (1991) argue that the subjective world - the phenomenology - of the role incumbents needs to be understood as a necessary precondition for engaging in any change effort with them.

It appears to be clear that although there are institutional conditions that have to be fulfilled for change to take place, the one who implements the change is the individual and the focus of change is his(her) practice. Arguments supporting this assumption are given by Fullan (1993), he says that "systems do not change themselves, people change them", and Fullan and Stiegelbauer (1991) say that "educational change depends on what teachers do and think - its as simple and as complex as that" (p. 117).

2.5.1 Innovation and computers

In the new 'Information Age' computers and telecommunication are key tools that permit (and eventually produce) the change from the traditional bureaucratic culture of organisations to a new professional culture. Schools are part of this new scenario and they are required to adopt this new paradigm, that means to innovate and change. The reasons found in the literature are related to the need of society to have citizens who can live and work productively in increasingly dynamic, complex world (Fullan, 1993).

In many cases the justification for investing in information technology is based on the need to innovate in some or several areas or dimensions, rather than in the need for the technology itself. In fact, speaking about the rationale for innovation in schools, Grunberg & Summers (1992) rephrasing Fullan (1982) say: "schools tend voluntarily to adopt innovations which promote their image as up-to-date and efficient" (p. 259). Therefore, besides efficiency, efficacy, effectiveness, and other justifications for their use, computers are often seen as innovation 'symbols' or 'signs' and once introduced, act as catalysts in the process of change (Hawkridge, et al., 1990).

When computers are introduced into the schools there is an expectation of innovation, of change. Also, when new software is introduced into a classroom there will be a change, students and/or the teacher will be confronted by something new. The introduction of the computers themselves, and then the introduction of new software can be seen as two different elements triggering innovation in the classroom. This section adds a new perspective of analysis, presenting the software and computers as part of a larger process of innovation and, further, playing a special role in such a process.

Computers produce a wide range of effects in an organisation and much has been written about the different levels in which they impact (for example Zuboff, 1988, makes the distinction between automating and informating effects). One interesting specific effect of computers in schools is reported by Olson (1988), where he was trying to understand the way teachers use the computer. As mentioned in section 2.2.3, he found two different ways in which teachers use the computer: as a Trojan horse (the computer is used as an aid to innovate in the teaching strategy) and as an expression tool (the computer is used as an instrument to express how they want to be seen as teachers). These ways of using the computer were not perceived by the teachers itself, but were revealed by the process of analysis and interpretation that the researcher applied to his observations of their working with computers. In both cases it is possible to ask whether this is a computer driven or a computer supported process. In the former, technology plays the role of catalyst and in the latter it is a support for the ongoing process of change.

This role of technology is coherent with that of Winograd and Flores (1986), who argue that when a change is made, the most significant innovation is the modification of the conversation structure, not the mechanical means by which the conversation is carried out. This focuses the ideas of change not in the technology (computers in this case), but in the relation or activities that are carried out through (with, by, using) computers, in this case, teaching and learning.

Placing technology in a supportive role implies first, understanding the on-going process of change and second, choosing an appropriate technology. This view is coherent with the reflexive concept of change (Olson, 1988). Olson (1988) defines teacher behaviour as 'reflect on action', research activity as 'understand teacher intentions' and innovation activity as 'engage in critical analysis of practice'.

This issue was also addressed by Grunberg & Summers (1992), who in a review of the literature on innovation and computers in education conclude that: "the previous emphasis on the technical characteristics of the proposed innovation has evolved into a more context-sensitive approach focusing on how the proposed innovation fits the teachers' working conditions and value systems" (Grunberg & Summers 1992, p. 272).

In reference to software, Olson (1988) defines it as:

> Software is at heart, 'ideaware,' and more the 'idea' of the software is transparent to the teacher, the more likely the challenge to the 'ideas' in every day practice can be discerned by the teacher, and the process of reflective conversation begin.
>
> (Olson 1988, pp. 55-56).

The transparency of an idea for teachers will depend on their conceptions, beliefs and practices, this means that to design such software we should know and understand what teachers think about software and how they think it could be used in the classroom. The software should be useful for them, and should fit into their strategies, enabling them to change from present actual practices to new ones.

In this vein, Sandholtz, et al. (1997) present an interesting study that reports a process of instructional evolution in technology rich classrooms. The study was part of the "Apple Classroom of Tomorrow" (ACOT), a research-and-development collaboration between public schools, universities, research agencies and Apple Computer. The project started in 1985 and was set out to investigate how routine use of technology by teachers and students would affect teaching and learning. Based on their observations, they defined five stages of instructional evolution:

- **Entry**: Is defined as the first contact of the teachers with computers. At this stage "teachers found themselves facing problems typical of first-year teachers: discipline, resource management , and the personal frustration that comes from making time-consuming mistakes" (Sandholtz, et al. 1997, p 37).
- **Adoption**: At this stage "teachers showed more concern about how technology could be integrated into daily instructional plans. Interspersed among traditional

whole-group lectures, recitations, and seat work, teachers incorporated computer based activities aimed primarily at teaching children how to use technology" (Sandholtz, et al. 1997, p. 38).

- **Adaptation**: "In this phase, the new technology became thoroughly integrated into traditional classroom practice. Lecture, recitation, and seat work remained the dominant from of students tasks, but students used word processors, databases, some graphic programs, and may computer-assisted-instructional (CAI) packages" (Sandholtz, et al. 1997, p.40).
- **Appropriation**: This stage is defined by them as "less a phase in instructional evolution and more a milestone. It is less by change in classroom practice and more by change of personal attitude toward technology. It comes with teachers' personal mastery of the technologies they are attempting to employ in their classes" (Sandholtz, et al. 1997, p. 42).
- **Invention**: "In the invention stage, teachers experimented with new instructional patterns and ways of relating to students and to other teachers" ... "Interdisciplinary project-based instruction, team teaching, and individually paced instruction became common" (Sandholtz, et al. 1997, p. 44).

From the perspective of the present study, this report is useful, in so far it describes what teachers do in the classroom in the different stages of instructional evolution (or innovation process). And it also describes the role of the technology in each of these stages. One of the interesting aspects of this study is that in their descriptions they analyse both the hardware and the software that was used during the different stages. In fact, they characterise some of these stages by presenting the type of software that teachers or students use (spreadsheets, word processors, etc.). In this sense, they provide interesting evidence not only about the changes in teacher's roles or perceptions of the technology, but also about their evolution as users of different pieces of software.

On the other hand, due to the design of their study, it is possible to argue that because these teachers were part of an experiment, they were continuously under 'experimental conditions' which, from the researcher's point of view, undermines the possibilities of generalising the results, and also due to the involvement of a private company in providing the hardware for the schools, its possibilities of replication are very low[5]. Nevertheless, the experiences described constitute an interesting reference for comparison.

[5] Since the research was sponsored by ACOT, another issue that could be considered is related to the 'power' position of the researchers, in so far as the teachers being researched could have been under 'pressure' to show good results. This tension will not be discussed here because there is not enough information available to make such claim.

As regards the description of what teachers do with technology, there are coincidences with other reports about the introduction of computers into schools. For example, in the report of a longitudinal study discussed by Cox (1997), she compares the actions of two secondary teachers (experienced versus inexperienced in using computers) during a lesson using software. She reports that the experienced teacher was able to take a facilitating role with the pupils, and was able to promote individualised learning and innovative uses of software. The inexperienced teacher on the other hand, restricted herself entirely to technical advice on using the software. Using Sandholtz, et al., (1997) classification, it would be possible to say that the experienced teacher was at an 'Invention' stage while the other was at an 'Adoption' stage of instructional evolution.

In a similar vein, in the report "Computers in Schools: A qualitative study of Chile and Costa Rica" by Potashnik, Rawlings, Means, Alvarez, Roman, Dobles, et al. (1998) they describe the role of the teachers in the observed 'technology rich' classrooms as being at an 'invention' stage, they say:

> The researchers found trends toward less explicit direction on the part of teachers, greater student initiative, and more collaboration among students.
>
> (Potashnik, et al. 1998, p.19)

But in the same report, it is possible to find some descriptions about classroom activities that would place these teachers at an 'adaptation' stage of instructional evolution, where lecture, recitation, and seat work remained the dominant form of student tasks. In a similar vein, in Olson (1988)'s report, it is possible to see that teachers' attitudes towards technology and actions in the classroom do not correspond to a single stage of instructional evolution such as the ones reported by Sandholtz, et al., (1997). That is to say, some teachers experimented with new instructional patterns while they kept the role of assisting the pupils in using the software.

The intention here is to show that it is quite difficult to characterise a general pattern of instructional evolution, in so far as what teachers do in the classroom depends on several factors, like: the school's conditions (the principal, the community, the teachers and the particular training they received), national policies (ongoing reforms processes, type of administration, etc.) and a relevant factor to be considered in this study is the software available in the school. As Fullan (1996) argues, "educational change is a dynamic process involving interacting variables over time" (p. 274) and computers add a large number of new variables to this process, in so far as these machines serve different purposes depending on the software used.

2.5.2 Implications for this study

From this section three implications can be drawn:

- Computers are highly correlated with innovation processes, generally acting as catalysts. On the other hand, current views about innovation, tend to emphasise the importance of relating need to decisions about innovations or change directions (Fullan, 1996). Therefore, computers should act as supports for ongoing changes prompted by the needs of the educational system itself.
- In order to use computers as supports for innovation it is necessary to understand the context in which they will be introduced and the role that this technology will play in this context.
- Expanding the previous implication, it is possible to say that in order to use software as support for an innovation process it is necessary to understand the dynamics and roles that it could play in a classroom situation and specially the teachers' view of these roles. In other words, if teachers will use the software, it should be clear which need this software will address. This is coherent with some of the critical factors that have been found to relate to successful implementation of change described by Fullan (1996) (i.e. need and clarity).

These implications are coherent with the overall picture presented in previous sections, they shift the focus of the process of innovation away from the technology and closer to the context and actions, that is a situated perspective of change.

It must be reminded that these implications refer to the process of innovation, not to the innovation itself. Whether the particular innovations are in the right path or not, is not the focus of this study, and therefore alternatives for innovations were not discussed here.

2.6 PEDAGOGY

In the previous sections of this chapter it has been argued that in the area of this study (Information Technology in Education) there is a lack of understanding of the teachers actions and roles in the classroom and that there is a need for knowing teachers' actual beliefs about computers and software in order to contribute to the understanding of the role of the software in education from a situated perspective.

One possible starting point to form this understanding is in previous studies reporting on the nature of the professional knowledge that teachers share and what teachers do

in the classroom. With this objective in mind and in order to form a basis for later in this thesis examining the pedagogy of information technology, this section briefly reviews the general literature on pedagogy. Particularly, it will look for answers to the following questions:

- **Teachers' Knowledge and Expertise**: What do teachers know that could be incorporated into the design of educational software?
- **Teachers' Roles and Actions in the Classroom**. What do teachers do in the classroom that could be incorporated into the design of educational software?

Nevertheless, it is not the intention here to present a complete survey on this area, as the main focus of this study is related to information technology in education.

2.6.1 Teachers' knowledge and expertise

The body of knowledge that teachers share can be analysed from two general perspectives: First, looking at studies about professional knowledge and expertise in general, and second to look for studies about the teaching profession in particular.

Within the first perspective, a complete account of the different theories developed to characterise professional knowledge and competence is found in Eraut (1994), who presents a review of the following theories:

- Dreyfus' model of skill acquisition (Dreyfus & Dreyfus, 1986) which is based on the development of "knowing how" rather than "knowing what". In this model, the pathway to competence is characterised mainly by the ability to recognise features of practical situations and to discriminate between them, to carry out routine procedure under pressure and to plan ahead.

- Theories of clinical decision making, he describes the more quantitative views about the process of medical diagnoses (rational approaches, templates approaches, Frame System theory, etc.) and the more qualitative approaches that try to explain decision making in a situated context and individual dependant.

- Hammond's Cognitive Continuum theory (Hammond & al., 1980), that defines analytic and intuitive thinking as poles of a continuum, arguing that most thinking is neither purely intuitive nor purely analytical.

- Shön's theories about reflection-on-action (Schön, 1983), where he, while criticising the technical rationally models for describing professional expertise,

> proposes to search for "an epistemology of practice implicit in the artistic, intuitive process which some practitioners do bring to situations of uncertainty, instability, uniqueness and value conflict" (Schön, 1983, p. 49)

These theories for characterising professional knowledge and competence in general and in particular, teachers' expertise, could be used as a general framework for this study. But it does not seem that these perspectives can in fact help us to see the key issues that could enable to define the characteristics of a piece of educational software that could serve as a tool for teaching.

About the second perspective, that is, characterisations of the teaching profession, Grossman (1995) describes it as covering six domains, these are:

- Knowledge of content
- Knowledge of learners
- Knowledge of general pedagogy
- Knowledge of curriculum
- Knowledge of context
- Knowledge of self

All these types of knowledge could be understood as isolate 'pieces of information' that teachers should acquire during their training period or early teaching practice. But what is interesting for this study is the knowledge constructed through the combination of these dimensions, that is, their professional knowledge or expertise.

In this latter view, while characterising the teacher's expertise, Berliner (1995) describes five levels of expertise that were defined by Dreyfus & Dreyfus, (1986):

1) Novice
2) Advanced beginner
3 Competent
4) Proficient
5) Expert

Each level is characterised based on the way teachers plan their teaching, their behaviour in the classroom and their actions while facing critical situations. In general terms the way to progress through these levels of expertise can be summarised as improving the management of the classroom and developing strategies to plan the lessons. As mentioned by Eraut (1994), they do not describe the improvement of the 'didactic strategies' (the way teachers explain the contents and try get to an

understanding of the pupils). Then he describes eight different propositions about teacher expertise, again without reference to didactic principles that expert teachers would apply.

This short review of the area of 'teachers knowledge or expertise' shows that, although there is a significant amount of research in the area', the key issues for this study are not presented in such a way as to have ready applicability in this particular study.

Another source of information that could help to answer the questions mentioned before, is the study by Alexanderson (1994) about the way in which primary school teachers reflect on their everyday practice. In his study, Alexanderson found that teacher consciousness is focused on three different areas: (i) to the activity itself (65% of all the quotations), (ii) to aims of a general character (22% of all the quotations), and (iii) to a specific content (13% of all the quotations). In relation to the first area – the activity itself-, he found that they reflected on:

- How pupils are developed socially
- How a deep communication and retention is growing
- How the pupils are being noticed
- How to teach pupils to listen
- How systematic teaching leads to activity
- How structured and balanced teaching is performed
- How I think and how the pupils think

In these reflections, "methods of teaching were mentioned as a way to establish contact in terms of a good relationship with the pupils. One could trace the teachers' intention as being the capture of the attention of the pupils" (Alexanderson 1994, p. 144).

In relation to the second area -aims of general character-, Alexanderson (1994) reports that the reflections were about:

- Aims of the present conversation
- Aims of the open attitude in the teaching
- Aims for the teacher's active discipline
- Aims for catching the pupil's attention

In relation to the third area -specific content-, he reports that teachers' consciousness was also directed towards the aim "that the pupils learn a specific content". He describes that:

> The teacher's comments concerned mainly the reflection activity of the individual pupil and his or her ability to verbalise the reflection or to describe the reflection in a concrete area. Consciousness was then directed towards the way in which the pupil expressed his or her action in relation to a fixed content.
>
> (Alexanderson 1994, pp. 145-146)

This is, the teacher was focused on a particular learning episode rather than on a general learning aim. In general terms, what Alexanderson (1994) indicates is that highly skilled teachers are not necessarily driven by the aim that their pupils should develop certain specific understanding, knowledge or skill; the teachers were in general not directed towards some specific content of the pupils learning. He concludes that teachers were not oriented towards specific learning aims and that they rarely focused on means-ends relations.

In the same vein, accordingly to Marton (1994), "the general picture that investigation yields is that learning goals in terms of the pupil's mastering of some specific content did not appear as a mayor driving force of the teachers' acts" (Marton 1994, p. 35).

These findings could help to answer the initial question, in so far they show that teachers are not necessarily aware of the strategies and tactics that they use while teaching. Further, some research about teachers' practice, indicates that both planning and classroom interaction are responsive, compositional and situated (i.e. contextualised) (Yinger & Hendriks-Lee, 1995).

In this framework it could be argued that the actual research to date has been successful in proposing how to characterise teachers knowledge, and proposing models that could be used to define such knowledge (for example: Eraut 1994's proposition). Also, that there is research proposing different levels of expertise that a teacher could acquire and some research showing that teachers are not aware of the pedagogic techniques that they use while teaching (Marton, 1994).

As a contribution to this, Elbaz (1990) argues that such knowledge does exist, and that in order to be elicited, research should provide a space for teacher's voice, so that 'ordinary' teachers can tell stories about their 'ordinary' praxis of 'ordinary' teachers. She argues for such a method of eliciting teacher's experiences and further on, she argues that 'story' is "sometimes implicit in teachers' knowledge, that teachers'

knowledge in its own terms is ordered by and as 'story' and so can best be understood in this way" (Elbaz 1990, p 33).

This last claim could be used as an interesting framework to support the methodology of this study, in so far it would indicate that in order to elicit teachers knowledge they should engage in a process were they could 'tell a story' about their teaching and particularly in this present case a story about the use of educational software in their teaching.

2.6.2 Teacher actions and roles in the classroom

Teachers actions and roles have been studied and explanations offered from different perspectives, Olson (1992) describes three models that have been used to describe what teachers do in the classroom (definitions found in pages 2 to 13):

- **Systems Model**: this is based on the idea that inputs (decisions) and outputs (what happens in the classroom) can be tightly coupled and that people will act accordingly to system plans and, if they do not, the nature of the system will be adjusted to improve the link between inputs and outputs.
- **Ecological Model**: this is based on the knowledge of the complex social/technical situation of the teacher's practice.
- **Cognitive Model:** the theory behind this model is that teachers consciously follow rules for processing the information taken from the teaching environment. This information is processed according to steps determined by rules. This process goes on at the same time that action goes on - thought and action are linked.

In his book, Olson (1992) analyses the reasons why these models fail in describing teachers' actions in the classroom. As a response, he develops the conception of 'folkways of teaching' which is based on classroom routines that he illustrates using examples of teachers using computers, he argues that such routines express what teachers know how to do and why they do it. He does not expand on the description of these routines, but turns to discuss the relevance of understanding the tacit dimension of practice and the process of change.

In this area, Evertson (1995) describes the research about classroom rules and routines that teachers follow while teaching. He defines rules as general norms for expected behaviours and routines as procedures for accomplishing particular classroom tasks (how-to's). For the purpose of this study, routines appear to be more relevant, so far they could throw light on what teachers do while teaching.

Speaking about classroom routines, Evertson (1995) describes the results of a study by Leinhardt, Weidman, & Hammond (1987), that defines routines as systems of exchange that are set up to accomplish tasks, describing three types of routines:

- **Management routines**, that include housekeeping, discipline maintenance and people moving tasks.
- **Support routines**, that is, specific behaviours and actions necessary for a learning-teaching exchange to take place, for example 'how to pass in papers'.
- **Exchange routines**, that is, the interactive behaviours that permit the teaching-learning exchanges to occur. They govern the language contacts between teachers and students - for example, routines for choral responses.

- **Management routines**

These routines are described in the literature as 'Classroom Management' (Jones, 1996). In this review, Jones (1996) describes classroom management as "a vehicle for providing students with a sense of community and increased skills in interpersonal communication, conflict management, and self control" (p. 503). He argues that:

> All teachers care about students, and all teachers need to work with students to develop a safe and orderly classroom climate. Nevertheless, a central issue in defending classroom management will always be the manner in which the teacher chooses to develop safety and order.
>
> (Jones, 1996; p.505)

- **Support routines**

These are related to 'expected behaviours' that pupils and teacher know and follow. But from the teachers' point of view, these routines could be described as a set of tactics that users apply for teaching. For example, in Woods & Jeffrey (1996) they describe how teachers created 'atmosphere' and used different tones in interaction with the children to achieve different effects.

In describing the way teachers create this atmosphere, they describe it as having the following characteristics:

- **Anticipation and expectation**: teachers are skilled in the construction of situations and their sense of timing.
- **Relevance**: teachers were able to ensure that pupils felt a strong sense of involvement by making the pupils identify and get involved in the atmosphere.

- **Achievement and success**: there is a sense of high teacher expectations and confidence in children's abilities to meet them.
- **Satisfaction**: Pupils feel satisfaction and the sense of 'a job well done' because of the achievement.

These characteristics of a 'Classroom Atmosphere' are part of the techniques that teachers use during the lesson. Woods and Jeffrey (1996) also describe the different tones that teachers use during the lesson as "Andante (to be performed in moderately slow time)", "Legato (smoothly and connectedly, no gaps or breaks)" and "Spiritoso (with spirit)".

- **Exchange routines**

Exchange routines are related to the particular interaction with the pupils. These are described in the literature referring to the way teachers conduct dialogue with the pupils, the way teachers make questions to the students or give feedback to them.

About this dialogue, Hammersley (1990) argues that is based on a question-answers dialogue and describes the way teachers construct these dialogues. He says that teachers present the lesson "pitched at a certain level of 'difficulty' according to the co-ordinate position of the class in relation to age and ability" (Hammersley 1990, p. 47). He argues that teachers gradually provide more and more clues to 'what the answer is' in order to place the lesson in such a way as to get the answer to emerge towards the end of the lesson. Further on, he gives some reasons why they do this:

> This pre-setting is designed not only to ensure that pupils are taught something 'new', that they 'keep moving', but also that they have the resources to understand what the teacher is to teach.
>
> (Hammersley 1990, p.47)

As a complement to this characterisation of teacher-student dialogues, its components have also been studied, that is, questioning and giving feedback. About the former, Gall & Artero-Boname (1995) provide a classification of the types of teachers' questions during a lesson:

- Lower or higher cognitive questions
- Recitation and discussion questions
- Test-relevant questions

In general terms, they present evidence showing that teachers' questions were frequently closed and factual, made for recitation and similar to the ones that will

appear in the tests. Nevertheless, as they argue, in the early 90's contextual factors were brought into the study of this area and enabled the researchers to have different perspective about this finding.

The feedback provided to students has also been studied. Mayer (1995) describes the educational uses of feedback as corresponding to academic learning tasks, behaviour management tasks and skill learning tasks. In the first, feedback provides information concerning the correctness of students performance, in the second, it provides information concerning the appropriateness of students behaviour and in the third it provides information concerning the accuracy of students behaviour.

Presenting a different view about teacher-student interaction, Cooper & McIntyre (1995)'s report describes a study aimed at characterising teacher-student interaction that could lead to effective teaching and learning in the classroom. During their account they describe two types of teaching: interactive and reactive. The former is described as the teacher actively employing student perceptions to mediate their learning objectives. The latter is described as the teacher selecting the lesson content and teaching strategy based on his(her) perception of students' concerns or interests. Later on in the report they report that typically reactive teaching existed within a broader interactive context.

In yet a different view about teacher-student interaction, Edwards & Mercer (1987) argue that learning partially consists of making knowledge explicitly common to all members of the class/school and therefore this interaction has particular characteristics or 'ground rules', as they say. In addition to the general ground rules of conversation, they argue that in the classroom there are additional rules, such as:

- It is the teacher who asks the questions
- The teacher knows the answers
- Repeated questions imply wrong answers

These basic rules provide the general context where the teacher is the 'expert', in control of the development of knowledge through careful guidance and scaffolding.

While arguing this, they define what they call an 'Ideology of Teaching' that is based on six principles that teachers reported they followed while preparing a lesson. These principles are:

1. Setting up conditions which they believe would allow children to discover things for themselves.

2. Planning their teaching to include activities which would give children direct, concrete experience, and which would require then to act, not just listen, read or write.
3. Attempting to refer to children's wider out-of-school experience when planning curriculum topics (in the sense of 'general knowledge', but hardly ever by reference to the particular life experience of any one child in the group).
4. By the use of techniques like the 'guessing game' question and answer sessions, to elicit 'key' ideas from children rather than informing them of these directly.
5. Never defining (for the children) the full agenda of any activity or lesson in advance.
6. Not defining explicitly (for the children) the criteria for successful learning which would eventually be applied to what they had done.

(partial transcript of: Edwards & Mercer 1987, pp. 33-34)

As Edwards & Mercer (1987) say, these principles correspond to "a version of what is popularly called 'progressive' [teaching] -an approach normally contrasted with the 'traditional' one, which relies heavily on didactic methods and formal procedures" (Edwards & Mercer 1987, p. 35, our brackets).

From the point of view of this study, these principles also describe the techniques that teachers could use while interacting with the students and therefore they belong to the 'Exchange Routines' defined by Leinhardt, et al. (1987).

Summarising, while using the general framework of the types of classroom routines provided by Leinhardt, et al. (1987), it was possible to review some studies conducted that could help to provide a better idea about what teachers do in the classroom. It has been showed that what teachers do in the classroom could be classified at least in three different types of actions or routines (management, support and exchange). Each of these routines can be implemented using different techniques that were reported by different studies.

The particular techniques described in the different studies presented serve as examples for this study, so far it is not the intention here to judge which of them could be more or less effective or appropriate. The important point here is that there is a body of theory that describes, starting from different theoretical areas, what teachers do in the classroom and the fact that the different descriptions coincide in several dimensions of these actions, provides a basis to argue that they are accurate descriptions of what teachers do in a classroom. Nevertheless, it could not be argued that they cover all the possible actions of a teacher in the classroom neither that there could be alternative descriptions to these actions.

Then, what is more relevant for this study is the fact that these descriptions exist and provide a framework to discuss and analyse particular uses of computers and software while teaching.

2.6.3 Summary

In the scope of this study, through the short literature review about teachers knowledge and expertise it was shown that this is still an arena of debate, where different models for characterising such knowledge are still being developed. In this sense, these studies provide additional support for the need of research aimed at characterising teachers' professional craft knowledge and therefore to research what teachers believe about, for example, educational software.

On the other hand, it was shown that there are a large number of studies aimed at characterising what teachers do in the classroom. These studies provide a rich framework for discussing the role of the computers and software in the teaching profession.

2.7. CONCLUSIONS

The previous sections presented five themes:

- Software design - where it was concluded that designing software should incorporate elements from the reality in which it will be used. At present there is a lack of understanding of the role of the teachers and about the activities that occur with and around the software that is being used in the schools.

- Software development - where arguments for a modification of traditional software development methods, incorporating techniques such as ethnography in early stages of the process, were supported. This, in order to understand the users' professional activity (the teacher's activity in this study) and through this understanding being able to design appropriate software.

- Software evaluation - where a case was made for the incorporation of the contexts of use as a new dimension for evaluation of educational software, transforming it into a process that could be understood as qualitative research. This highlights the situated nature of software use.

- Educational innovation - was presented as a phenomenon that is highly correlated with the introduction of computers in schools. In this sense the role of computers

as support tools for such a process rather than as catalysts for it was emphasised. Therefore, and in order to be able to play this role, it is essential to understand the contextual setting in which it will be used.

- Pedagogy - the brief literature overview showed that an interesting body of knowledge about teachers' actions and 'routines' in the classroom has been already reported in the literature and can provide a rich framework to discuss and analyse the pedagogy of educational software.

These points all indicate the importance of knowing and considering the reality of use in order to design, develop and evaluate educational software that could be used to support an innovation process which engages the teacher and the school.

This review clearly demonstrates the need for further research in the area of educational software. Particularly, it highlights the need for knowing teachers' actual beliefs about computers and software in order to contribute to the understanding of this technology in education from a situated perspective. The main question that was supported is: what are teachers' concepts and beliefs about educational software?

III. METHODOLOGY

3.1. INTRODUCTION

The literature review presented in the previous chapter demonstrates a need for further research in the area of educational software. In response to this, the present research project was designed with the aim of improving the understanding of the concepts and beliefs that teachers can build-up about educational software and thereby construct a model about teachers' understanding of educational software.

During the design stage of this study, it was considered that neither a review of existing pieces of software (like the propositions made by Squires & McDougall, 1994) nor ethnographic studies of teachers' use of software (for example: Olson, 1988; Schofield, 1995; Fraser, et al., 1991) would be enough to enable us to do this because of teachers' misleading preconceptions about software and because of the constraints imposed by the actual software itself.

Considering this argument, and in order to investigate these issues, a situation was created in which teachers would need to think deeply and purposefully about the characteristics and particularities of educational software. The situation created was a process of educational software development, in which two teachers, a software engineer, a psychologist and a graphic designer, were committed to developing a piece of educational software in a period of time. This situation was observed, recorded and studied as a case (Stake, 1994).

The raw data for this research was not the software being developed, but the content of each meeting of the development team. In each of these meetings teachers expressed ideas and conceptions about educational software and were continuously reflecting on educational software. Because of power issues and preconceived concepts about educational software, the researcher was a non participant observer (or complete observer) (Adler & Adler, 1994) in each meeting. The data were gathered using video tape recording and transcripts of each session were analysed from qualitative and quantitative stand points, using systemic networks (Bliss, et al., 1983) to organise and give a structure to the categories of analysis.

From the point of view of its design, this research can be conceived as an instrumental case study in which this particular case was examined in order to provide insight into the issue or refinement of theory (Stake, 1994). In this situation the case itself is of secondary interest; because it played a supportive role, facilitating our understanding of something else, i.e. teachers' concepts and beliefs about educational software.

The following sections present some theoretical foundations about the methodology used, describe the research design and present the analysis methods used.

3.2. THEORETICAL FRAMEWORK

3.2.1 Case study methodology

As a form of research, case study is defined by Stake, (1994) as "not a methodological choice, but a choice of the object to be studied", he argues that "it is defined by the interest on individual cases, not by the methods of inquiry" (Stake 1994, p.236). He also differentiates the case from what is done in the case, that is, the object of study from the domain of action of this object. As Stake (1994) writes, "coming to understand a case usually requires extensive examining of how things get done, but the prime referent in case study is the case, not the method by which the case operates" (Stake 1994, p. 245).

An alternative conception of case study can be found in Yin (1994), where he attempts a technical definition, starting with the scope of a case study:

> a) Case Studies can be defined as an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident
>
> (adaptation of: Yin 1994, p. 13)

Here he highlights the importance of situating the inquiry in a real-life context as a distinctive characteristic of this kind of study compared with other forms of inquiry (for example, the experiment method– which deliberately divorces the phenomenon from its context).

As a second part of the definition, Yin (1994) describes:

> b) The case study inquiry copes with the technically distinctive situation in which there will be many more variables of interest than data points, and as one result relies on multiple sources of evidence, with data needing to converge in a triangulation fashion, and as another result benefits from the prior development of theoretical propositions to guide data collection and analysis.
>
> (adaptation of: Yin, 1994, p. 13)

With these additional definitions, he claims that a case study is a comprehensive research strategy, explicitly arguing against Stake's definition of it as an 'object to be studied'.

Stake (1994) classifies case studies in three groups:

1) **Intrinsic**: case study is undertaken because one wants a better understanding of a particular case (phenomena). Here the purpose is not theory building.

2) **Instrumental**: a particular case is examined to provide insight into an issue or refinement of theory. In this situation the case is of secondary interest; it plays a supportive role, facilitating our understanding of something else.

3) **Collective**: researchers may study a number of cases jointly in order to inquire into the phenomenon, population, or general condition. It is not the study of a collective but an instrumental study extended to several cases.

3.2.2 Critiques of case studies

Some of the traditional critiques of case studies are reported by Yin (1994) as:

- The lack of rigor of case study research.
- They provide little basis for scientific generalisation.
- They take too long and they result in massive unreadable documents.

The first point is one that the researcher using case studies must continually try to address, and introduce as much rigor as possible into the process. About the second point, case studies provide evidence not for populations or universal generalisations, but for theoretical generalisation. Another way of dealing with this problem is using triangulation techniques. The third point will depend on the method of inquiry used (ethnographic v/s other observation method, for alternatives see Ball, 1993; Hammersley & Atkinson, 1983 or Adler & Adler, 1994).

Despite these critiques there are several quality measures to judge a case study. These have been summarised by Yin (1994) as presented in Table 3.1 (found in Yin 1994, p. 33, Source: COSMOS Corporation).

Construct validity is specially problematic in case studies because of the prior uncertainty of what could be the focus of observation during the study of the case. At some point during the period of data collection it is possible to have a better understanding of the case and therefore it is possible to define a set of measures to be taken. The problems here are twofold:

- It is not clear at which stage of the data collection period this should be done and
- that it could happen that the operational measures are chosen because they validate the types of change the researcher is trying to demonstrate (i. e. these changes are produced by these operational measures) or the other way around (i.e. these set of operational measures produce this change and that is the one I will observe).

The second problem mentioned is that the researcher will be using something like grounded theory (Guba & Lincoln, 1994) approach for these definitions and in this sense these definitions will constitute the research itself.

Tests	Case Study Tactic	Phase of Research in which Tactic Occurs
Construct Validity: establishing correct operational measures for the concepts being studied	• use multiple sources of evidence • establish a chain of evidence • have key informants review draft case study report	• data collection • data collection • composition
Internal Validity: establishing a causal relationship, whereby certain conditions are shown to lead to other conditions, as distinguished from spurious relationships	• do pattern-matching • do explanation-building • do time-series analysis	• data analysis • data analysis • data analysis
External Validity: establishing the domain to which a study findings can be generalised.	• use replication logic in multiple-case studies	• research design
Reliability: demonstrating that the operations of a study -such as the data collection procedures- can be repeated, with the same results.	• use case study protocol • develop case study data base	• data collection • data collection

Table 3.1. Case study tactics for four design tests

Internal validity is used only for causal (or explanatory) case studies where the investigator is trying to determine whether event x led to event y.

External validity is one of the critiques mentioned before in terms of the possibility of generating a 'universal theory' based on the findings from a case study. Some authors claim that case studies provide evidence for theoretical or analytical generalisation (like Yin, 1994) and others claim for naturalistic generalisation (like Stake, 1994 referencing Stake & Trumbull, 1983). It could be interesting to explore the notion of abduction in logic in order to analyse the possibility of generalising the case.

Reliability in case studies is probably not possible in the sense of 'experiment replication', the point here seems to be in the replication of the findings and conclusions drawn from a particular case. In this sense, the researcher should provide enough information about the process of analysis such that someone else could follow his(her) reasoning/interpretation strategy/path.

3.3. RESEARCH DESIGN

3.3.1 General design

In order to design a piece of research as a case study, Yin (1994) suggests that the researchers should define the following points:

i) Study's questions: In case study, these questions are generally: how or why.
ii) Study propositions: It means the focus of the research, and if there are no propositions, it should have a purpose as well as the criteria by which it will be judged successful.
iii) Units of analysis: It implies the definition of the 'case' itself. It is related to the way the initial research questions have been defined and will define what will be studied.
iv) Linking data to propositions: It is not well defined in the literature of case study research. Apparently it will depend on the nature of the case and the kind of data recorded.
v) Criteria for interpreting the findings: Probably this will be part of the process of analysis and it will become clear in a late stage of the research.

In this research, these are:

i) **Study's questions**:

The main question of this research is: What are teachers' concepts and beliefs about educational software?, which can be expressed in the following questions:

a) What do teachers refer to while designing educational software?
b) What is the nature of the software's features (software characteristics, roles, uses, etc.) that teachers design?
c) How do these elements match with existing descriptions?
d) What are the implications of such designs?

ii) Study propositions

a) To identify the elements of educational software that were referred to during the design process. This first study proposition was aimed at looking for the different kinds of educational software elements that the development team expressed during the process. The classification of the elements to which they refer was:

- Characteristics of the software:
  - Subject Areas
  - Content Organisation
  - Browsing
  - Interaction
  - Interface Element
- Pedagogic issues:
  - Aim
  - Teaching Strategy
  - Actions
- User

This classification of the elements resulted after several revisions of the data and constituted the basic categories to build the systemic network used in the analysis.

b) To examine the shape of the elements of educational software. This second proposition was aimed at examining the characteristics of the specific elements designed by the development team, focusing on teachers' expressions.

c) To quantify the distribution of participation in the design process. This third study proposition was aimed at quantifying the 'amount' of participation of each member of the development team in each of the categories, looking at the distribution of frequencies. This was done in order to quantify and compare teachers' participation during the process.

d) To identify patterns of design. This fourth study proposition was aimed at finding patterns of sequences of categories that were spoken by the development team (as a group) and by each member (as individuals). This was done in order to define new semantic categories that could carry different meanings.

e) To compare the elements found with reported descriptions about educational software, about its use and about its role in the classroom.

iii) Units of analysis

The unit of analysis was defined as a sequence of one or more utterances made by one or more members of the team in which they refer to one particular aspect of the software. Where they referred to two (or more) aspects simultaneously (in parallel), these have been considered as two (or more) separate units. The unit was called 'Discussion about Software Design' and was described through a systemic network.

iv) Linking data to propositions

The data gathered consisted of transcripts of the meetings that were recorded using video camera and a sound recorder. These transcripts were inspected and then categorised in a systemic network that was developed during this first stage of the analysis process. Using the systemic network as the basic structure, the data were analysed using three different techniques:

- Participation analysis (quantitative): Aimed at establishing individual participation profiles during the development process, based on the calculation of the frequencies and distributions of participation of each team member in each category.
- Sequences analysis (quantitative): Aimed at establishing the inter-relations among the different components of the software, based on the analysis of the patterns of sequences of units in the data for the group and each team member.
- Contents analysis (qualitative): Aimed at looking for the meanings expressed by the teachers about the different elements of a piece of educational software.

v) Criteria for interpreting the findings

The process for interpreting the findings was first to develop a theoretical model of these teachers' understanding of information technology (theoretical perspective) and then compare it with previous models and hence show where it differs from previous models (practical perspective). The theoretical framework used in each perspective was:

- From a theoretical perspective findings were interpreted using existing theories about the use and role of educational software in classroom settings (for example: Fraser, et al., 1991; Olson, 1988; Sandholtz, et al., 1997; Schofield, 1995) and the characteristics of the software were contrasted with existing descriptions about what teachers do in the classroom (for example: Cooper & McIntyre, 1995; Edwards & Mercer, 1987; Hammersley, 1990; Woods & Jeffrey, 1996) and about their professional knowledge (for example: Alexanderson, 1994; Elbaz, 1990; Eraut, 1994; Marton, 1994).

- From a practical perspective findings were compared with existing conceptions about educational software design (for example: Lajoie & Derry, 1993; Laurillard, 1993; Orhun, 1995; Taylor, 1980; Watson, 1993), software development (for example: Char & Hawkins, 1986; Hawkins & Kurland, 1986; Watson, 1987; Watson, 1993) and software evaluation (for example: Squires & McDougall, 1994).

3.3.1 Case selection

In order to carry out this research it was necessary to find a development team that could be the source of the data to analyse, i.e. a group of persons willing to develop a piece of educational software under observation.

Based on previous research (Hinostroza, et al., 1996), it was decided that the members of the group should be:

- Two teachers: Teachers are the essential part of this research, thus, the aim was to analyse, mainly, what they said about educational software. The reasons for having two teachers were, first to minimise the risk of desertion and second, to provide a better scenario for discussion about the different software elements to be designed. A third requirement was that these teachers were experienced information technology users with some formal background in information technology in education. This requirement was imposed to provide a realistic view on what can be done with computers in the classroom and an informed view about its technological and methodological possibilities.

- A Software Engineer: The software engineer was responsible for the programming and technical design of the software. One requirement was that (s)he should have the technical qualification but little experience of developing this type of project. The rationale in this case was that the engineer should not bias the development project drawing on previous experience of developing similar projects.

- A Psychologist: The Psychologist was responsible for providing theoretical support in aspects of the software development such as learning theories, cognitive development of the potential users, etc. It was also a requirement that the Psychologist was also had little experience in participating in software development projects.

- A Graphic Designer. This person was in charge of the design and illustration of the interface elements of the software. Again, it was required that (s)he had little experience in software development, so (s)he could stay free of biasing this particular design. The participation of the Graphic Designer was considered to be optional at the beginning of the research due to the lack of personnel available. Therefore she started to participate in the development process at session 14 of 19 sessions.

Following these considerations, the group was constituted of two school teachers and by three undergraduate university students. The university students were studying Software Engineering, Psychology and Graphic Design (visual arts). These students were advanced in their careers but had not yet graduated. They were hired as research assistants to participate in the software development process[6].

In order to select the teachers, the first step was to select a school that would satisfy the following requirements:

a) The school should have at least two teachers participating in the diploma course of the Universidad de la Frontera[7] about Information Technology in Education[8]. The rationale behind this was to have teachers with a formal background in information technology in education in order to focus on software development and not on general issues of information technology in education.

b) The school should have at least two years of experience participating in the Enlaces project (more about Enlaces in: Hepp, et al., 1994; Potashnik, 1996). This restriction was imposed in order to assure a certain amount of experience in the use of information technology in education and to avoid having the school involved in an early stage of an information technology innovation process.

c) The school should not have explicit religious or sectarian ascription of any kind. This restriction was included to avoid possible doctrinaire bias.

d) The school should be from Temuco, Chile. This was a practical reason to avoid long distance travelling out of the city or the country.

At the time the research started, there were only two schools that met these requirements. The schools were:

Vista Verde: In 1996 it was a rather new school with a population 430 students and 16 (female only) teachers, 2 of them were in the diploma course. It is located in a low social class sector of Temuco. This school had seven computers acquired in 1993.

---

[6] Funds for this were provided by the research project #1960854, of the Chilean research agency, Fondecyt.

[7] This is the University were the researcher actually works.

[8] The diploma course started in March 1995 and finished in January 1996. From March 96 to June 96 students had to do a project in their respective schools, applying what they had learned.

Standard: This is a rather old school that in 1996 had a population of 420 students and 22 teachers, 5 of them were in the diploma course (including the head teacher), all are female. It is located in a low-medium social sector of Temuco. This school had seven computers acquired in 1993.

After this initial definition an unstructured interview (following Fontana & Frey, 1994's guidelines) with the head teacher of each school was conducted to explore the feasibility of carrying out the research project in one of the schools and to be able to get a feeling of the internal climate of each school. The interviews are presented in Appendix A1.

Comparing these interviews, it was possible to conclude that the 'Standard' school had a more authoritarian style of direction compared with 'Vista Verde' school. Also, 'Vista Verde' showed a positive disposition to be part of the project whereas 'Standard' set up conditions to be part of it. Stemming from this background information, 'Vista Verde' school was chosen. The selection of the teachers in this case was then determined by the fact that only two teachers of this school fulfilled the requirements.

3.3.2 Development of the case

In order to start the development of the piece of software an initial explanation of the aims and goals was given by the researcher. For the development team, the aim was to develop a piece of software and during the process, they would be engaging in an 'Action Research' project (Reason, 1994). In this sense, it was not revealed that their expressions and opinions would be used as raw data for a different analysis. As mentioned before, from the research point of view, the aim of this process was to elicit concepts and ideas about educational software and the development process was used as instrument (a created situation) to elicit such ideas. In this framework, the presence of the researcher during the meetings was justified arguing that he would observe what was happening and was responsible for recording the information only.

This decision to not inform the development team of the final purpose of the project, was taken to release the pressure the teachers could feel if they realised that every opinion would be transcribed and then analysed. In this sense, the focus of pressure was the software product, not the process. Another reason was to avoid the situation where they would focus on finding arguments and ideas that would justify their beliefs about and actions with the software. In this sense, it was decided that a

realistic perspective[9] of obtaining the data would be more effective and closer to their actual beliefs, compared to a process of reflection or interviews.

From an ethical point of view, this decision could be questioned considering 'the principle of informed consent', described by Barret (1995), as:

> Wherever it is possible, researchers should inform participants in psychological research of all aspects of that research which might reasonably be expected to influence their willingness to participate in that research; in addition, researchers should normally always explain any aspect of the research about which a participant inquires.
>
> (Barret, 1995, p 30)

In this case, the participants agreed to be recorded, and knew that this data would be afterwards analysed. The difference was the focus of analysis of the data, instead of looking at the software product, the researcher would look for their ideas about educational software. At the time this decision was taken it was considered that, this difference, between what was informed and what was actually done, would not change the possible consequences for the participants, and therefore, would not influence their willingness to participate in this research.

In order to check this assumption, the researcher conducted a meeting with the teachers that participated in the study (September 1998). The aim of the meeting was to present and discuss the main findings. After this discussion they gave full permission for dissemination of the results.

Another important consideration was that the development team should be able to end up with a product, or at least with a prototype of the software, because in this way they would address all the aspects of a piece of software, through completing a development cycle. In this way it was assured that they would have the opportunity to express all their ideas about the different aspects of a piece of educational software. For this purpose, they decided to follow the general guidelines of the software development method reported in Hinostroza, et al. (1996).

The decision to follow this method was taken without explicit arguments, but it seems reasonable to assume that at least two factors were implicitly considered, first that they had some experience in using the method (in the diploma course) and second, that the researcher was one of the authors of the method. Although this could indicate

[9] It is realistic because the development team was engaged in a 'real' activity, that is they believed that they would develop a piece of educational software, so all the conversations, opinions and discussions were purposeful and therefore authentic. So, what was recorded was a real situation, from which the data was extracted.

some power issues and therefore a source of bias, the method itself is not influential in the contents or decisions of the development process, it simply proposes a set of stages to be developed and, even if it would do so, the development team did not follow the method exactly.

The development process consisted of 19 meetings of approximately 90 minutes each. 16 meetings were conducted at the teachers' school and because of practical reasons (that is, in order to review a prototype) the last three meetings were conducted at the Universidad de La Frontera. The first meeting was on April the 18th 1996 and the last meeting was on November the 18th, 1996 (a seven month period).

3.3.3 Data collection

The data collection was performed using a video camera and a sound recorder. The camera was installed in a fixed position during each meeting and the focus of the image was always on the teachers. The sound recorder was left over the table.

During the meetings the researcher adopted the role of a non participant observer (Adler & Adler, 1994), trying not to interfere with the natural flow of each meeting. Nevertheless, in some meetings it was not possible to keep to this role because of direct questions from some members of the development team (asking for the researcher's opinion about different topics). It was also noticed that at some points some members of the team expressed ideas and opinions looking at the researcher and, in some way, asking for approval. In this sense, the 'invisibility' of the observer can be questioned, but considering the relative small number of interventions it was assumed that it would not bias the results. Also, during the initial meetings the researcher took notes about what was said, but it was realised that both teachers observed with great curiosity what was being written, so it was decided to stop taking notes.

The data collected this way consisted of 19 super-8 video cassettes (120 minutes each) and 19 audio cassettes (90 minutes each). After several attempts to analyse the data contained in the video in its original audio-visual format (for example following Erickson & Wilson, 1984 guidelines), it was decided that for the purposes of this analysis the corporal expressions and gestures would not be considered, because the oral expressions were enough to capture the meanings expressed and in the few situations were these expressions did complement the meaning, the transcript was annotated. So, all tape dialogues were transcribed to text in their original language, i.e. Spanish (audio tapes were used to support video tapes' audio and vice versa). The

transcription process was done by the researcher and some hired persons who received instructions about the coding standards that should be used[10].

The transcript conventions used were rather simple, because for the purpose of this analysis it was not necessary to include all the expressions (facial, gestures, etc.) of the members of the development team. The convention was to use the marks '<' and '>' to include relevant comments on what was happening. That is, if one person expressed an idea about the software ironically (like making a joke) or if someone made a gesture to complement an oral expression, it should be explicitly marked in the transcripts (for example, if the size of something was indicated using the hands), but if one simply looked in another direction, it was not necessarily annotated (unless this gesture would change the meaning of what (s)he was expressing). Also, the overlapping of dialogues in some discussions were explicitly marked, but not rigorously. So, there are a small number of written comments in the transcripts. Because the transcripts were done by external persons, the researcher reviewed all the transcripts, complementing and/or amending where required. The data transcribed to text is presented in table 3.2 and represents the raw data for the analysis.

Characters:	985,832
Words:	192,391
Lines:	19,167
Paragraphs:	12,619
Pages (A4, single space, times 12)	387

Table 3.2. Figures of the raw data

Because of the amount of data, a qualitative analysis software was used to support the analysis. After a short review of alternatives, the software QSR NUD*IST was selected because of its possibilities to have dynamic reconfiguration of the category tree (move one branch or node to another position), the possibility of programming and executing complex commands and the text search functionality.

The text (of the transcripts) was 'inserted' into QSR NUD*IST considering each session as a document and defining the unit of text as a paragraph (usually one paragraph corresponded to one utterance of one of the members of the development team). The alternative of defining the unit of text as a line was not chosen because the definition of the unit of analysis considered dialogues that could consist of several lines and paragraphs in which one particular aspect of the software was designed.

[10] Funds were partially provided by the research project #1960854, of the Chilean research agency, Fondecyt.

3.3.4 Categorisation process

The categorisation process consisted of two stages, the definition of the categories and the coding using the software for qualitative analysis QSR NUD*IST.

3.3.4.1 Definition of the coding categories

In order to categorise and give structure to the data a systemic network (Bliss, et al., 1983) was designed. The process of defining the systemic network, that could well represent all the categories of data, was done following a refinement process. This process is described in the next chapter which also explains and discusses the coding categories defined. The resulting network is presented in figure 3.1.

This network describes the two main attributes that characterised the unit of analysis. These are: the 'Topic' discussed (to answer: what are they speaking about ?) and the 'Participants' in the discussion (to answer: who is taking part in the discussion ?).

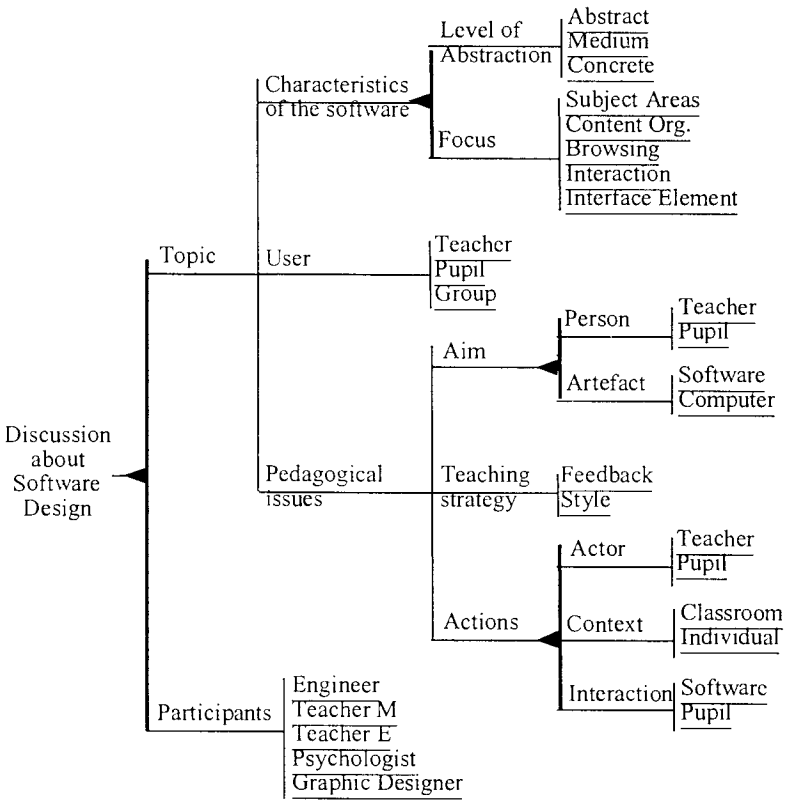

Figure 3.1. The systemic network representing the coding categories

The 'Topic' of discussion was divided in three main subjects related to the software design. The first one was related to the dimensions of software engineering ('Subject Areas', 'Content Organisation', 'Browsing', 'Interaction' or 'Interface Element'); the second one was about the characteristics of the software's user ('Teacher', 'Pupil' or

'Group'); and the third one dealt with 'Pedagogic Issues' that were related to software and/or computers ('Aim', 'Teaching Strategy' or 'Actions').

The rationale behind this design was to be able to separate the more concrete ideas about software from the more abstracts ideas about how to use the software or computer in the classroom or individually, who they imagine will use the software and the aims of using this resource. In other words, the classification used was designed to gather data concerning to the following areas:

1 The software engineering characteristics of the educational software designed.
2 The conceptions that teachers have of the users of the software.
3 The beliefs that teachers have about the aims of software and computers for themselves and for the pupils.
4 Teaching issues that are reflected in the software design process and could become part of the software.
5 Actions that include data regarding the roles that teachers attribute to themselves and to the pupils while using the software in the classroom (either individually or while orchestrating the lesson).

The data, classified in each of these categories in its original language (i.e. Spanish), were the source for the different analyses carried out, and thereby, enabled the understanding of some aspect of teachers' conceptions about educational software, and given the aim of this research, to a lesser degree the conceptions of the other members of the development team.

3.3.4.2 Data coding

The systemic network was used as the 'category tree' of the qualitative analysis software used to support the process (QSR NUD*IST). This was possible because the resulting network was simple enough to do so, that is, it did not have recursions or simultaneous entries (Bliss, et al., 1983).

Nevertheless, some adaptations were required, because QSR NUD*IST does not differentiate between 'AND' and 'OR' branches as can be found in a systemic network, therefore the process of categorising the data was done following a self imposed convention. That is, for example, if a unit of analysis was defined as 'Topic: Characteristic of the software: Focus: Browsing' it was also classified by its 'Level of Abstraction' (for example: 'Topic: Characteristic of the software: Level of Abstraction: Medium'). In this way it was possible to use the same model of the systemic network in the software.

The process of coding the data was carried out in three steps:

i) The first step was to read all the data in order to pre-select the dialogues that matched each category. This process was done on paper and helped to develop the definitive categories of the network.

ii) The second step was to revise the data again, and re-assign dialogues to the definitive categories. This process was done using the software QSR NUD*IST. This second review process was done intensively (as continuous as it was possible), in order to keep freshly in mind of the researcher the non-written criteria of assignment.

iii) The third step was to print out all the transcripts organised by the categories defined in the systemic network (using the reports facilities of the software) and revise each category again in order to identify possible errors in the assignment or miss-interpretations of the dialogues.

With the described coding procedure, 848 units of analysis were classified in the different categories which constituted the raw data for the analysis.

3.4. ANALYSIS PROCESS

After the data were coded in some category of the systemic network, three types of analysis were carried out, as described earlier, these were: 'Participation Analysis' (aimed at establishing individual participation profiles during the development process), 'Sequences Analysis' (aimed at establishing the inter-relations among the different components of the software) and the 'Contents Analysis' (aimed at looking for the meanings expressed by the teachers about the different elements of a piece of educational software). The combination of qualitative and quantitative analysis methods enabled the production of an overall interpretation which might be more convincing than if only one type of analysis method were used (Wegerif & Mercer, 1997).

In order to reduce the possible loss of meanings or misinterpretations during the translation process, the analyses were done using the original transcripts in Spanish[11].

[11] The researcher is Chilean, speaks Spanish and lives permanently in Chile, he learned English rather late in his life, and it was therefore felt that any translation at this stage might have resulted in some loss of meaning.

Only selected dialogues were afterwards translated into English to include them in this thesis.

3.4.1 Participation analysis

The frequency of units spoken by each member of the development team was obtained using the software QSR NUD*IST, by searching for the name of each member of the development team in all the documents. The resulting set was then intersected with the sets defined by the systemic network (categories). The result of this intersection produced the distribution of spoken units in each category of the systemic network for each member of the development team. The number of units spoken in each category by each member of the development team was not relevant for this analysis because it did not correspond to the number of units of analysis used, what was relevant was the distribution of these units (percentage).

These frequencies were transformed into percentages of the distributions relative to: (i) their participation in a group of categories ('Characteristics of the software', 'User', 'Aim', 'Teaching Strategy' and 'Actions'), and (ii) relative to each member's total units spoken. The aims corresponding to each calculation were:

i) To compare the contributions of each member of the development team to each group of categories. In other words, this analysis provides an answer to questions like: How much did Teacher M contribute to the group of categories 'Characteristics of the software' compared to the contributions of the other participants ? or How much did Teacher M contribute to the category 'Abstract – Interaction' compared to the contributions of the other participants ?.

ii) To compare the distribution of the units spoken by each member of the development team in all categories. These distributions are called here 'profiles of participation' and were calculated for each member of the development team. The aim of this analysis was to answer questions like: Did the teachers participate with similar emphasis on the different categories (i.e. show similar profiles)? or Did the teachers participate in a different way compared to the Software Engineer (i.e. show different profiles) during the development process ?.

The hypothesis for this analysis was that members of the development team would show a differentiated contribution to each group of categories and therefore they would show different 'profiles of participation' in the development process. And the underlying assumption was that these profiles would be related to their professional

backgrounds. This, because they would participate more frequently when discussing topics (i.e. categories) that were more interesting for them and would participate less frequently in those topics that they were not so interesting.

If this hypothesis was proved to be right, the profiles of participation of the teachers would point out their concerns about educational software (i.e. those categories in which they contributed 'differently'). and would serve as a guide for the other analyses, showing the categories that could be of more interest. As it will be shown in Chapter V, the hypothesis was proved to be right for some groups of categories.

The analysis also served as a validity check of the research process that ensures the teachers' participation and therefore supports the argument that the results corresponds to the teachers' expressions.

3.4.2 Sequences analysis

In this analysis, the development process was considered to be a string of 848 units of analysis that represented one set of data. In order to have a better profile of each member, individual contributions during the development process were separated and thereby it was possible to obtain one string of units for each member. Through the process of separating individual contributions the following number of units for each member (or 'strings length') were obtained:

Member	Number of Units
Teacher E	2,038
Teacher M	2,414
Psychologist	1,894
Engineer	2,498
Graph Designer	819
Group	848

Table 3.3. Total units per member

The higher number of units of the individual participation of each member is due to the division of the dialogues. That is, one dialogue between several members could have been classified as one unit of the group, but, when looking at their individual participation the same unit could represent three or more units of each member, depending on the number of interactions occurred. Because of the difference in the number of units and her late entrance to the development team, the Graphic Designer was not included in this analysis.

For the purposes of this analysis a 'sequence' was defined as a consecutive set of 'n' different units that were present in a string of units more than once. The process of

searching for sequences was done using a piece of software implemented by the researcher (using the programming language 'C'). This software accepts as input a string of units and looks for sequences of different lengths in the input string.

Using as input for the software each member's string of units (Group, Teacher M, Teacher E, Psychologist and Software Engineer), a search for 3 to 30 units long sequences was carried out, but no relevant additional information was found after 9 units long sequences. That is, 3,4,5,... or 9 consecutive units that constitute the sequence in each string.

For example, consider the following string of units (each letter represents one unit of analysis and the different letters represent different categories defined in the systemic network):

n b a d f s b m c c c c c a d f j e j a d f s e e e e a d f s

In this string, the 3-units long sequence 'a d f', is repeated 4 times (underline) and the sequence 'd f s' is repeated 3 times (strike through):

n b a ~~d f s~~ b m c c c c c a d f j e j a ~~d f s~~ e e e e a ~~d f s~~

There is also a 4-units sequence 'a d f s' that is repeated 3 times:

n b a d f s b m c c c c c a d f j e j a d f s e e e e a d f s

In the same example the sequence of units 'c c c' can be found 3 times (i.e. strike through, underline and bold):

n b a d f s b m ~~c c c~~ **c c** a d f j e j a d f s e e e e a d f s

This latter kind of sequence was considered apart from the former example, because it means that the person talked about 'c' during the whole time and did not change the subject. In the string that represent the units of the Group this sequence could appear as only one 'c', but in the process of separating each member's contribution, it is transformed into five (or more) 'c's, depending on the number of interventions of other members of the development team.

After identifying the different sequences, a new set of categories was defined in order to make groups of sequences. These new categories were based on the units that compose each sequence. That means, sequences that share the same set of units but in

different orders correspond to the same category. For example, the sequences 'abc', 'bac', 'bacab' and 'cababcbcb' were considered to be in the category 'a-b-c'.

The hypothesis for this analysis was that some of the categories defined in the systemic network have a closer relation to each other and that this relation provides additional meanings (the particular meaning of each relation was not analysed at this stage due to the very large number of possible combinations). This hypothesis was proved to be right and the results of this analysis are presented in Chapter VI.

3.4.3 Contents analysis

The analysis of the contents was done revising the transcripts attached to the categories of the systemic network in the branch 'Topic'. The resulting categories from the combination of the AND nodes of the branch 'Topic' of the systemic network and the number of units spoken by the group are shown in table 3.4.

Table 3.4 shows that 20 units were classified as the development group speaking about characteristics of the software 'Subject Area' at an 'Abstract' level of abstraction and 32 about the same focus but at a 'Medium' level of abstraction, and so on. Each of these units could be a single assertion of one the members of the development team or a several pages dialogue where they defined some specific aspect of the software.

Because of the small number of units found in some categories, these were grouped into one unit, for example, the units about the 'User' were grouped in the unit 'Pupil' (number 16) and the units in the categories of 'Teaching Strategy' were grouped in the category 'Feedback' (number 29). The dialogues classified under each of these 29 categories were analysed looking for evidence that could by related to some conceptual framework and looking for internal contradictions or discussions among the members of the development team. The process of revising and searching for meanings in each category was done several times in order to refine and complement the emergent concepts in each revision. Afterwards, these concepts were refined and are presented in Chapter VII, including examples of the some 'representative' dialogues. These dialogues were translated into English, including additional words between '[' and ']' brackets in order to preserve the meanings of the dialogues in Spanish.

#	Category	Units
	Characteristics of the Software	**583**
1	Subject Areas - Abstract	20
2	Subject Areas - Medium	32
3	Subject Areas - Concrete	19
4	Content Organisation - Abstract	22
5	Content Organisation - Medium	48
6	Content Organisation - Concrete	38
7	Browsing - Abstract	16
8	Browsing - Medium	42
9	Browsing - Concrete	46
10	Interaction - Abstract	43
11	Interaction - Medium	62
12	Interaction - Concrete	70
13	Interface Element - Abstract	19
14	Interface Element - Medium	39
15	Interface Element - Concrete	67
	User	**57**
	Teacher	1
16	Pupil	52
	Group	4
	Pedagogic Issues	**208**
	Aims	**77**
17	Teacher-Software	23
18	Teacher-Computer	14
19	Pupil-Software	28
20	Pupil-Computer	12
	Action	**79**
21	Teacher - Classroom - Software	7
22	Teacher - Classroom - Pupil	4
23	Teacher - Individual - Software	14
24	Teacher - Individual - Pupil	8
25	Pupil - Classroom - Software	3
26	Pupil - Classroom - Pupil	9
27	Pupil - Individual - Software	24
28	Pupil - Individual - Pupil	10
	Teaching Strategy	**52**
29	Feedback	50
	Style	2
	Total	**848**

Table 3.4. Categories and number of units spoken

3.5. DISCUSSION OF THE METHODOLOGY

Defining this research as a case study is not a clear choice, in so far as there are different definitions that lead to different conclusions. On the one hand, based on Stake (1994)'s definition, it is possible to call this research a case study, because it examines a bounded system, *i.e.* five persons that meet once a week with the common goal of producing a piece of educational software. On the other hand, based on Yin (1994)'s definition, it is possible to argue that this research is not a case study,

because it does not investigate a contemporary phenomenon within its real life context. That is, the analysis of teachers' understandings of educational software is within a context of a software development process and not in a context of reflections about software, neither is this a 'natural' situation, because it was provoked by the researcher.

The tension between these two definitions is what motivates this discussion. It would indeed even be possible to take a more extreme position and to call this piece of research an 'experiment', where a group of people (subjects) where exposed to a stimulus (i.e. to develop a piece of software) and their reactions were recorded (that is the recorded output were the ideas and claims that came out during this process).

It is possible to argue that, given that a certain process was happening (no matter who initiated it), the researcher was 'observing' and recording this situation and in that sense doing naturalistic observation (Adler & Adler, 1994). This observational study is similar to other classroom observations (like: Fraser, et al., 1991; Hammersley, 1990; McNamara, 1994; Olson, 1988; Schofield, 1995; Woods & Jeffrey, 1996), where the object of analysis is some dimension or characteristic of what is happening in the situation. In the present case, the focus of analysis was the teachers' talk about educational software characteristics. Therefore, despite the fact that the situation was created, this piece of research can in fact be called a case study.

IV SYSTEMIC NETWORK DEFINITION

4.1. INTRODUCTION

This chapter describes the design and definition of the systemic network used to define the categories of analysis of the data. The network was developed following the guidelines and conventions described by Bliss, et al., (1983). The purpose of this network was to provide structure and coherence to the data contained in the transcripts, in order to carry out the analysis process.

The network itself describes the unit of analysis defined for this research and constitutes an additional product of the research process. Although the research used a software development process as a means to elicit data, the network does not, necessarily, describe such a process. It only describes the different categories of data that were considered to be relevant for the aims of this research. That is, the network is not intended to be used as a comprehensive characterisation of a software development process. It represents however the set of (relevant) topics that were considered during the design of a piece of educational software and therefore, it can be considered as an accurate structure that represents the categories of teachers' beliefs about educational software.

The following sections describe: the process of definition of the network, its general structure, the definition of each category and the main difficulties that arose whilst using the network categories. Finally it presents a discussion of the network designed.

4.2. CONSTRUCTION OF THE SYSTEMIC NETWORK

In order to categorise and give structure to the data a systemic network (Bliss, et al., 1983) was developed. The initial version of the network was designed containing most of the technical aspects to be considered during the process of software design. It consisted of a hierarchical list of characteristics of a piece of software described in literature about human computer interface design (for example Laurel, 1990, Thimbleby, 1990) and containing most of the dimensions (learning models, psychological models, didactic models) described in literature about educational software development methods (for example: Galvis, 1994 and Hinostroza, 1994). The initial version therefore represented a rather theoretically driven description of the unit of analysis. The reason for this design choice was to ensure that it would be possible to describe through the network the greatest possible amount of the technical aspects that might appear in the data.

The initial version of the network was used as a starting point of a process of refinement of both the concepts contained in the network and the organisation of these concepts (i.e. the network, branches and leaves). This process was carried out reviewing examples of the data and coding them using the network. After the network seemed to be stable, all the data was coded using the software QSR NUD*IST (as explained in section 3.3.4). Then, in order to make further refinements the data attached to each leaf of the network was printed and re-examined, relocating the data that was considered to belong to a different category. The network was further refined at this stage and the data was moved from one branch to another using the functions provided by the software.

In general terms the process of defining the systemic network, that could well represent all the categories of data, was done following an iterative refinement process. This process considered two dimensions of progression:

a) A qualitative dimension, that considers the concepts expressed through the network, including the internal consistency and the general coherence. The internal consistency of the network is defined here as the similarity in the meanings of the categories grouped in one branch of the network. That is, in order to be internally consistent the elements grouped in one branch should correspond to the same category of meaning. General coherency was defined here as the degree in which the meanings of the different nodes located in similar levels of the network should express similar levels of abstraction. During the design process, this dimension showed a convergence by the redefinition and relocation of nodes and branches in the network.

b) A quantitative dimension, which is expressed in terms of the number of end-nodes and level of deepness of the network. This dimension is useful to measure the complexity of the network. During the design process, this dimension showed first an increment (divergence), increasing the number of end-nodes and the level of deepness and then the complexity decreased (convergence).

These dimensions of analysis of the systemic network are briefly described in this section, presenting four stages of the network design process. These were:

i) The first version of the network had 55 end nodes (or leaves) and 9 levels of deepness. This version of the network had many concepts grouped without a clear conceptual similarity or difference among them. This resulted in low internal consistency because the elements grouped in one branch referred to different

categories and it was not coherent, because the meanings of the different nodes (located in similar levels of the network) expressed different layers of abstraction.

ii) The second version of the network was more complex than the first one, it had 88 end-nodes (63% more than the previous one) and has the same 9 levels of deepness. Additionally it showed two other small networks that were used to classify data that were not included in the main network. Compared to the previous one, this version of the network showed a convergence in conceptual terms and a divergence in the granularity of the concepts constituting end-nodes. The convergence could be appreciated by the fact that the previous branches were now included in the right branches and some AND nodes were included, helping to visualise the relation between branches.

iii) After this divergent process in terms of complexity, a new version of the network was developed, this new version showed a dramatic decrease in its complexity, having only 23 end-nodes (25% of the previous one) and 6 levels of deepness (66% of the previous one). The categories in each branch were refined and the network showed a higher degree of coherence, that is, the intermediate nodes expressed concepts in similar layers of abstraction and the network also showed a higher internal consistency, in so far as the concepts in one branch refer to comparable characteristics. The main changes were that in the new network some previous end-nodes were subsumed into the prior hierarchical branch and that several concepts were grouped into one branch, seeking for a higher internal consistency.

iv) Finally, the definitive version of the systemic network is presented in figure 4.1. This version of the systemic network has 28 end-nodes and has 6 levels of deepness. In this sense the network has a similar degree of complexity to the previous version. The main changes in this case were that some nodes and branches were eliminated, that some AND nodes were extended, including three branches (for example 'Actions') and that some new hierarchies of classification were added or renamed. These changes gave a higher degree of coherence in so far the as nodes at each level of the branches refer to comparable concepts (the exemption is the branch "User", which does not accomplish this requirement). In terms of internal consistency, it is similar to the previous version.

In the systemic network shown in figure 4.1, the AND nodes are represented by the thick vertical lines with an arrow (representing a '{' bracket) and the OR nodes are the thin vertical lines (representing a '[' bracket). The next section of this chapter contains the operational definition of the categories contained in the network.

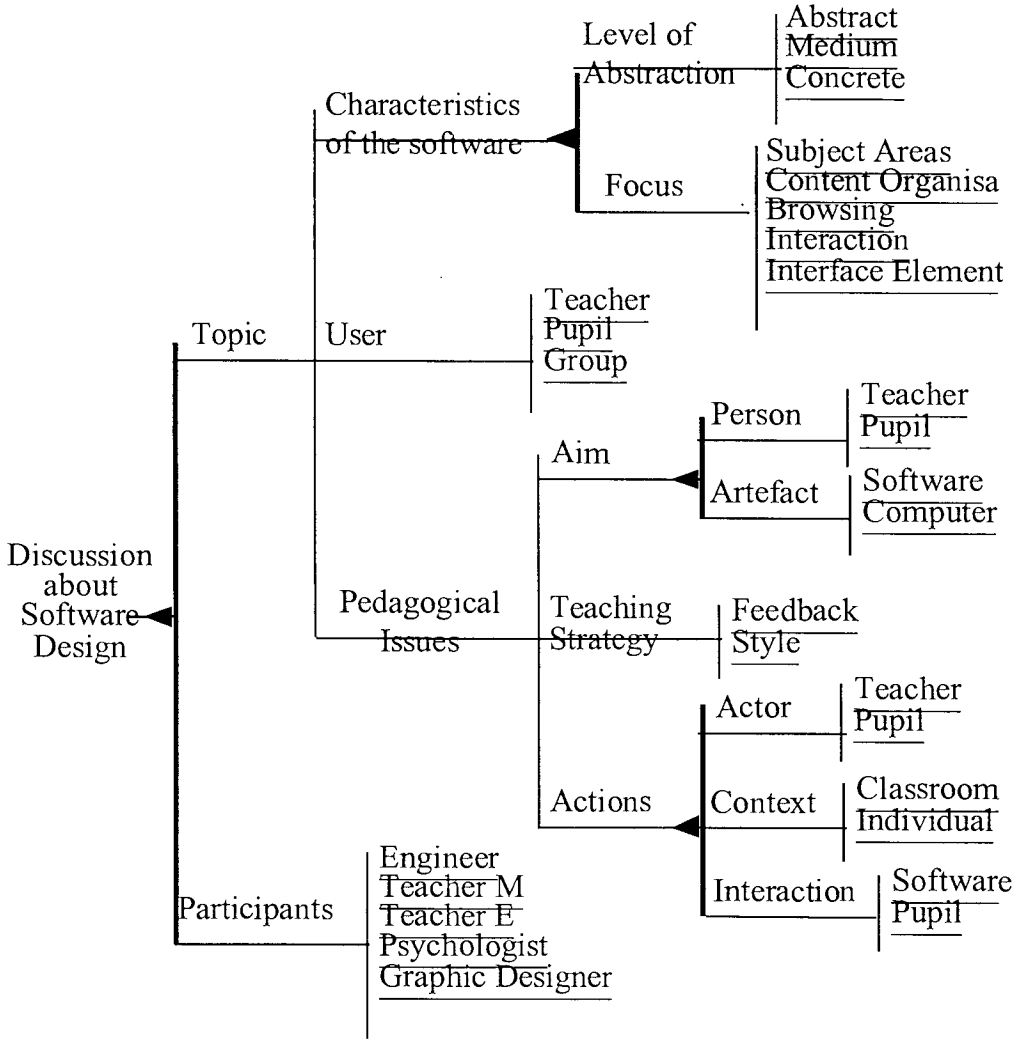

Figure 4.1. Final systemic network

4.3. OPERATIONAL DEFINITION

As it was presented in the Methodology Chapter, the final version of the network contains two general dimensions that characterising the unit of analysis[12]. These are: the 'Topic' discussed (to answer: what are they speaking about ?) and the 'Participants' in the discussion (to answer: who is taking part in the discussion ?).

The topic of discussion was divided in three main subjects related to the software design; the first one related to the dimensions of software engineering ('Subject Areas', 'Content Organisation', 'Browsing', 'Interaction' or 'Interface Element'), the second one was about the characteristics of the software's user ('Teacher', 'Pupil' or

[12] As a reminder, the unit of analysis was defined as a sequence of one or more utterances made by one or more members of the team in which they refer to one particular aspect of the software. In cases where they referred to two (or more) aspects simultaneously (in parallel), they were considered as two (or more) separate units.

'Group') and the third one was 'Pedagogic Issues' that were related to software and/or computers ('Aim', 'Teaching Strategy' or 'Actions').

The operational definition of each branch of the network is presented below. In order to present these definitions, those branches that involve an AND node are presented as double entry tables.

4.3.1 Characteristics of the software

This branch contains the data regarding the software engineering characteristics of the piece of software. These characteristics were classified into five different types ('Focus'):

- **Subject Areas**: Content that could be included in the software.
- **Content Organisation:** Static organisation of the contents of the software
- **Browsing**: Dynamic sequence of contents that will be displayed while the user is advancing through the different stages of the software.
- **Interaction**: Actions designed in the software for the user to interact with. That is, the human-computer dialogue.
- **Interface Element**: Objects and entities that are included in the human-computer interface of the software and its aesthetic properties.

Then, each characteristic was classified according to the level of abstraction in which they were described, these levels of abstraction were defined as:

- **Abstract** level, which corresponds to claims that were related to general principles or issues about software.
- **Medium** level, which corresponds to claims that referred to a piece of software and could be applied to all sections or elements of it.
- **Concrete** level, which corresponds to claims that referred to specific elements of a piece of software.

Table 4.1 presents a definition of each of the 15 resulting categories (3*5), giving examples taken from the data to illustrate them.

Level / Characteristic	Abstract	Medium	Concrete
Subject Areas	Curriculum subjects areas or skills (i.e. history, reading or basic skills).	Sections of contents to include in the software (i.e. which contents to include given the area).	Specific contents to include in each module or screen (i.e. screen one has content 'x').
Content Organisation	The principle that guided the contents' organisation (i.e. in terms of grades v/s achievements or cross curriculum v/s subject oriented).	How to organise the contents to be presented (i.e. to have a learning and evaluation section or a have a round story).	Decisions about the combination of content in each screen (i.e. which contents to combine in one screen).
Browsing	Type of sequence of contents to provide. (i.e. to progress based on difficulty levels or achieved behaviour.	Possible sequences of browsing and behaviours in special situations (i.e. given the organisation, which will be the browsing sequence).	Sequence of contents in a screen. (i.e. next content to present in one screen or which is the next screen to show).
Interaction	Interaction metaphor that was build in the software (i.e. users' fantasy while using the software).	Style of interaction with the software (i.e. type of software responses, sound, movies. Times a question is repeated).	Particular interactions in screens (i.e. specific actions that the user will do and software's responses).
Interface Element	Graphic style, choice of colours, type of casts (i.e. persons v/s animals, level of vocabulary).	Elements such as buttons, icons, casts that are present in all the software (i.e. the story teller character of the software).	Elements in a screen (i.e. backgrounds, scenarios and specific objects).

Table 4.1. Definition of the group of categories 'Characteristics of the software'

4.3.2 User

This branch was defined so as to group claims made about the characteristics of the intended users (i.e. the categories: teacher, pupil or groups) of the software. The data in these categories correspond to specific characteristics of each type of user, which could be the teacher, the pupil and in some cases the composition of the groups (when the software is meant to be used in groups). For example, the members of the development team could define the age of the users, they could describe their social background, their cognitive development stage, etc.

4.3.3 Pedagogic issues

The 'Pedagogic Issues' branch was defined so as to group expressions about issues that refer to beliefs about software or computers and projections of how these could be used afterwards. This branch was divided into three categories: 'Aim', 'Teaching Strategy' and 'Actions'.

4.3.3.1 Aim

This branch refers to the meaning that members of the development group gave to the software or computer, expressions about what it meant for the 'Teacher' and for the 'Pupil' were grouped under these categories. The definition of each category is:

	Software	Computer
Teacher	The software for the teacher, what does it stand for (i.e. a cross-curriculum- and cross-level resource, a lesson)	The computer for the teacher, what does it stand for (i.e. control instrument, professional tool)
Pupil's	Role of the software for the pupils (i.e. a game where they use their time in purposeful ways)	Role of computer for the pupils (i.e. as source of motivation, as rehearsal station)

Table 4.2. Definition of the group of categories 'Aim'

4.3.3.2 Teaching strategy

This branch included discussions about the software behaviour/characteristics that relate to the teaching conceptions of the teachers. For example, the decision as to what kind of feedback should the software give to the user (category 'Feedback'), or what should the software do when the user gives a wrong answer, or a long description of a particular character in the software that is supposed to guide the user in the different exercises (category 'Style').

4.3.3.3 Actions

This branch grouped expressions about different actions that both, teachers and pupils could be doing while using the software. In this sense, this group of categories reflects the roles that the development team envisioned for the people involved in using the software.

The resulting categories were the product of three branches that had two categories each, this is, eight different categories defined by three different dimensions: 'Actor', 'Context', 'Interaction'. The definition of each category is:

Actor/Interaction	Context	
	Classroom	Individual
Teacher/Software what the teacher does with the software	What the teacher does with the software while teaching in the classroom (i.e. projecting images, running simulations, etc.).	What the teacher does with the software (i.e. setting up levels of difficulty in the software, remotely controlling pupils' work).
Teacher/Pupil what the teacher is doing with the pupils	What the teacher does to manage the classroom (i.e.: asking questions, telling a story, motivating)	What the teacher is doing with the pupils (i.e. help, give advise, evaluate the content).
Pupil/Software what pupils do while interacting with the software.	The style of interaction and activities in the classroom including the computer (i.e. groups in front of the computer or individual based activities)	Type of interaction between pupil and software (i.e. receiving instructions, discovering, reflecting about something on the screen, using a tool)
Pupil/Pupil what pupils do when not interacting with the computer.	The type of activities that design for the classroom (i.e. collaborative activities, competitions among groups).	Type of interaction among the pupils (i.e. discussions around, through the computer, or peers tutoring)

Table 4.3. Definition of the group of categories 'Actions'

4.4. DISCUSSION OF THE CODING PROCESS

The operational definition of the coding categories presented in the previous section was used by the researcher during the coding process. It should be remembered that this was the result of the iterative process of inspecting the data, specifying some ill defined categories and solving some inconsistencies. This refinement process was aimed at improving the clarity, completeness and self-consistency of the network (Bliss, et al., 1983).

This section presents some of the problems found during the coding process and the way these were solved or dealt with. Despite these problems, the conventions defined during the coding were followed in order to ensure validity, (i.e. that it was appropriate in kind and, within that kind, sufficiently complete and faithful), and to ensure reliability in the sense that there existed an acceptable level of agreement between people[13] as to how use the network system to describe data (Cohen & Manion, 1994).

The problems found were:

- **Characteristics of the software**

  During the coding process the data that alluded to the characteristics of the software was rather easy to identify and to assign to some of the different categories of 'Focus'. The limits between consecutive groups of categories were sometimes difficult to differentiate, for example, between 'Content Organisation' and 'Browsing' or between 'Browsing' and 'Interaction', but similar data were classified under the same category in order to be consistent.

  Another problem with these categories was that there was no clear division between the different 'Levels of Abstraction'. This, because frameworks, such as the ones associated with cognition[14], were not used to define these limits. Rather, for each subject or focus, an arbitrary division was defined that enabled the researcher to separate claims about aspects that could appear more general or more specific in relation to the piece of software that was being designed.

- **User**

  As presented in its definition, this category assumed that data concerning the three types of users would be found, but data regarding the teacher or groups was

[13] In this case the supervisor and the researcher.

[14] For example the ones defined by Piaget to describe the different stages of cognitive development: pre-operational, operational, concrete, abstract.

almost absent. So, these categories mainly contained descriptions made by the development team concerning the pupils.

- **Pedagogic Issues: Aim**

  In this case the difference made between the computer and software was sometimes difficult to define. For the purposes of this analysis, claims about software or computers were differentiated by the type of reference the development team made about it. That is, if they spoke about a generic artefact that could be any software, it was classified as claims about computers. If they spoke about any specific software product or 'idea ware', it was classified as software.

  Also, it was difficult to separate their references to 'Teacher' or 'Pupil', because if they said that "the software is for rehearsal", they expressed their (pedagogical) goal not the pupil's aim when using the computer (pupils do not realise that they are 'rehearsing'). The criteria for defining the limit here was to assume that they spoke about pupils' aim when they referred to the effects or roles that the computer or software will produce on or play for the pupil. For example, if they referred to rehearsal, it was assumed to be an effect on the pupil.

  On the other hand, there were cross categories issues, like the control of the lesson flow and classroom management issues. These type of expressions were classified in both categories, 'Teacher' and 'Pupil'.

- **Pedagogic Issues: Teaching Strategy**

  These dialogues were separated because, in designing these particular elements, teachers spoke about other issues that dealt more with a certain behaviour of the software that was closer to the design of a strategy to teach rather than to a simple response of the software. These behaviours were then embedded in the software.

- **Pedagogic Issues: Actions**

  Some problems regarding these categories were, for example, that it was difficult to make a clear division between the activities of the teacher interacting as an individual with the software, or interacting with the software in front of a class in a classroom context (similar to the report by Fraser, et al., (1991), where the teacher 'orchestrates' the activities with the software). In this case it appeared that, because of the type of software designed, there where very few (if any) utterances that could be classified as an activity of the teacher with the software in a classroom context. There was one example where the teacher was supposed to

set up the level of the software before the children start using it, but this was considered to be an individual interaction rather than a classroom interaction.

Also, certain claims could be understood as either 'Aim' or 'Actions'. For example, the issue of the computer taking the role of the teacher is both: an aim of the computer for the teacher (for example, considering it as an auxiliary teacher), and a definition of a role of the teacher (for example, as the principal teacher).

4.5 DISCUSSION OF THE NETWORK

As mentioned before, the network can be considered as an accurate structure to represent the teachers' beliefs about educational software. In this sense, there are two dimensions to be analysed, first the categories defined and second, the organisation of these categories.

In relation to the first dimension it could be argued that the categories grouped in the branch 'Topic' show the different elements that were considered during the development process and, considering that the focus of this study was on the teachers' expressions, it could be argued that these categories represent the range of concerns that teachers expressed about a piece of educational software. Therefore it seems reasonable to assume that, for example, teachers selecting a piece of software will be thinking of a similar set of categories to inspect the software.

Accepting such assumptions, a valuable source for comparison of this structure can be found in the literature about educational software selection (or evaluation). In particular, the categories in the network could be compared with the different dimensions that are described as relevant to the evaluation and selection of educational software. In particular, these categories could be examined using the 'Perspectives Interactions Paradigm' defined by Squires and McDougall (1994) and described in section 2.4.2 of this thesis. The correspondence between the perspectives and the categories in the network could be described as:

- The teacher-student perspective could be related to the categories of the branch 'Actions' of the network. In fact, both, the network and this perspective, describe a similar range of possible actions.

- The student-designer perspective could be related to the categories of the branch 'Aim' when the 'Person' is the 'Pupil'. This perspective is focused on "identifying implicit theories of learning and decide whether (i) these are appropriate to perceived needs, and (ii) the software design is consistent with the

theory" (Squires and McDougall 1994, p. 99). The network, on the other hand, adds the aims associated to the use of the computer (alone), which is not explicitly considered in this perspective.

- The designer-teacher perspective could be related to the categories of the branch 'Aim' when the person is the 'Teacher'. This perspective is "concerned with identifying and judging the appropriateness to a given educational setting of the curriculum assumptions in the software design" (Squires and McDougall 1994, p. 110). The network also includes the consideration of the aims that the teacher can have regarding the use of the computer.

- The categories included in the branch 'Teaching Strategy' of the network are indirectly considered in the Teacher-Student perspective, but without a particular emphasis on the 'teaching strategy' dimension.

- The categories included in the branches 'User' and 'Characteristics of the software' are not included in this paradigm.

Through this comparison it was possible to ensure a higher degree of validity to the network (Cohen & Manion, 1994), in so far it was complete enough to contain the characteristics described in the 'perspectives interactions paradigm'. Also, it could be argued that the categories defined in the network provide additional perspectives that could be considered while selecting a piece of educational software. Squires & Preece (1996) propose the 'Jigsaw Model' for software evaluation, which is a result of integrating learning and usability issues for evaluating software. The model proposes the evaluation of three different tasks: the learning task, the operational task and the integrated task. In this sense, they are including the categories that are defined in the 'Characteristics of the software' branch of the network. Whether the inclusion of these additional categories would result in a better selection or not goes beyond the aims of the present study, but the fact that these authors tend to integrate these categories into their selection or evaluation methods opens interesting possibilities for the proposed network.

In relation to the second dimension - the organisation of these categories - the network constitutes an analytical tool for examining teachers' work with educational software. An alternative representation for all these categories would be to transform it into a check list of the relevant dimensions to be considered while observing teachers' work with educational software or while selecting a piece of software. In doing so, it would loose one important feature, i.e. the holistic representation of the topic of study. This feature enables the researcher to have a complete view of the

different alternatives of classification that are available and, further, to keep these alternatives in mind while reading the data (or reviewing a piece of software). In this sense, it could be argued that this representation could help to overcome, for example, some of the difficulties in using the 'perspectives interactions paradigm' reported by McDougall & Squires (1995), regarding the difficulty that assessors have in applying the method.

Another aspect that needs to be considered in this dimension are the choices made for the separation and organisation of the categories into a system of interconnected branches and nodes. This particular classification of the dimensions to be considered in the design of a piece of software conveys the organisational interactions that surround its use (an important consideration as Winograd, 1995, points out for 'environments for designing software'). In this sense, the network is expressing not only the characteristics of the piece of software but also the way and context in which it will be used by these particular users. Therefore it can be argued that it can represent these teachers' model of understanding educational software from the perspective of its use in the classroom.

V. PARTICIPATION ANALYSIS

5.1. INTRODUCTION

This chapter presents an analysis of the participation of each member of the development team in the discussions during the software development process. The participation is expressed in terms of the frequency of units spoken by each member of the development team in each category. The categories used for this analysis are those defined in the systemic network and explained in Chapter IV, these are: 'Characteristics of the software', 'Pedagogic Issues', (which has the sub categories: 'Aim', 'Teaching Strategy', 'Actions') and 'User'.

The overall distribution of units spoken during the design process is presented in the next figure:

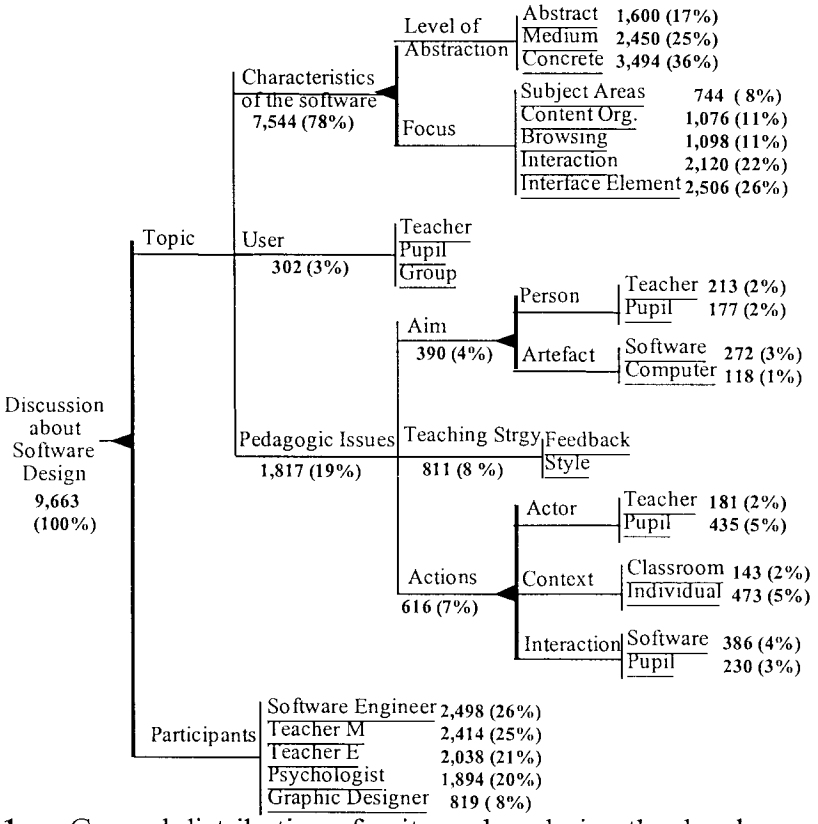

Figure 5.1. General distribution of units spoken during the development process

This distribution shows that the majority of the time invested (i.e. number of units spoken) was in designing the software (78%), particularly the 'Interaction' (22%) and 'Interface Element' (26%). The development team spoke half of the time about these issues but spoke very little about the 'Pedagogic Issues'. The participation of the Software Engineer (26%), the teachers (25% and 21%) and the Psychologist (20%) were similar, which suggests that they did interact during the development process.

The low participation of the Graphic Designer is because of her later entrance to the process.

Although this level of information is useful for such an analysis, this research is aimed at understanding what these teachers believed about educational software and therefore their participation during the development process should be analysed.

In order to do this analysis and as mentioned in Section 3.4.1, two calculations were done, first the individual frequencies of units spoken by each member of the development team in each category was calculated and second, these frequencies were transformed into percentages of the distributions relative to: (i) their participation in a group of categories ('Characteristics of the software', 'User', 'Aim', 'Teaching Strategy' and 'Actions'), and (ii) relative to each member's total units spoken. The aim corresponding to each calculation was: (i) to compare the contributions of each member of the development team to each group of categories (presented in section 5.2, 'Analysis of Contributions') and (ii) to compare the distribution of the units spoken by each member of the development team in all categories (presented in section 5.3, 'Analysis of the Profiles of Participation')

Finally, section 5.4 presents some of the conclusions drawn, based on the findings of these analyses.

During the design of this analysis, it was supposed that the contribution and participation profiles of the individual members of the development team during the design process should be different, in so as far they would contribute to this process from different knowledge domains. In other words, it was assumed that professionals would participate accordingly to their professional craft knowledge (Cooper & McIntyre, 1995; Edwards & Mercer, 1987), and that this knowledge would influence their design decisions (Hughes, et al., 1995; Mantovani, 1996) and consequently their frequency of participation in some of the categories would be different. The assumptions made, therefore, were:

- All members had the same opportunity to speak (except for the Graphic Designer that was incorporated to the development process in session 14), therefore the expected value for each MDT's contribution is 20% or 25% (if the Graphic Designer is excluded).
- The differences between the expected contribution and the observed contribution of each MDT can be attributed to professional 'deviations' (which are expected to be present). The assumption is that each of them has a professional background

that should induce him/her to be more or less active in certain topics, showing different frequencies of participation.

- The profile of participation of each member of the development team will depend on his/her professional background.

Given these assumptions, the hypotheses for this analysis were that[15]:

a) The two teachers would have contributed more to the categories that are closer to their professional background. In particular, they would have contributed comparatively more to the categories of the branch 'Pedagogic Issues' (i.e. 'Aim', 'Teaching Strategy', 'Actions') and to the groups of categories 'Subject Area' and 'Content Organisation' at an 'Abstract' level of abstraction. Consequently, they would have contributed comparatively less to the other categories.
b) The teachers' profiles of participation would be very similar compared to each other and it would be different compared to the profiles of the Software Engineer and the Graphic Designer.
c) The Psychologist would have contributed more to the categories that are closer to his professional background. In particular, he would have contributed more to the categories of the branch 'Users' and 'Aim'. Consequently, he would have contributed less to the other categories.
d) The profile of participation of the Psychologist would be close to the profile of participation of the teachers, because they share a certain amount of professional knowledge (that is, they both study learning and development theories). On the contrary, it would be different compared to the other member's profiles, in so far they do not study such theories.
e) The Software Engineer would have contributed more to the categories that are closer to his professional background. In particular, he would have contributed comparatively more to the categories of the branch 'Characteristics of the software', particularly to the groups of categories 'Browsing', 'Interaction' and 'Interface Element' at a 'Medium' and 'Concrete' levels of abstraction. Consequently, he would have contributed less to the other categories.
f) The Software Engineer would have a very different profile of participation compared to the other team members. This is because he studies computer science, software engineering theories, programming methods, etc. So, his professional knowledge is highly differentiated.

[15] Note that these assumptions are based on the curriculum of these professions in Chile and more specifically to the Universities in which the MDTs studied (Universidad de La Frontera, Universidad Católica de Temuco and Universidad de Temuco), and therefore they do not necessarily apply to other countries or Universities.

g) The contribution of the Graphic Designer can not be considered due to her late incorporation in the development process.
h) The Graphic Designer would have a very differentiated profile of participation compared to the other team members. The Graphic Designer has a very particular corpus of knowledge (painting, communication, etc.). Nevertheless, her late incorporation in the development process should be considered.

Both the analysis of contributions and the analysis of the profiles of participation will show that these hypotheses are supported only for some groups of categories and that it was not possible to generalise such behaviour to all the categories.

In the following sections the acronym 'MDT' will be used to replace the expression 'Member of the Development Team'.

5.2. ANALYSIS OF CONTRIBUTIONS

This section presents, for each group of categories, the distribution of frequencies of each MDT. These data provides the basis to analyse the quantity of participation of each MDT in each group of categories, and to contrast the observed behaviour with the expected participation of each MDT.

5.2.1 Characteristics of the Software

The distribution of frequencies for this group of categories is presented in table 5.1. It shows that in the distribution of the Group (upper left table), the highest frequency corresponds to a concrete level of conversation (46%), and the lowest corresponds to an abstract level (21%). In fact, frequencies in the abstract level are very low, except for the focus of conversation 'Interaction' (12%) which is over the half of the total frequency of this level (this can be explained by the fact that the category 'Interaction' at an 'Abstract' level is like telling the 'story' of the software and the user). The medium level of conversation of the Group is close to the expected value (32% v/s 30%).

Looking at the focus of conversation of the Group, it is possible to see that the highest frequency is for the focus 'Interface Element' (33%), closely followed by the focus 'Interaction' (28%). The lowest frequency is for the focus 'Subject Areas' (10%). This can be explained by the fact that the design of interface elements and interactions with the software are activities that people expect to design when developing a piece of software, they belong to the visible attributes of the software.

Comparing the contribution of each MDT to this group of categories, the low percentage of the Graphic Designer (11%) can be attributed to her late entrance to the development process. The relative high participation of Teacher M (25%) in this category, similar to the Software Engineer (27%), does not support hypothesis (a).

	The Group (7544)			
	Abstract	Medium	Concrete	Total
Subj. Areas	2%	6%	2%	10%
Content Org.	3%	7%	4%	14%
Browsing	2%	6%	7%	15%
Interaction	12%	7%	9%	28%
Interface El.	2%	6%	25%	33%
Total	21%	32%	46%	100%

	Graphic Designer (793)			
	Abstract	Medium	Concrete	Total
Subj. Areas	0%	0%	0%	0%
Content Org.	0%	0%	1%	1%
Browsing	0%	0%	0%	1%
Interaction	2%	0%	1%	3%
Interface El.	0%	1%	5%	6%
Total	2%	2%	7%	11%

	Software Engineer (2011)			
Subj. Areas	1%	1%	0%	2%
Content Org.	1%	2%	1%	3%
Browsing	0%	2%	3%	5%
Interaction	4%	2%	3%	9%
Interface El.	0%	1%	6%	8%
Total	6%	7%	13%	27%

	Psychologist (1425)			
Subj. Areas	1%	2%	0%	3%
Content Org.	1%	2%	1%	4%
Browsing	0%	2%	1%	3%
Interaction	2%	1%	2%	5%
Interface El.	0%	1%	4%	5%
Total	4%	7%	8%	19%

	Teacher E (1409)			
Subj. Areas	1%	2%	0%	3%
Content Org.	1%	2%	1%	3%
Browsing	1%	1%	1%	3%
Interaction	1%	2%	1%	4%
Interface El.	1%	1%	4%	6%
Total	3%	8%	8%	19%

	Teacher M (1906)			
Subj. Areas	1%	1%	0%	2%
Content Org.	1%	2%	1%	3%
Browsing	0%	1%	2%	3%
Interaction	3%	2%	2%	8%
Interface El.	1%	2%	6%	9%
Total	6%	8%	11%	25%

Table 5.1. Relative frequencies of units to the total units in the group of categories 'Characteristics of the software'

Looking at the relative distributions among the MDTs, and observing the levels of abstraction ('Abstract', 'Medium' and 'Concrete'), it is possible to say that:

- In the group of categories 'Concrete' level of abstraction, the highest percentage belongs to the Software Engineer (13%), followed by the Teacher M (11%). The high percentage of contribution of the Software Engineer supports the initial hypothesis (e), but the fact that Teacher M shows a similar percentage of contribution does not support the initial hypothesis (a), about the teachers having a relatively lower contribution to these categories.

- In the group of categories 'Medium' level of abstraction, the distribution is similar for all (7%), except for the Graphic Designer (2%). This does not support the hypothesis (a), (c) and (e).

- In the group of categories 'Abstract' level of abstraction, there are two groups, firstly the Software Engineer and Teacher M that have more participation (6%) and secondly the Psychologist, Teacher E together with the Graphic Designer that have less participation (3%). This result does not support hypotheses (a) and (e).

Looking at the relative contributions of each of the MDTs to each focus of conversation, it is possible to say that:

- There are no differences between MDTs in the groups of categories 'Subject Areas' and 'Content Organisation' (3%), except for the Graphic Designer, who has a lower participation (1%). This evidence does not support hypothesis (a), about the teachers having a relatively higher contribution to these categories.

- In the focus 'Browsing' the Software Engineer appears with the highest frequency in the conversations (5%). Although the difference is small, this result supports hypothesis (e).

- In the focus 'Interaction', the contribution of the Software Engineer (9%) matches the initial hypothesis (e), but the contribution of Teacher M (8%) does not support hypothesis (a).

- In the focus 'Interface Element' Teacher M (9%) and the Software Engineer (8%) have the highest percentage, followed by the Graphic Designer (6%). In this case, the contribution of Teacher M (9%) does not support hypothesis (a).

Adding the percentages of the teachers, they represent 44% (=19+25) of all units about 'Characteristics of the software', the Software Engineer represents 27% and the Psychologist represents only 19% of these units. So, the overall result does not support the hypotheses (a), (c) and (e).

On the contrary, looking at the general distributions it could be said that there are three groups with different degrees of participation, the first group including the Software Engineer and Teacher M (approximately 26%), the second group including the Psychologist and Teacher E (19%) and the third group including the Graphic Designer (11%).

The first group could be characterised by being more concrete and with an emphasis in contributing to the groups of categories 'Interface Element' and 'Interaction'. The second group could be characterised by being more uniform in their participation, contributing in a similar degree to all the categories in the branch 'Characteristics of

the software'. The third group is radically different and could correspond to a specialised role during the development process, focused on the group of categories 'Interface Element', but this result can not be considered in this study because the late incorporation of the Graphic Designer to the development process.

5.2.2 User

The distribution of frequency for this group of categories is presented in table 5.2.

The Group (302)				
	Teacher	Pupil	Group	Total
User	1%	86%	12%	100%

Graphic Designer (2)			
Teacher	Pupil	Group	Total
0%	1%	0%	1%

Software Engineer (45)				
User	0%	14%	1%	15%

Psychologist (75)			
0%	21%	3%	25%

Teacher E (88)				
User	0%	24%	5%	29%

Teacher M (92)			
1%	27%	3%	30%

Table 5.2. Relative frequencies of units to the total units in the group of categories 'User'

This distribution of frequencies of table 5.2 shows that during the design process very little was said about the characteristics of the teacher or the Group (of the branch 'User'). This fact may reflect the relative low importance that is given to the teacher as a user of the software or that the teacher's characteristics were assumed to be known by the team.

The highest percentages here are from Teacher M and Teacher E (approximately 29%), followed by the Psychologist (25%). The Software Engineer has a much lower participation in this category. The lower contribution of the Software Engineer supports hypothesis (e), but the relative lower contribution of the Psychologist does not support hypothesis (c).

5.2.3 Aim

The distribution of frequency for the group of categories in the branch 'Aim' is presented in table 5.3. Looking at the distribution of the contributions of the Group, the aims for the software and those expressed for the computer (of the branch 'Artefact') have different frequencies (70% and 30%, respectively). This difference can be attributed to the fact that the team was designing a piece of software, rather than a computer. The other interesting result is that they express more aims for the teacher (55%) than for the pupil (45%) (of the branch 'Person').

The Group (390)			
	Teacher	Pupil	Total
Software	36%	34%	70%
Computer	19%	12%	30%
Total	55%	45%	100%

Graphic Designer (0)		
Teacher	Pupil	Total
0%	0%	0%
0%	0%	0%
0%	0%	0%

Software Engineer (67)			
Software	9%	4%	14%
Computer	2%	2%	4%
Total	12%	6%	17%

Psychologist (81)		
6%	11%	17%
2%	2%	4%
8%	13%	21%

Teacher E (132)			
Software	11%	9%	21%
Computer	9%	4%	13%
Total	20%	14%	34%

Teacher M (110)		
9%	9%	19%
6%	4%	9%
15%	13%	28%

Table 5.3. Relative frequencies of units to the total units in the group of categories 'Aim'

In this group of categories Teacher E and Teacher M (34% and 28% respectively) have the highest relative frequency, followed by the Psychologist (21%) and the Software Engineer (17%). This distribution supports the initial hypotheses (a), (c) and (e) because this group of categories should reflect what they decide to be the aim of the educational software to be developed and in this sense it this is an area that is of more concern to the teachers and to the Psychologist in some degree. The absence of units from the Graphic Designer is due to her late incorporation to the development process.

Looking at the aims expressed for each of the users (categories 'Teacher' or 'Pupil'):

- The particular relative low frequency of the Software Engineer (6% v/s 13% of the other MDTs), when speaking about the aims for the pupil, is consistent with the initial hypothesis (e) and shows his low involvement in these decisions.

- Comparing the different frequencies about the aims for the teacher, the low percentage of the Psychologist in this area could indicate that he assumes the software as a tool for the pupil, rather than for the teacher. The high percentage of Teacher E in this category differentiates her from Teacher M, not supporting the initial hypothesis that teachers would show similar contributions, nevertheless, they still have the highest percentages of contributions.

Looking at the aims of the software or computer (categories 'Software' and 'Computer'):

- The Software Engineer and Psychologist share a relative low frequency when speaking about the aims of the computer (4%). It could show that they are focused on the software development and they do not give a role to the computer in this process, as opposed to the teachers who do speak about the aims of the computer (approximately 20%).

- The Software Engineer has the lowest percentage of contribution to the categories 'Software', particularly to the category 'Software-Pupil'. Again, this result is consistent with the initial hypothesis (e).

In general terms this group of categories was 'dominated' by teachers, which supports the initial hypothesis (a). It is interesting to note the fact that, compared to the other members of the development team, the teachers contributed most of the units of the categories 'Computer' (22% of 30%). This result could indicate that they consider the computer as a separate artefact from the software. One possible explanation could be that in a classroom, as opposed to the use of computers in an office, the mere presence of the computer irrespective of the software it is running is noticed by the pupils and will be motivated to use it. In this way they would be incorporating their professional background to the design process. Also, while expressing comparatively more aims for the teacher (35% of 55%), it could be said that they were considering the computer as a resource for the teacher.

5.2.4 Actions

To provide a clearer analysis, the group of categories 'Actions' is presented separating the ones designed for the teacher from the ones designed for the pupil. The distribution of contributions for the actions designed for the teacher is presented in table 5.4.

	The Group (181)			**Graphic Designer (1)**		
	Classroom	Individual	**Total**	Classroom	Individual	**Total**
Software	22%	55%	**77%**	0%	1%	**1%**
Pupil	4%	19%	**23%**	0%	0%	**0%**
Total	**26%**	**74%**	**100%**	**0%**	**1%**	**1%**
	Software Engineer (46)			**Psychologist (45)**		
Software	5%	16%	**21%**	5%	14%	**19%**
Pupil	1%	4%	**4%**	2%	4%	**6%**
Total	**6%**	**20%**	**25%**	**7%**	**18%**	**25%**
	Teacher E (51)			**Teacher M (38)**		
Software	6%	14%	**20%**	6%	10%	**17%**
Pupil	1%	7%	**8%**	1%	3%	**4%**
Total	**7%**	**22%**	**28%**	**7%**	**14%**	**21%**

Table 5.4. Relative frequencies of units to the total units in the group of categories 'Actions-Teacher'.

The distribution of contributions of the Group shows that they were very much oriented to defining actions of the teacher acting individually with the software (55%), compared with the definition of teacher's actions in the classroom with the pupil (4%).

In this group of categories, all the MDTS show a similar percentage of contribution. This result does not support the hypothesis (a), about the teachers showing comparatively more contributions. Also, the difference between the contribution of Teacher E (28%) and Teacher M (21%), does not support what was expected about teacher's similar frequencies of contributions. Consequently, it does not support the hypotheses about a lower percentage of contribution of the other MDTs ((c) and (e)).

The distribution of contributions for the actions designed for the pupil is presented in table 5.5.

	The Group (435)			Graphic Designer (3)		
	Classroom	Individual	**Total**	Classroom	Individual	**Total**
Software	3%	53%	**57%**	0%	1%	**1%**
Pupil	19%	25%	**43%**	0%	0%	**0%**
Total	**22%**	**78%**	**100%**	**0%**	**1%**	**1%**
	Software Engineer (106)			**Psychologist (59)**		
Software	1%	15%	**15%**	0%	6%	**7%**
Pupil	3%	6%	**9%**	3%	4%	**7%**
Total	**3%**	**21%**	**24%**	**3%**	**10%**	**14%**
	Teacher E (140)			**Teacher M (127)**		
Software	1%	14%	**16%**	1%	17%	**18%**
Pupil	7%	9%	**16%**	6%	5%	**11%**
Total	**9%**	**23%**	**32%**	**7%**	**22%**	**29%**

Table 5.5. Relative frequencies of units to the total units in the group of categories 'Actions-Pupil'

In table 5.5. the distribution of contributions of the Group shows that the team designed much more actions for the pupils individually with the software (53%) rather than actions for the classroom with the software (3%). This indicates that they were not designing software for classroom use, they design software for individual use.

In this group of categories the teachers show the highest percentage of contributions (32% and 29%), followed by the Software Engineer (24%). The Psychologist has a relatively low percentage of contributions (14%). Contrary to the previous results, this distribution of contributions supports the hypothesis (a), about the teachers showing comparatively more contributions.

Looking at the design of actions for the pupil in the classroom or individually (categories of the branch 'Context'):

- In the individual actions, the teachers and the Software Engineer have similar percentages of contribution (22%) and the Psychologist has the lowest one (10%). The comparatively high contribution of the teachers was expected (hypothesis (a)), but the one of the Software Engineer was not (hypothesis e), in so far as this category is closer to designing what pupils will be doing during the lesson.

- In the classroom actions, teachers have a higher frequency (approximately 8%) than the Software Engineer and the Psychologist (3%). This supports hypotheses (a), (d) and (e).

Looking at the design of actions for the pupil interacting with the software or other pupils (categories in the branch 'Interaction'), the highest percentage of contributions in the design of actions interacting with the software is the one of Teacher M (18%), especially while speaking about individual interaction (17%). Teacher E, on the other hand, shows the highest percentage of contributions in the design of actions with other pupils (16%).

Considering all the categories of the branch 'Actions', it is possible to say that the development team tended to speak more about actions of the pupil rather than of the teacher. Also, that they tended to speak much more about individual actions rather than actions in the classroom context and that their priority was the individual interaction of the pupil with the software.

This apparent behaviour could be explained by the traditional use of software in the classroom, which is very much oriented to consider the pupil as the end-user of the software (an argument supporting this view will be found in Chapter II: Literature Survey).

If this result is compared with the one for the group of categories 'Aim', there is an apparent contradiction, because there they talked more often about the aim of the software or computer for the teacher than for the pupil, whereas in this case they talked more about actions for the pupil than for the teacher. A possible interpretation of this is that they are thinking about the computer as a resource for the teacher that would be used by the pupils.

5.2.5 Teaching strategy

The distribution of contributions to this group of categories is presented in table 5.6. In this group of categories the Software Engineer, Teacher M and the Psychologist have similar percentages of contribution (27%) and Teacher M has the lowest percentage (17%). This distribution does not support hypothesis (a), about teachers showing a higher contribution in this group of categories.

	%
Group	100%
Graphic Designer	2%
Software Engineer	27%
Psychologist	26%
Teacher E	27%
Teacher M	17%

Table 5.6. Relative frequencies of units to the total units in the group of categories 'Teaching Strategy'

5.3. ANALYSIS OF THE PROFILES OF PARTICIPATION

This section presents an analysis of the distribution of the units spoken by each member of the development team (MDT) in all categories, that is, their profile of participation. These profiles constitute an additional source of information for comparing the participation of each member of the development team in the definition of each category.

In this analysis an assumption was made that the expected values of distribution for each participant are the ones from the group, so the analysis was focused on the difference in individual distributions compared with the group. Table 5.7 shows the distributions of frequencies of units relative to the total participation of each member of the development team.

Although the data presented in table 5.7 is interesting, it is difficult to make an analysis based on the numbers presented. Therefore, figure 5.2 shows a graphic representation of the profiles of participation.

Category / Participant	Teacher E	Teacher M	Psychologist	Software Engineer	Graphic Designer	Group
Characteristics of the software	**69%**	**79%**	**75%**	**81%**	**97%**	**78%**
Subject Areas - Abstract	2.0%	1.8%	2.0%	1.7%	0.0%	1.7%
Subject Areas - Medium	7.2%	4.4%	7.0%	3.1%	0.1%	4.8%
Subject Areas - Concrete	1.0%	1.1%	1.4%	1.3%	1.1%	1.2%
Content Organisation - Abstract	2.8%	2.1%	2.9%	1.8%	0.2%	2.2%
Content Organisation - Medium	6.8%	5.2%	7.4%	4.8%	1.8%	5.6%
Content Organisation - Concrete	3.0%	2.6%	3.7%	3.6%	4.6%	3.3%
Browsing - Abstract	2.0%	1.4%	1.4%	1.4%	0.1%	1.4%
Browsing - Medium	4.4%	3.6%	6.2%	5.0%	1.2%	4.4%
Browsing - Concrete	5.1%	5.3%	4.2%	7.6%	4.4%	5.5%
Interaction - Abstract	2.3%	10.8%	7.6%	12.7%	18.1%	9.5%
Interaction - Medium	6.5%	6.8%	4.4%	5.9%	1.3%	5.6%
Interaction - Concrete	4.6%	7.2%	6.2%	8.3%	8.9%	6.9%
Interface Element - Abstract	2.5%	2.4%	1.4%	1.0%	1.6%	1.8%
Interface Element - Medium	3.1%	5.7%	4.8%	3.7%	11.1%	4.9%
Interface Element - Concrete	15.8%	18.5%	14.6%	18.6%	42.1%	19.2%
User	**4%**	**4%**	**4%**	**2%**	**0%**	**3%**
Teacher	0.0%	0.1%	0.1%	0.0%	0.0%	0.0%
Pupil	0.7%	0.4%	0.5%	0.2%	0.0%	0.4%
Group	3.6%	3.4%	3.4%	1.6%	0.2%	2.7%
Aims	**6%**	**5%**	**4%**	**3%**	**0%**	**4%**
Teacher-Software	2.1%	1.5%	1.3%	1.5%	0.0%	1.4%
Teacher-Computer	1.7%	0.9%	0.4%	0.3%	0.0%	0.8%
Pupil-Software	1.8%	1.5%	2.2%	0.6%	0.0%	1.4%
Pupil-Computer	0.8%	0.6%	0.4%	0.2%	0.0%	0.5%
Action	**9%**	**7%**	**5%**	**6%**	**0%**	**6%**
Teacher - Classroom -Software	0.5%	0.5%	0.5%	0.4%	0.0%	0.4%
Teacher - Classroom - Pupil	0.1%	0.1%	0.2%	0.0%	0.0%	0.1%
Teacher - Individual - Software	1.3%	0.8%	1.3%	1.2%	0.1%	1.0%
Teacher - Individual - Pupil	0.6%	0.2%	0.4%	0.3%	0.0%	0.4%
Pupil - Classroom - Software	0.3%	0.2%	0.1%	0.1%	0.0%	0.2%
Pupil - Classroom - Pupil	1.6%	1.1%	0.6%	0.4%	0.0%	0.8%
Pupil - Individual - Software	3.1%	3.1%	1.4%	2.6%	0.4%	2.4%
Pupil - Individual - Pupil	1.9%	0.9%	1.0%	1.1%	0.0%	1.1%
Teaching Strategy	**11%**	**6%**	**11%**	**9%**	**2%**	**8%**
Feedback	10.4%	5.8%	11.0%	8.8%	2.4%	8.3%
Style	0.3%	0.1%	0.0%	0.1%	0.0%	0.1%
Total	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

Table 5.7. Relative frequencies of units to the total units of each member of the development team.

In order to have a clear representation of each member's contribution, a linear transformation was made, adding a factor to the frequencies of participation of each member of the development team (scale shift). This transformation allows us to see each member's profile in a different row of the graph. Also, because of the relative differences in the magnitudes of the participation in some categories, the frequencies corresponding to some categories were amplified. In summary, the frequencies shown

in table 5.7 were transformed using the following formula and The results of these calculations are presented in figure 5.2.

Profile of participation = (Original Frequency * Amplification) + Scale Shift

MDT	Scale Shift
Teacher M (TM)	0
Teacher E (TE)	2
Psychologist (PY)	4
Software Engineer	6
Graph Designer (GD)	8
Group	10

Group of Categories	Amplification
Charact. of the software	1
User	1
Aim	5
Actions	5
Teaching Strategy	1

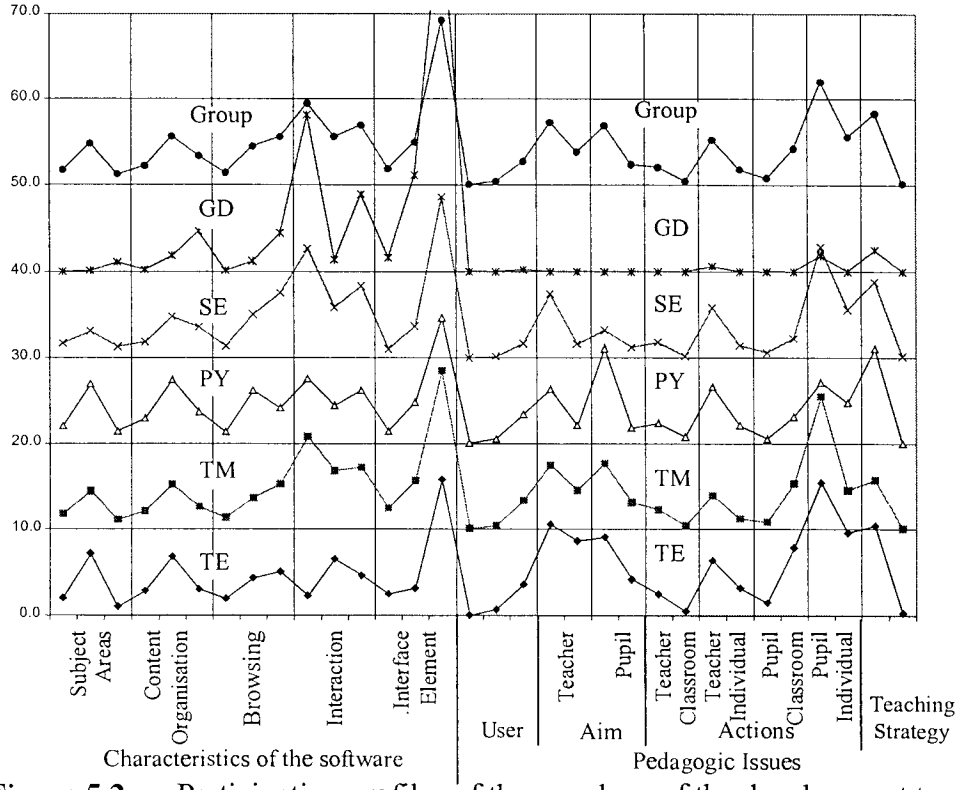

Figure 5.2. Participation profiles of the members of the development team

Figure 5.2 shows the profile of participation of each member of the development team. The Graphic Designer is not included in the detailed analysis of these profiles because, although she has a very different profile (which could support hypothesis (h)), this result can be attributed to her late incorporation to the development team, in so far as her participation opportunities were very different from the other members. Therefore, it can not be assumed that her profile of participation is a consequence of using her professional background.

Some observations arising from the comparison of each member's profile of participation with the Group's profile of participation, are:

- The Software Engineer has a relatively lower score in the group of categories 'Subject Areas' at a medium level of abstraction and a relatively higher score in the groups of categories 'Browsing' and 'Interaction'. Also, he shows a relatively lower score in the groups of categories 'Aim-Teacher' and 'Aim-Pupil'.

- The Psychologist has relatively higher score in the groups of categories 'Subject Areas' and 'Content Organisation' at a medium level of abstraction. His profile of participation is different in the group of categories 'Browsing', he has relatively lower score in his participation at a 'Concrete' level of abstraction. Also, he shows a relatively lower score and then a relatively higher score in the groups of categories 'Aim-Teacher' and 'Aim-Pupil' respectively. Lastly, he shows a relatively lower score in the group of categories 'Actions-Pupil-Individual' and a relatively higher score in the group of categories 'Teaching Strategy'.

- In general terms, Teacher M's profile of participation is very similar to the profile of the Group, the differences are in the group of categories 'Actions-Pupil-Individual' where she shows a relatively higher score and in the group of categories 'Teaching Strategy' where she shows a relatively lower score.

- Teacher E has a relatively higher score in the groups of categories 'Subject Areas' and 'Content Organisation' at a medium level of abstraction (similar to the Psychologist). Her profile is different in the group of categories 'Interaction', where she has a relatively lower score at an abstract and concrete levels of abstraction. Also, in the groups of categories 'Interface Element', she has a relatively lower score at a medium level of abstraction. Finally, in the groups of categories 'Aim-Teacher, 'Aim-Pupil', 'Actions-Pupil-Classroom', 'Actions-Pupil-Individual' and 'Teaching Strategy', she shows a relatively higher score.

Looking at the groups of categories presented figure 5.2, it can be observed that at least one member's profile of participation is different in the groups of categories: 'Subject Area' (SE, PY and TE), 'Content Organisation' (PY and TE), 'Browsing' (SE and PY), 'Interaction' (SE and TE), 'Interface Element' (TE), 'Aim-Teacher' (SE, PY and TE), 'Aim-Pupil' (SE, TE and PY), 'Actions-Pupil-Classroom' (TE), 'Actions-Pupil-Individual' (PY, TM and TE) and 'Teaching Strategy' (PY, TM and TE). It could be said that these are the categories where some of the members of the development team did apply their professional backgrounds.

Contrasting these results and the profiles shown in figure 5.2 with the initial hypothesis, the conclusions that can be derived are:

- That hypothesis (b), about the teachers having similar profiles compared to each other and different profiles compared to the Software Engineer, can be supported only for the groups of categories 'Browsing', 'Aim-Pupil', 'Actions-Pupil-Classroom' and 'Actions-Pupil-Individual'.
- That hypothesis (d), about the Psychologist having similar profile compared to the teachers and different to the other members, can be supported only for the groups of categories: 'Subject Area' and 'Content Organisation'.
- That hypothesis (f), about the Software Engineer having a different profile of participation, could be supported for only the groups of categories: 'Subject Area', 'Browsing', 'Interaction', 'Aim-Teacher', 'Aim-Pupil', and 'Actions-Pupil-Individual'.

5.4. CONCLUSIONS

This section will present a summary of the results of both analyses and then it will relate these results. The results of the analyses presented in the previous sections that support some of the hypotheses were:

i) The Software Engineer did contribute more than the other members of the development team in the groups of categories 'Browsing' and 'Interaction' and the analysis of the profiles of participation showed that his profile of participation was different in these groups of categories. Also he contributed less to the groups of categories 'Aim' and 'Actions-Pupil', and the analysis of the profiles of participation showed that his profile was different in these groups of categories.

ii) The teachers did contribute more than the other members of the development team in the groups of categories 'Aim' and 'Actions-Pupil' and the analysis of the profiles of participation showed that their profile of participation was different in these groups of categories.

The remaining hypotheses were not supported.

These results indicate that during the development process the Software Engineer and the teachers did use their professional backgrounds in those sections of their discussion classified in the groups of categories: 'Browsing', 'Interaction', 'Aim' and 'Actions-Pupil'. This result provides a higher degree of confidence about the data that is being used, in so far as it shows that at least in these categories the teachers were contributing from their professional stand point. We will turn in the next chapter to

examining what they actually said in their discussion which were classified in these categories.

Two additional observations resulting from the data were: first, if the groups of categories 'Aim' and 'Actions' are compared, it is interesting to notice that they did consider aims of the computer and software for the teacher, but they spoke very little about actions for the teacher with the software or the pupil. This configuration of participation suggests that they assume the computer and software as relevant classroom resources, but, as opposed to a blackboard, these resources are to be manipulated by the pupils and not by the teachers. This may be attributed to the fact that the machine is part of the activity in the classroom, so they need to consider it separately. It is interesting to compare the role of the computer in an office with the role of the computer in a classroom. In the former if the machine is off it is just furniture, in the latter, even if it is off, the teacher will need to consider its presence while designing activities. Teaching involves several roles (for descriptions see: Leinhardt, et al., 1987), and one of them is transmission of contents (or skills, thoughts, etc.), but teachers need to manage the classroom (Jones, 1996) also, and provide an adequate atmosphere (Woods & Jeffrey, 1996) and therefore the physical artefact (computer) matters. In this sense the very presence of the computer would have some effect on the activity performed in so far it affects pupils' attention and motivation.

Second, in the group of categories 'Actions' the whole group talked more about actions of the pupil than actions of the teacher, this difference could show that the group did not see a particular role for the teacher while using software or computers. Within the actions of the pupil they talk much more about actions with software interacting in an individual context than in a classroom context. This may be attributed to the existing trend in educational software development to produce learning centred software, which imposes a high degree of interaction between the pupil and the software.

VI. SEQUENCES ANALYSIS

6.1. INTRODUCTION

This chapter presents an analysis of the patterns of sequences of units that members of the development team repeated during the process of software design. For the purposes of this analysis a 'sequence' was defined as a consecutive set of 'n' different units that were present in a string of units more than once. The categories that identify the units of this analysis are the same as those defined in the systemic network and explained in Chapter IV. These categories are: 'Characteristics of the software' (which has the sub categories: 'Subject Areas', 'Content Organisation', 'Browsing', 'Interaction' and 'Interface Element'), 'Pedagogic Issues', (which has the sub categories: 'Aim', 'Teaching Strategy', 'Actions') and 'User'.

This information is useful in that it provides an additional level of analysis of the data, this is, the combination of certain categories form yet a different category that has its own meanings associated. In this sense, the aim of this analysis is to identify and analyse these new categories.

As explained in the Methodology chapter (section 3.4.2), the development process was considered to be a string of 848 units of analysis that represented one continuos string of data. The units were ordered, but not separated, accordingly to the chronological order of the consecutive sessions. In order to have a better profile of each member, their individual contributions during the development process were separated and one string of units for each member was obtained. Through the process of separating individual contributions the following number of units for each member (or 'string lengths') were found:

Member	Number of Units
Teacher E (TE)	2,038
Teacher M (TM)	2,414
Psychologist (PY)	1,894
Software Engineer (SE)	2,498
Graph Designer (GD)	819
Group	848

Table 6.1. Total units per member and the Group

Because of the difference in the number of units and her late entrance to the development team (session 14), the Graph Designer was not included in this analysis.

In order to have a better understanding of the findings, two types of analysis were carried out. The first, presented in section 6.2, was aimed at finding the relevant

sequences that appeared in the data, without making any discrimination for the person who was the 'author' of the sequence. This analysis showed that there were two types of sequences, one containing two different units of analysis (independently from the length of the sequence, i.e. the number of units of the sequence analysed) and another that contained three different units of analysis (sections 6.2.1 and 6.2.2 respectively). This analysis enables us to visualise the relation between units and thereby to have a new set of categories of analysis (constituted by the combination of two or three of the categories defined in the systemic network).

The second analysis, presented in section 6.3, was about the different degrees of participation of each member of the development team in the sequences found in the first analysis (the new set of categories of analysis). That is, given the most frequent sequences, this analysis allowed us to see who were the authors of these sequences and with what frequency. This analysis was aimed at understanding what units did each MDT relate during the design process and thereby to see what categories of the systemic network did each of them relate.

In order to have a quantitative parameter to define a degree of significance of the sequences found for each member, the probability of each sequence was calculated and compared to each MDT's probability for each sequence. This, because the number of units shown in table 6.1 allow the possibility that sequences were found by chance, that is, that a certain sequence was found simply because the units appeared in the data so many times that the probability of combining these units was higher than combing other units. Therefore the analysis is focused on those sequences that, having a low probability to happen, did came up in the data. The calculation of the probability for each unit in the data, for each sequence found and the expected probability for all units is presented in Appendix A.2.

As complementary information, Appendix A.3 shows the detail of each sequence, presenting the specific categories that composed each sequence. This detailed view is useful to provide an insight into the particular patterns of the data.

In order to present this analysis, the categories of the systemic network were grouped into their hierarchical branch (for example: 'Aim: Teacher-Software', is presented as 'Aim'). The reason for this simplification is to present the data in a more readable format (the original analysis was done using the detailed categories and is presented in Appendix A.3). The names of the groups of categories that are used in this chapter are presented in table 6.2

Category of the systemic network	Name given to the group of categories
haracteristics of the software	
ubject Areas at levels of abstraction: 'Abstract', 'Medium' or 'Concrete'	Subject Areas
ontent Organisation at levels of abstraction: 'Abstract', 'Medium' or Concrete'	Content Organisation
rowsing at levels of abstraction: 'Abstract', 'Medium' or 'Concrete'	Browsing
nteraction at levels of abstraction 'Abstract', 'Medium' or 'Concrete'	Interaction
nterface Element at levels of abstraction 'Abstract', 'Medium' or Concrete'	Interface Element
ser	
eacher, Pupil or Group	User
im	
eacher or Pupil, for the Software or Computer	Aim
ctions	
f the Teacher or Pupil, in the or individual, with the Software or Pupil	Actions
eaching Strategy	
eedback or Style	Teaching Strategy

Table 6.2. Names of the groups of categories

6.2. ANALYSIS PER CATEGORY OF SEQUENCE

At a general level these results showed that almost all of the sequences were combinations of two units. For example: 'Interaction-Actions-Interaction' or 'Interaction-Interaction-Actions' for 3-units-long sequences and 'Interaction-Actions-Interaction-Actions-Actions' or 'Actions-Interaction-Actions-Interaction-Actions' for 5-units-long sequences. That is, a combination of two different units (for example, 'Interaction' and 'Actions') that was repeated. Very few sequences composed of three different units were found, and almost no sequences composed of four different units.

The next sections present the analysis of sequences composed of two different units (section 6.2.1) and of three different units (section 6.2.2).

6.2.1 Sequences involving two units

The most frequent units found consecutively forming a sequence were:

- 'Interface Element' and 'Interaction'
- 'Browsing' and 'Interface Element'
- 'Interaction' and 'Actions'
- 'Browsing' and 'Teaching Strategy'
- 'Subject Areas' and 'Content Organisation'
- 'Browsing and 'Content Organisation'
- 'Content Organisation' and 'Interface Element'
- 'Content Organisation' and 'Interaction'
- 'Aim' and 'User'
- 'Actions' (combinations of actions of the pupil individually or in the classroom)

Underlined are those sequences that based on the probability analysis seem likely to be meaningful and not simply a result of high frequencies of units. The total number of sequences found for each length of string was:

Sequences / Length of string	3	4	5	6	7	8	9	**Total**	1/%*
Interface Element and Interaction	392	440	446	453	422	395	345	**2893**	**3**
Browsing and Interface Element	191	184	158	146	140	127	103	**1049**	**8**
Interaction and Actions	185	153	150	147	135	132	129	**1031**	**8**
Browsing and Teaching Strategy	227	211	171	140	105	87	57	**998**	**9**
Subject Areas and Content Organisation	221	174	134	105	67	50	29	**780**	**11**
Browsing and Content Organisation	162	131	91	81	73	59	42	**639**	**14**
Content Organisation and Interface Element	91	70	66	65	55	52	46	**445**	**19**
Content Organisation and Interaction	143	96	53	42	36	34	20	**424**	**20**
Aim and User	73	56	38	29	16	12	0	**224**	**39**
Actions	59	39	24	17	12	6	0	**157**	**55**
Total	**1744**	**1554**	**1331**	**1225**	**1061**	**954**	**771**	**8640**	

*The number was calculated dividing the total number of sequences (8,640) by the total number of sequences of the row. It represents the inverse of the percentage of each sequence (1/%).

Table 6.3. Total number of sequences involving two units.

Table 6.3 shows that the most frequent sequence involved the group of categories 'Interface Element' and 'Interaction'. These software characteristics appeared very frequently together and accordingly to the probabilities analysis this can be attributed to chance, in fact, these were the most frequently spoken units (they represent 50% of all units), as was shown in Chapter V 'Participation Analysis'. Because these units represent the most 'tangible' elements of the software that was designed, this sequence could show that they were designing the elements of the interface (digital images that appear in the screen) and the way in which these elements should be used (functionality of these images) at the same time.

The second most frequent sequence involved the groups of categories 'Browsing' and 'Interface Element'. Again, accordingly to the probabilities analysis this result can be attributed to chance. Nevertheless, it might indicate that they were designing the images of the interface (pictures) and the order in which these images would be visited (in this case browsing) simultaneously.

The third most frequent sequence involved the groups of categories 'Interaction' and 'Actions' (as it is shown in table A.3.1 of Appendix 3, the actions designed were of the pupil interacting individually with the software). In this case they were designing the actions that the pupil would be performing in front of the computer individually. It should be noticed that due to the specific categories found in these units, this sequence does not represent design activities related to the activity that the pupils would carry on interacting with other pupils in the classroom.

The fourth most frequent sequence involved the groups of categories 'Browsing' and 'Teaching Strategy'. Considering the definitions of the categories in these groups, it could be argued that this sequence represent design activities related to the progression of the user through the contents. This is, while browsing in the software the user would be actively searching for the next content and the software would provide him/her with the next content.

The fifth most frequent sequence involved the groups of categories 'Subject Areas' and 'Content Organisation'. The relation between these two units was not surprising in so far it could be argued that the development team first defined the subjects and then organised how they would fit together.

The sixth most frequent sequence involved the groups of categories 'Browsing' and 'Content Organisation'. In this case it could be argued that the development team was designing the way in which the contents should be organised and visited by the user. This sequence represents all the claims made about the way in which the contents should be structured in order to be accessible for the user.

The seventh most frequent sequence involved the groups of categories 'Content Organisation' and 'Interface Element'. Again, accordingly to the probabilities analysis this sequence can be attributed to chance. Nevertheless, it could be interpreted as evidence that they were designing the 'physical representation' of the contents, that is how to organise and present them.

The eight most frequent sequence involved the groups of categories 'Content Organisation' and 'Interactions'. Here it could be argued that they were designing the actual use of the contents, relating the interaction possibilities and the content structure. Nevertheless, accordingly to the probabilities analysis this sequence can be attributed to chance

The ninth most frequent sequence involved the groups of categories 'Aim' and 'User'. This relation may indicate that they consider the user while designing the aims of the software or computer. In fact, table A.3.6 (Appendix 3) shows that they mainly combine the aims of the software for the teacher with the definition of the pupil as user.

The last sequence found involved combinations of the group of categories 'Actions'. This sequence was included here as a separate category due to its diversity. Table A.3.7 (Appendix 3) shows that while speaking about actions, the development team combined actions of pupils interacting with other pupils. They did not combine

actions that would involve the software as the target of interaction as they did in the sequence involving the groups of categories 'Interaction' and 'Action'.

Looking at the sequences found at a more general level, it is possible to say that there were mostly of the branch 'Characteristics of the software' like: 'Subject Areas' and 'Content Organisation'; and 'Browsing' and 'Content Organisation'. These are sequences composed by groups of categories that have some degree of overlapping in their definitions. In this sense, the fact that they appeared to be correlated shows some internal consistency while coding. As mentioned, some other sequences like 'Interface Element' with 'Interaction', 'Browsing' with 'Interface Element' and 'Content Organisation' with 'Interface Element' could be explained by chance.

Sequences that relate groups of categories as 'Content Organisation' and 'Interaction', 'Interaction' and 'Actions', 'Aim' and 'User', 'Browsing' and 'Teaching Strategy' are less obvious and constitute a matter of further analysis. These relations will be analysed in the 'Discussion and Implications' chapter and it will be shown that they share common characteristics that constitute the reason why they appear together in this analysis.

Finally, it is interesting to note that they did not relate internal characteristics of the software (such as the groups of categories: 'Subject Area', 'Interface Element', 'Browsing', etc.) with the groups of categories 'Aim' or 'User'. They separated this more abstract design topics from the concrete ones (the actual piece of software).

As an attempt to visualise the relative closeness of the different units that are part of the sequences in a different way, a 'distance index' was calculated. This corresponds to the inverse of the percentage of each sequence (see * in table 6.3). This parameter is presented in table 6.4 (only meaningful sequences are included).

	Interaction	Browsing	Actions	Teaching Strategy	Subject Areas	Contents Organis.	Aim	User
Interaction	--------		8					
Browsing		--------		9		14		
Actions	8		-------					
Teaching Strategy		9		-------				
Subject Areas					-------	11		
Contents Organis.		14			11	-------		
Aim							------	39
User							39	------

Table 6.4. Distance matrix for pairs of units

The matrix presented in table 6.4, allows to appreciate the 'relatedness' of the different groups of categories. It is possible to see that the group of categories 'Content Organisation', appears to be related to the groups of categories 'Browsing' and 'Subject Area'. This matrix can also be represented in a web schemata, where the link between two nodes represents the existence of a relation (i.e. a sequence composed by these two groups of categories) and the distance between nodes shows the relative frequency of the relation (i.e. the more frequently found together the closer the nodes). For the matrix shown in table 6.4, the web is presented in figure 6.1.

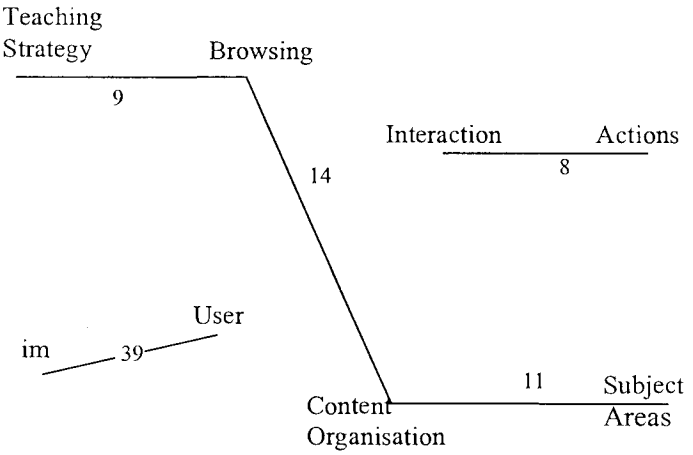

Figure 6.1. Web of relations between units

In this web the number of links starting from each group of categories ('Interaction', 'Actions', etc.) represents the frequency of appearance of this unit in the sequences. Looking at this, it can be appreciated that the most connected group of categories is 'Content Organisation', which has three links to other groups of categories. These links are rather long, meaning that the categories did not appear so frequently related to the other concepts. Nevertheless it shows the 'centrality' of the task of organising the contents during the development process.

Secondly, the groups of categories 'Browsing' and 'Interaction' show two links each. These links are rather short, particularly the ones between the groups of categories 'Interaction' with 'Action' and 'Browsing' with 'Teaching Strategy'. This represents the fact that these groups of categories were very frequently related to each other during the development process. Lastly, it is possible to appreciate that the groups of categories 'Actions', 'Subject Areas' and 'Teaching Strategy' have only one link, this is, they are end-nodes of the relation web.

6.2.2 Sequences involving three units

The most frequent units found combining three different units were:

- 'Interaction', 'Subject Areas' and 'Actions'
- 'Browsing, 'Subject Areas' and 'Content Organisation'
- 'Subject Areas', ' Interface Element' and 'Interaction'
- 'Interface Element', 'Subject Areas' and 'Content Organisation'
- 'Interaction', 'Aim' and 'Teaching Strategy'
- 'Content Organisation', 'Aim' and 'Actions'

Underlined are those sequences that were meaningfully based on the probabilities analysis (Appendix A.2). The total number of these sequences found for each length of strings is presented in table 6.5.

The first thing to be noticed is that the number of sequences found in this case is much lower than in the previous one (table 6.3, two units sequences). This fact shows something about the nature of the interaction during the development process. That is, the conversations during the development process were much more focused in designing two different elements of the software, rather than relating three or more different elements.

Sequences / Length of strings	3	4	5	6	7	8	9	Tot.	1/%
Interaction, Subject Areas and Actions	5	24	29	5	0	0	0	63	2
Browsing, Subject Areas and Organisation	8	4	4	4	4	4	4	32	4
Subject Areas, Interface Element and Interaction	23	0	0	0	0	0	0	23	6
Interface Element, Subject Areas and Organisation	8	0	0	0	0	0	0	8	18
Interaction, Aim and Teaching Strategy	8	0	0	0	0	0	0	8	18
Organisation, Aim and Actions	8	0	0	0	0	0	0	8	18
Total	**60**	**28**	**33**	**9**	**4**	**4**	**4**	**142**	

*The number was calculated dividing the total number of sequences (142) by the total number of sequences of the row. It represents the inverse of the percentage of each sequence (1/%)

Table 6.5. Total number of sequences involving three units

The most frequent sequence involves the groups of categories 'Interaction', 'Subject Areas', and 'Actions'. It is possible to argue that the development team was designing the way in which the user should work with the software, considering the subject areas as the framework for designing these activities. It could perhaps also be suggested that they were designing content driven activities for the user. If this sequence is related with the ones involving two units only, it could be argued that this sequence shows that they were defining the activities in the classroom (context) for the user based on the subject areas.

The second sequence involves the groups of categories 'Browsing', 'Subject Areas' and 'Content Organisation', but according to the probabilities analysis this sequence can be attributed to chance. Nevertheless, it could be argued that in these sequences they were organising the way in which the contents should be visited. It may reflect that they were designing how to structure the contents and how this structure should be accessed by the user, in this sense this sequence could be indicating a content oriented design tendency.

The third sequence involves the groups of categories 'Subject Areas', 'Interface Element' and 'Interaction' and as for the pervious one, accordingly to the probabilities analysis this sequence can be attributed to chance. Nevertheless, it could be interpreted as showing that in these sequences the development team was designing how to represent and interact with the software based on its contents.

The last three sequences appeared only eight times during the development process which is considered a rather low frequency, and therefore they will not be considered in this analysis. Although the last two sequences were found meaningful in the probabilities analysis due their low frequency it can not be argued that they would have any important influence in the design process.

So far, the only three unit sequence that could be considered to have some influence in the design process and that was found meaningful in the probabilities analysis was the sequence containing the groups of categories 'Interaction', 'Subject Areas' and 'Actions'. In order to visualise the 'distance' between these units the information can be presented using a double entry table, as shown in table 6.6.

	Interaction	Actions	Subject Areas
Interaction	-------	2	2
Actions	2	-------	2
Subject Areas	2	2	-------

Table 6.6. Distance matrix for pairs of units

As in table 6.4, the 'relatedness' of the different units can be seen. The possible web that would represent this matrix would be an triangle of side 2. This finding constitutes an extension of the relation between the groups of categories 'Interaction' and 'Actions' found in the two unit long sequences (Section 6.2.1).

6.3 ANALYSIS PER MEMBER OF THE DEVELOPMENT TEAM

The analysis of the contribution of each member only includes the sequences that were meaningful accordingly to the probabilities analysis (Appendix A.2).

6.3.1 Two unit sequences

Table 6.7 presents the number of sequences of each member of the development team in each sequence found.

Sequences of 2 different units	TE	TM	PY	SE	Group	Total
Interaction and Actions	262	433	0	334	2	**1029**
Browsing and Teaching Strategy	220	277	87	412	2	**996**
Subject Areas and Content Organisation	129	179	148	319	5	**775**
Browsing and Content Organisation	84	189	207	153	6	**633**
Aim and User	57	86	65	14	2	**222**
Actions	86	42	0	29	0	**157**
Total	**838**	**1206**	**507**	**1261**	**17**	**3812**

Table 6.7. Total number of sequences involving two units for each member of the development team

In table 6.7 it is possible to see that there was a significant difference between the number of sequences of Teacher M (TM) and the Software Engineer (SE) compared to the sequences of the Psychologist (PY) and Teacher E (TE). This is consistent with the findings of the previous Chapter 'Participation Analysis' in which they were shown to have had similar participation profiles during the development process.

We present two different analyses, the first looking at the rows and showing the relative participation of each member of the development team in each sequence, and the second looking at the columns indicating the frequencies of the sequences.

- **Row analysis**

Table 6.8 presents the percentages based on the rows. The sequence involving the groups of categories 'Interaction' and 'Actions' was used most often by Teacher M, followed by the Software Engineer and Teacher E.

As discussed previously, this sequence contains the design of the pupil's interaction with the computer and the fact that a teacher has the highest frequency here shows that this part of the software design was influenced by Teacher M. This combination of designing the interaction with the computer and the actions of the pupils defines what occurs in the classroom, and so it is not surprising that 67% of these sequences

were spoken by one or the other teacher. The lack of participation of the Psychologist here may indicate that he is not so much involved in defining what happens in the classroom.

Sequences involving 2 different units	TE	TM	PY	SE	Group	Total
Interaction and Actions	25%	42%	0%	32%	0%	100%
Browsing and Teaching Strategy	22%	28%	9%	41%	0%	100%
Subject Areas and Content Organisation	17%	23%	19%	41%	1%	100%
Browsing and Content Organisation	13%	30%	33%	24%	1%	100%
Aim and User	26%	39%	29%	6%	1%	100%
Actions	55%	27%	0%	18%	0%	100%
Total	**22%**	**32%**	**13%**	**33%**	**0%**	

Table 6.8. Relative frequency of participation of each member of the development team in each sequence involving two units.

The sequence involving the groups of categories 'Browsing' and 'Teaching Strategy' was used most often by the Software Engineer, followed by the teachers. The relative 'dominance' of the SE in the use of this sequence was not expected, in so far as the design of a certain teaching strategy would seem to be more related to teachers' expertise. However, the two teachers together do account for 50% of use of this sequence.

The sequence involving the groups of categories 'Subject Areas' and 'Content Organisation' was used most often by the Software Engineer, followed by Teacher M and the Psychologist. Here, it might be expected that the teachers might play a greater role in so far this sequence is very much related to 'curriculum' issues, but in fact, the teachers only account for 40% of these sequences.

The sequence involving the groups of categories 'Browsing 'and 'Contents Organisation' was used most often by the Psychologist, followed by Teacher M and the Software Engineer. This sequence may be viewed as representing claims about the structuring and accessibility of the contents and the fact that the Psychologist had the highest participation here could be an indication of his concern with the way in which the user would acquire these contents.

The sequence involving the groups of categories 'Aim' and 'User' was used most often by Teacher M, followed by the Psychologist and Teacher E, this result is consistent with the view that teachers would participate relatively more in the definitions of 'non-technical' issues of the software. The low participation of the Engineer in this sequence can be similarly interpreted.

The sequences involving the categories 'Actions' of the 'Pupil' 'Individually' or in the 'Classroom' were used most often by Teacher E followed by Teacher M, which may reflect a focus on defining what should pupils do during the lesson.

- **Column analysis**

The table of percentages based on the rows is shown in table 6.9. The Group shows its highest frequency of participation in the sequence involving the groups of categories 'Browsing' and 'Content Organisation' followed by the sequence involving the group of categories 'Subject Areas' and 'Content Organisation'. This could be taken as indicating that they were engaged in discussing these issues together as a group.

Sequences involving 2 different units	TE	TM	PY	SE	Group	Total
Interaction and Actions	31%	36%	0%	26%	12%	**27%**
Browsing and Teaching Strategy	26%	23%	17%	33%	12%	**26%**
Subject Areas and Content Organisation	15%	15%	29%	25%	29%	**20%**
Browsing and Content Organisation	10%	16%	41%	12%	35%	**17%**
Aim and User	7%	7%	13%	1%	12%	**6%**
Actions	10%	3%	0%	2%	0%	**4%**
Total	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

Table 6.9. Relative frequency of participation of each member of the development team in all sequences involving two units

Teacher E, as well as Teacher M, show their highest level of use in the sequence involving the groups of categories 'Interaction' and 'Actions'. Their priority then appears to be to define the activities that will be carried on in front of the computer. Both have their second highest participation in the sequence involving the groups of categories 'Browsing' and 'Teaching Strategy' which may reflect their concern with the design of the strategies associated to the pupil's interaction with the software (eventually the pedagogy embedded in the software).

Teacher E shows a higher level of use of the sequence involving the groups of categories 'Subject Area' and 'Content Organisation' and Teacher M has her third highest level of use for the sequence involving the groups of categories 'Browsing' and 'Content Organisation'. These two sequences are linked by the group of categories 'Content Organisation' and while the former relates it with the subjects to be taught the latter relates it with the navigation through these contents.

The Psychologist shows his highest level of use of the sequence involving the groups of categories 'Browsing' and 'Content Organisation', followed by the sequence involving the groups of categories 'Subject Areas' and 'Content Organisation'. Because these two sequences share the group of categories 'Content Organisation', it could be argued that he was designing the way specific content should be organised

and visited by the user. Also, he has zero participation in the sequences involving the groups of categories 'Interaction' and 'Actions' and the sequence involving group of categories 'Actions' alone, which may imply that he was not interested in designing the 'learning' activities of the pupils.

The Software Engineer shows his highest level of use of the sequence involving the groups of categories 'Browsing' and 'Teaching Strategy', followed by the groups of categories 'Interaction' and 'Actions' and also by the groups of categories 'Subject Areas' and 'Content Organisation'. This distribution may indicate that rather than showing any preferred domain of design he was simply responding to other member's propositions.

6.3.2 Three unit sequences

Table 6.10 below presents the number of three unit long sequence spoken by each member of the development team.

Sequences	TE	TM	PY	SE	Group	Total
Interaction, Subject Areas and Actions	0	26	0	37	0	63

Table 6.10. Total number of sequences involving three units for each member of the development team

In this table it is possible to see that the sequence involving the groups of categories 'Interaction', 'Subject Areas' and 'Actions' was spoken by Teacher M and the Software Engineer only. This result could show that these two members engaged in a discussion defining the activities that would be carried on with the software, in so far they relate those to the subjects to be taught.

6.4. CONCLUSIONS

Conclusions drawn from this analysis are mostly based on the two units sequences found. The reason is that the frequency of the three units long sequences is not enough for drawing conclusions. The conclusions are:

- Members of the development team combined units frequently. These sequences were mainly composed of two different units and only a few sequences were composed of three units. This result shows that the interaction between members of the development team was characterised by 'long' or repeated iterations considering two different dimensions of the software, rather than combining three or more dimensions in one discussion.

- The sequences found were mostly combinations of groups of categories of the branch 'Characteristics of the software'. There were two exceptions:

  i) The units involving the group of categories 'Actions', which were combined with the units involving the group of categories 'Interaction'. This result could be attributed to the operational definition of these units. Nevertheless, while analysing the specific categories that were spoken (table A.3.1 in appendix A.3), it would seem likely that the development team were designing the actions for the pupil interacting individually with the software, which implies that they were designing the software interaction thinking about the pupil as the only user. This supports the view that the development team thought about the software as a self-learning site (Olson, 1992) or as a rehearsal tool, and did not visualise the software as teaching centred (Hinostroza, Mellar, Rehbein, Hepp, & Preston, 1997).

  ii) The units involving the group of categories 'Teaching Strategy' that were combined with the group of categories 'Browsing'. This combination might be an indication that they considered that the activity of browsing with the software was closely related to some teaching roles. The rationale behind this assumption is that browsing is the activity by which users looks for new contents and while doing this they are guided by the computer. In the classroom context this would correspond to the exchange routines described by Leinhardt, et al., (1987), that is, the way teachers enter into dialogue with the pupils, the way teachers pose questions to the students or give feedback to them.

- The three unit long sequences found show that the development team designed the software combining the group of categories 'Subject Areas' with the groups of categories 'Actions' and 'Interactions' mainly of the pupils individually with the software. This could be interpreted as indicating that they designed the pupil's activities focusing on the contents and thinking about 'self-directed' work. Because this is a common practice of teachers' classroom activities design (Olson, 1992), it would not be surprising if they did indeed transfer this model of preparing a class to the design of this particular software.

- The group of categories 'Content Organisation' seemed to have a relation with the groups of categories 'Subject Areas' and 'Browsing', in so far it was found that this group of categories was frequently close to the other ones. This could show that the organisation of the contents to be taught has influence in several other

dimensions of a lesson design, such as the way in which teachers organise their teaching (Hammersley, (1990) describes similar activities in normal classroom teaching).

- The fact that very few sequences were found that combined groups of categories like 'Aims', 'User', 'Actions' (of the teacher), with groups of categories such as 'Subject Areas', 'Interface Element', etc. (the exception are some three units long sequences) may indicate that the development team did not relate these design elements with the specific characteristics of the software. They did not relate the learning aims and purposes of the technology with what should be done in the classroom with the software. In other words, their design was focused either on the pupil's learning a specific content while interacting with the computer.

There are several other types of analysis that could be done with these data, for example, the levels of abstractions in which they interact or the length of cycles of iteration for each sequence. But for the purpose of understanding teachers' concepts of educational software, they do not appear to be so relevant.

VII. CONTENTS ANALYSIS

7.1. INTRODUCTION

As discussed in the Methodology chapter (section 3.4.3) the analysis presented here was aimed at understanding the meanings of what the Members of the Development Team said during the software development process. Although all units were analysed, the focus was on the teachers' expressions about the different dimensions of the piece of software developed, rather than on the expressions of the other members of the development team. The need for such an analysis arises from the very purposes of this research, i.e. to understand teachers' conceptions of educational software. This constitutes the main analysis and which is supported by the previous analyses of chapters 5 and 6.

In order to facilitate the understanding of this chapter, the following section provides an overview of the contents (section 7.2.1), the development process (section 7.2.2) and of the piece of software developed (7.2.3). Then, sections 7.3 to 7.7 present the detailed contents of each category defined in the systemic network.

7.2. CONTEXTUAL INFORMATION

7.2.1 Summary of the contents

The number of units analysed in each category was presented in table 3.4. As a reminder, table 7.1 presents the same information, but grouping the categories into their hierarchical branches of the systemic network.

Groups of categories	Units
Characteristics of the Software	**583**
Subject Areas	71
Content Organisation	108
Browsing	104
Interaction	175
Interface Element	125
User	**57**
Pedagogic Issues	**208**
Aim	77
Actions	79
Teaching Strategy	52
Total	**848**

Table 7.1. Groups of categories and number of units spoken

This means that 71 units were classified as the group speaking about the group of categories 'Subject Areas' (including 'Subject Areas-Abstract', 'Subject Areas-Medium' and 'Subject Areas-Concrete'), and 108 about the group of categories 'Browsing' and so on. Each of these units could be a single assertion of one member of the development team or a several page dialogue where they defined some specific aspect of the software.

The general ideas that were discussed during the development process corresponding to each category are:

- **'Subject Area'**: After exploring several ideas about the subject area of the software at an abstract level of abstraction, they decided that it would be about 'basic skills' and at a medium level of abstraction they decided that this subject would be embedded into a playful story that children would 'read' (follow) while doing exercises. At a concrete level of abstraction the particular contents of each screen (exercises) were then adapted from a text book.

- **'Content Organisation'**: In this group of categories all members of the development team had different views on the way in which the contents should be organised in the software. At an abstract level of abstraction, the main difference was that the teachers organised it focusing on the different degrees of difficulty of the software's contents, rather than on the specific subjects or school grades as proposed by the Software Engineer and Psychologist. At a medium level of abstraction the teachers proposed organising it as a matrix, while the other members of the development team proposed organising it following a tree-like structure. The teachers' arguments for this organisation was that it should be based on pupils' progress (levels of difficulty). The Software Engineer and Psychologist proposed an organisation based on subject areas and grades (therefore a tree-like structure). At a concrete level they decided the number and order of the contents in each screen of the software.

- **'Browsing'**: The general browsing possibilities were designed at an abstract level of abstraction, the teachers proposed designing the progression strategy based on achieved behaviour (different levels of difficulty) and the Software Engineer and Psychologist proposed reviewing a specific subject at each time. At a medium level of abstraction they discussed the different paths that pupils would follow in the software. There was an interesting discussion between the teachers and the other members of the development team about the end of the software versus the end of the story. Teachers assumed that the end of the software was the end of the story (the 'software' metaphor). The Software Engineer and Psychologist, on the

other hand, used 'end' to refer to the end of using the software, which sometimes would be the end of the story but sometimes it would not (for example, if somebody 'quits' while working through the story). At a concrete level of abstraction they discussed about particular problems that could appear while browsing, for example, whether it would be possible to see the same screen twice or not. Also, they decided that the user should not know what would come next in the software.

- **'Interaction'**: At an abstract level of abstraction they designed the 'spirit' of the interaction with the software. They decided that the user should feel like the protagonists of an adventure and therefore (s)he should be able to directly manipulate the objects in the screen and have a dialogue with a special character in the software. They placed the user outside the computer, that is, they rejected of the idea of having the user inside the computer. At a medium level of abstraction they designed sequences of questions (exercises prompted by the software), answers by the user (actions) and feedback that the computer would give to the user. At a concrete level of abstraction they designed the particular responses of the software.

- **'Interface Element'**: At an abstract level of abstraction they designed the general shape of the interface elements (colours, type of figures, etc.). At a medium level of abstraction they designed the objects in each screen, such as backgrounds, characters and several elements that were part of the scenarios in which the story would happen. One particular element that they designed was the principal character of the story, who would guide and tell the user what to do. At a concrete level of abstraction they discussed the appearance of each element of each screen, taking care that each should look like a 'real' element and should be similar to the ones found in their neighbourhood.

- **'Actions'**: They designed the actions of the teacher so as to help the pupils to use the software, solving technical problems and to keep the discipline and management of the classroom. They rejected the idea of designing particular interactions of the teacher with the software arguing that children should be able to learn by themselves and that the teacher should guide them in this process of solving computer related problems. They argued that in the computer lab teachers would act as 'guides', whereas in the classroom they would teach the subjects. They did not consider scaffolding, tutoring or coaching to be part of their role in the computer lab. They described the pupils' actions as rehearsing while answering the questions prompted by the software, and they did not explicitly design any other type of actions. They believed that due to the playful interaction

designed in the software pupils would not realise that they were learning. They talked about group work and pupils interacting with each other but they did not incorporate these ideas into the software design.

- **'Aim'**: They expressed the idea that the aim of the software for the teacher was to save time, this is, due to its multimedia capacity the class could go through the contents much faster because the computer was used as a rehearsal tool. The aim of the computer for the teacher was defined as both, a teaching resource and a control tool due its motivational power. That is, a stimulus that they use as a resource that engages children in learning. They said that the aim of the software for the pupils was as a game, that while engaging in a playful interaction children would not realise that they were 'learning' or rehearsing. They said that the computer was a rehearsal tool that could serve as a complement to what is taught in the classroom.

- **'User'**: They described the user as a child of age 4 to 6 attending the early school levels. They described the children as socially deprived and with several behavioural problems. They did not describe the teacher as user of the software.

- **'Teaching Strategy'**: They spoke about teaching strategies while designing the feedback that the software should give to the user's wrong answers, and while designing the criteria by which the user would be able to jump to a higher level of difficulty, and also while considering the case of users misbehaviour in the classroom.

In order to give support to the results presented, there are several transcripts included that illustrate what is said about the different ideas and concepts that teachers have about educational software[16]. The protagonists of these transcripts are: 'TM' and 'TE' who are the two teachers involved in this study (i.e. Teacher M and Teacher E), 'SE' who is the Software Engineer, 'PY' who is the Psychologist and 'GD' who is the Graphic Designer who participated in the process only from session fourteen onwards (out of nineteen). Also in some dialogues some input from the researcher ('RE') is included.

Some ideas were elaborated through extensive dialogues, which sometimes lasted several sessions (not continuous), where they repeatedly addressed the same point, but they did not express the complete idea in one dialogue. This type of interaction

[16] In order to present the dialogues in a more readable format the gender of the pupils was translated as male. This does not mean that they referred always to male pupils, in Spanish some expressions do not distinguish between genders.

can only be understood by reading through several pages of disconnected dialogues. Although these type of dialogues are not included as transcripts, the conclusions of such processes are summarised.

7.2.2 Overview of the development process

As mentioned, the development process lasted 19 sessions and in each of these sessions, several different categories were discussed. Figure 7.1 presents a graph of the relative percentage of each group of categories spoken in each session.

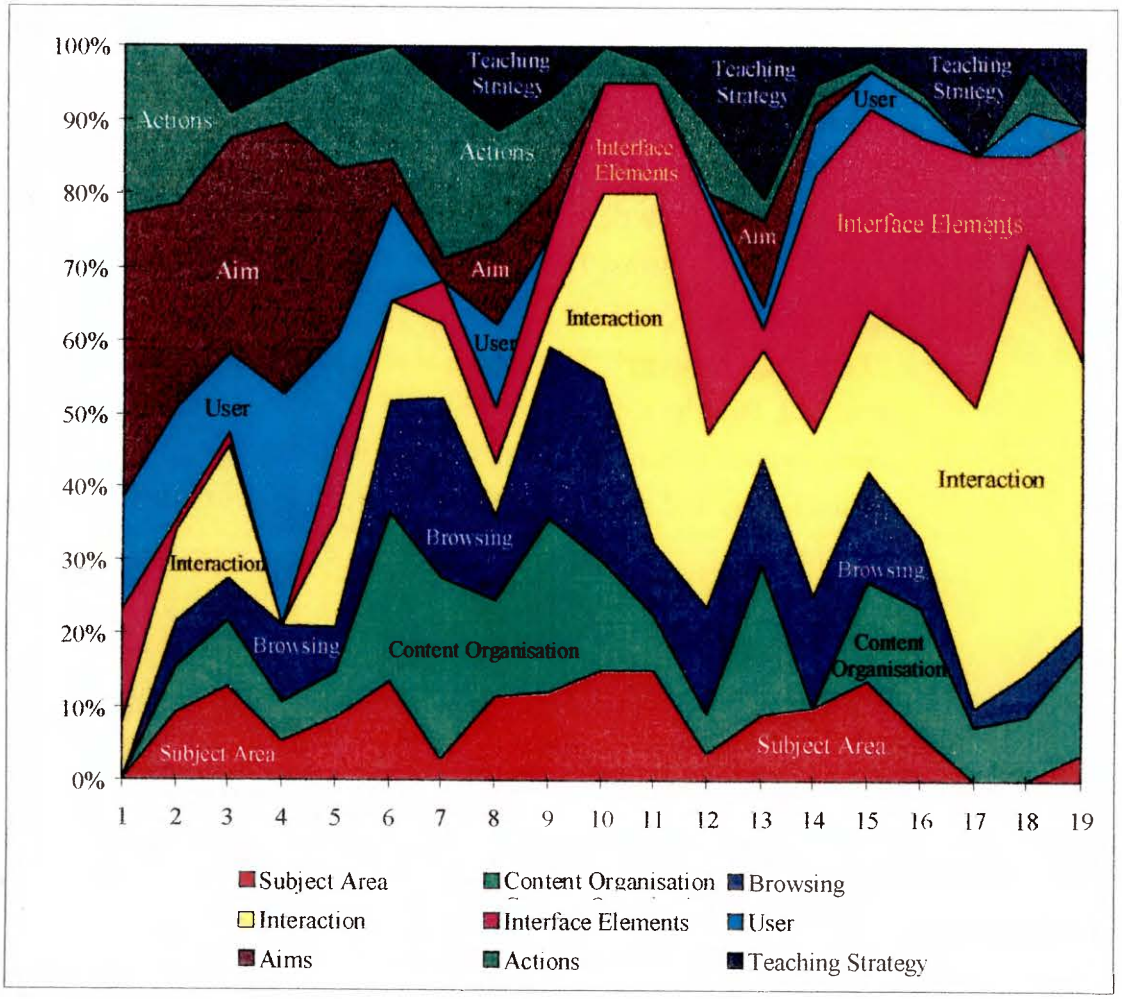

Figure 7.1. Categories spoken in each development session

Figure 7.1 shows that during the first five sessions the development team spoke mainly about aims and something about actions and the users (i.e. data in the groups of categories 'Aim', 'Actions' and 'User' respectively). Then, in sessions 6 to 10 the predominant topics were about the content's organisation, the browsing strategies and actions (i.e. data in the groups of categories 'Content Organisation', 'Browsing' and 'Actions' respectively). From session 10 to 19 the tendency was to speak more about the interaction with the software and the human-computer interface elements (i.e. data

in the groups of categories 'Interaction' and 'Interface Element' respectively). It can also be noticed that during the complete development process topics like subject area and teaching strategy were continuously spoken about (i.e. data in the groups of categories 'Subject Area' and 'Teaching Strategy' respectively).

In general terms, it could be argued that the distribution of topics across the sessions corresponds to a 'normal' pattern while developing software (for example, see the method described by Hinostroza, et al., 1996). They started by defining the general issues about the software, presenting examples and alternatives about possible products and discussing the possible subject areas to be included. Then, after defining the subject area and the target user, they focused on the definition of the structure of the software, organising the contents and, sketching the possible actions of the user and the browsing possibilities. After these characteristics were defined, they started to design the interface and particular functionality of the software. During the last 9 sessions, a prototype of the software was presented and analysed, and so, they focused on designing the human-computer interface elements and interactions of the user with the software. So, at a general, level the development process shown in figure 7.1 follows the stages described in some models of software development (for example the Revised Waterfall Model described in section 2.3.1). Nevertheless, there are a number of 'anomalies' that can be observed, for example:

- The development team started to discuss the browsing strategy of the software almost from session one (data in the group of categories 'Browsing'). Whereas it would have been expected that this characteristic would have been defined and agreed during some later stage of the development process. This could be an indication of the importance given to this characteristic of the software, in so far as it contains the way in which the user will navigate through the software, and in a sense, will follow the lesson.

- The discussion about the subject areas and its organisation occurs in almost all the sessions of the development process (data in the groups of categories 'Subject Area' and 'Content Organisation' respectively). It could have been expected that these topics of discussion would have been defined and agreed during some early stage, without requiring further discussion.

- The discussions about the human computer interface elements of the software and the interaction with them occurred mainly during the last sessions of the development process (data in the groups of categories 'Interface Elements' and 'Interaction' respectively). It could have been expected that, for example, the human computer interface elements would have been discussed earlier on.

This evidence illustrates the complexity associated with the software development process in so far as it shows the recurrence of some topics and thereby the eventual difficulty associated with defining them.

Finally, this way of presenting a software development process constitutes an interesting source for further research in so far it shows the range of topics discussed at each stage of the development process and could well be used to change and refine the development process, for example, by recommending a more focused discussion of the participants at each stage of the process.

7.2.3 Overview of the software developed

The software starts by presenting an animated introduction where the user is guided from the street into a house and ends in his/her bedroom. In this room the user is supposed to select one of three places to go: a farm, a neighbourhood and a school. Each of these choices is supposed to 'transport' the user into a particular story, a 'fantasy' world in which the user is asked to solve problems and answer questions.

The general shape of these scenarios is illustrated by some screen shots of the story in the scenario of the farm (figure 7.2). In the first screen (left to right) the user is asked to identify from which volcano is emitting smoke (click on the volcano) and then to click on the open window of the house. After successfully finishing these exercises the user is presented with another screen, that represents the next scenario in the farm (second screen). The type of exercises in the second screen vary in complexity. In this case the user is asked to click on the yellow flowers, then the red ones and so on. If the user does not correctly answer all the questions then the software presents a different scenario (still in the farm) that asks him to do exercises that are at a similar level of complexity as the ones in the first screen, but of different type.

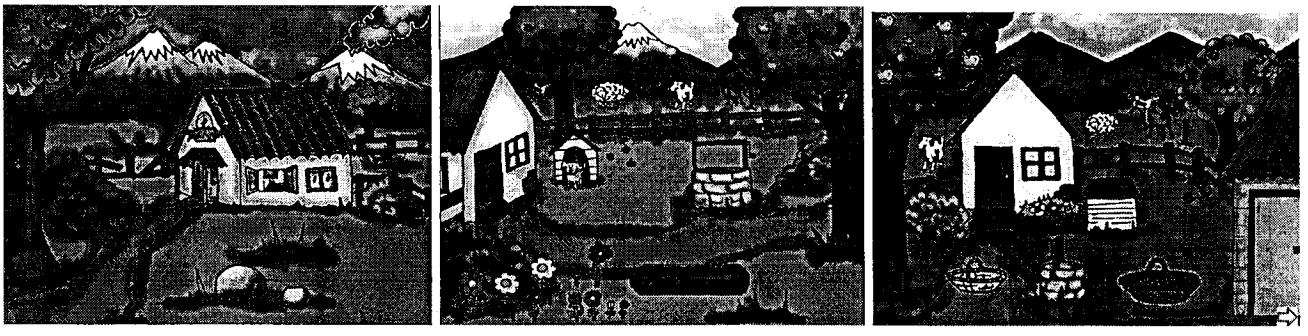

Figure 7.2. Example of three screens of the software

The third screen shows this option, where the user is asked to drag the apples from the tree into the basket. This last exercise is supposed to have a lower degree of difficulty, in that the concepts 'in' and 'out' should be known by the user (since they has already been rehearsed).

In the following sections, the ideas developed in each category of the systemic network are presented, including examples of the relevant dialogues.

7.3. CHARACTERISTICS OF THE SOFTWARE

This group of categories correspond to the data that were coded as descriptions of the software engineering components of the software. In order to present the data, this section is organised presenting each characteristic of the software at each level of abstraction defined.

7.3.1 Subject Areas - Abstract

At an abstract level, teachers defined the subject of the software to be about 'basic skills', that is, about a curriculum subject of the initial school levels. This choice was a result of five sessions of discussions, where they started to think about cross-curriculum contents for the software that could be useful for the first eight years of school, ending up with one curricular subject for the first two years of school. This section will show how these teachers decided on the subject area of the software, that is, the process of convergence and their final decision about the general subject area of the software.

- **Process of convergence**

The teachers started defining the subject avoiding references to the level of the pupils or to curriculum subjects, their goal was to develop a 'source of general information'. The reason for this, as they said, was that they had pupils that were already using sources of information in other grades. For example, in session 2 they said:

TE: That's why the idea is to do something with different subjects and groups was good. That is, in essence, that allows you to rehearse some topics like general culture. Not necessarily to enclose them into a subject based structure, but as more general topics. That is, suddenly, topics more for everyone also, that is, that they are useful from the third grade upwards, or neither so, so, <thinking>. Like when you are looking for some information and you need to browse the third's year book, or the fourth's one ore the fifth's one

TM: Yes, and this is happening [now in the school]

TE: And it is happening, you need everything [all the subjects]

TM: It is happening definitively. Say, today there was a work of the eighth level to investigate about flowers and animals of the ninth region

TE: And you need to look for [sources of information] to get around

TM: then, I said to them [to the pupils], take the third level book for natural sciences and onwards. And we started to browse the book and you know, almost everything was there.

In this dialogue teachers are saying that the topic of the software should not be restricted to a specific school grade, rather it should be useful for all grades ("from third grade upwards"). Apparently, this idea was a consequence of their experience ("it is happening") and they thought that the software could have provided this type of information source.

Later on, in Session 5, TM expressed what she now wanted to develop, which had changed from her original ideas. She said:

TM: Because the idea, I suppose, is that we stimulate basic skills with things. That is, that we have clear which are the basic skills [in general] and which are the basic skills that we want to develop; and, for example, which activities could be done to develop them, true ? Which could be different from the ones that the teachers are already using .

Compared to the previous dialogue, TM is now clear about what the subject of the software should be, that is, basic skills, which is part of the curricular aims of the initial levels of the school.

- **The definition of the subject area**

Finally, in Session 6, they started to define the contents of the software. TE gave examples about the type of basic skills that she would like to include, she said:

TE: I think that if one writes [a piece of software], for example, with really all the subjects of Kindergarten or [taught] in Kindergarten or not in Kindergarten, I mean, forget about Kinder, [lets say] first and second [grade]. If one writes a piece of software correctly for basic skills, there shouldn't be a lesson for it, because there is the writing [to be taught], you can do the initial sounds, the finals ones, you can work with words that start with A, show the letter. You can do a lot of things, and everything will be in the area of basic skills.

PY: Mh

TE: OK, then I say: why write [a piece of software], something apart for second and first grade, why not consolidate this and go deeper in it. That is, I am not sure if I made my self clear, I mean making like levels, that is to make levels or something.
<silence>

TM: And not get into the letters section [of the software]

TE: No, yes, but inside all, that is, inside where sensory motor co-ordination is. For example, [basic skills concerning] everything that is visual discrimination OK, auditory discrimination OK, where the computer says 'Ala' (wing) for example and it shows an 'A' and tells him 'A', there we have auditory discrimination, isn't it true ?

PY: Mh

TE: And also, visually the child is discriminating, and at the same time, for example, say [that] he is in grade 2 and he can use the keyboard, true? or in grade 1. He can already use the keyboard, and he could do more letters with 'A', I don't know, ..., that ... it contains all about vocals first, but inside the discrimination exercises, inside sensory motor co-ordination, inside the same [type of basic] skills. That is, that's what I mean.

In this dialogue TE was proposing a certain set of contents for the software that she automatically related to the school grades ("for Transition") but she tried to avoid the reference to the school level ("forget about ..."). Also, she was trying to express the need for having 'deeper' contents in the software ("that is to make levels or something") and to relate the subject with specific contents, like learning to read as the same time as developing basic skills. In this sense, she was looking for a piece of software that integrates different learning aims, but restricted to a specific school grade.

In a different sense, this shows that she had an 'ideaware' in mind (as described by Olson, 1988) and that she was trying to explain it to the other members of the group. The way in which she did this was by giving examples of the type of exercises that could be developed and in doing this, she was already designing the model of the software's interaction (exercising) and the user's actions (responding to the computer's prompting).

7.3.2 Subject Areas - Medium

At this level of abstraction, they designed the contents of the particular piece of software and they separated the curriculum related contents from the 'contents' of the software itself. That is, a playful story that they used as a metaphor to present (or deliver) the curriculum related contents.

- **The metaphor**

They designed a story as a metaphor, that would provide the framework where the user was supposed to exercise basic skills, interacting with elements of the different scenes. The specific dialogues where they developed this idea were in Session 5:

TM: Hey, and what if we invent a story, and the child develops basic skills through the narrative of the story ?

TE: Too much summarised, go on, I do not understand.

RE: How ?

TM: For example, the little Red Riding Hood, and there she goes [to visit her grandmother], because almost all children know little Red Riding Hood, then he [the user] has to put apples into the basket [of little Red Riding Hood], and she goes walking to her grand mother's house, and she goes into the house.

In this dialogue TM proposed the core idea of the software, which was that the pupils would be developing exercises through a story. Then she gave an example to clarify the concept: the story was about "little red riding hood going to visit her grandmother" and the exercise was: "then he has to put apples into the basket". She assumed that in doing this, the pupil would be rehearsing the concepts 'in' and 'out'.

The idea just presented by TM was not accepted immediately and in Session 6 they continued arguing about it, but this time the Software Engineer took the control of the argument, he said:

SE: Sure, if we choose a story

PY: Mh

SE: This idea, did you like it or not ? That of doing all these rehearsals through a mask with a story for example, that is something [known by the pupils]

TE: That all is camouflaged, but

SE: Clear, that it is, that it should be something more motivating than telling the child: "take this Apple and put it into the basket", for example.

TE: And after that he does something like

SE: Doing [Presenting] him a story

TE: Sure, because this [,the direct instructions,] would be like, doing like separate things, and I think that it would be like

SE: Too classic

TE: Like too classic, too

SE: Classic

TE: Scholastic, in this way too scholastic, or too, [for example:] here do a labyrinth, there do that, that is

SE: Sure

TE: It would be rehearsing exactly the skills but it would be too [instructional], OK [the story is] more motivating.

SE: It is like making him start.

TE: To bring him inside a story or inside of [a fantasy]

At the beginning of the dialogue the Software Engineer presented the idea to the teachers and then he justified the use of a story using a counter example (to give direct instructions like "put this thing there") and arguing that a story would be more motivating for the pupils. TE agreed and said that in the other way (direct instructions), it would be similar to typical classroom activities, where the teacher gives such instructions. This image of the teacher giving instructions shows some of the stereotypes she had about 'traditional' teaching . This can be related to Olson (1988)'s arguments about the computer serving as a Trojan horse to innovate. Then, the reason why this teacher is willing to use the computer is because she is rejecting the image of 'traditional' teaching.

After these discussions the idea of using a story to 'hide' the contents was accepted and further developed. These type of decisions, as to create a story that engages the pupil in a playful interaction with the computer, are described by Woods & Jeffrey, (1996) as a common strategy used by primary teachers as a way to 'bring home' to the child the contents of the curriculum.

7.3.3 Subject Areas - Concrete

At this level they defined the specific basic skills that they would include in the software. They defined the skills based on a book and in the meetings they simply stated what could be and what could not be included. As a result the number of dialogues coded in this level of abstraction were very few.

Because of the subject chosen, the role of the Psychologist at this level was much more clear, he was the one that defined the specific skills to be exercised through the software and designed the specific exercises to be included.

It is interesting to realise that in order to define the specific curriculum-related contents of the software they relied on a textbook and the Psychologist. It could imply that these teachers were aware of the strategies to teach (create the story and interactions) and the content to teach was given by some one else (Official

Curriculum) and the learning strategies by yet another person (textbook and Psychologist). This fact can be related to ideas about the craft knowledge of teachers, which is oriented to technical and affective aims and to the goal of learning a given set of contents, but not on defining those contents nor the learning strategies to be implemented (Marton, 1994), in so far as they receive a curriculum already designed by someone else, i.e. the Ministry of Education[17].

7.3.4 Content Organisation - Abstract

At this level of abstraction, the teachers shared one view, which differed from that of the Software Engineer and the Psychologist, about the basic criteria to organise the subject areas. On the one hand, the Software Engineer and Psychologist tried to define the different areas of the software based on the contents: the different curriculum areas (Language, Maths, Science, etc.) or different areas of cross curricular subjects (development of sensory motor co-ordination, visual perception, etc.). On the other hand, the teachers tried to organise the information based on a more 'dynamic' basis. That is, they organised the information considering its level of difficulty and they showed an integrated view of the curriculum subjects (which was consistent with their talk about the subject area of the software).

This section will present these different views, for example, in Session 2 the teachers talked about the organisation of the contents based on different levels of difficulty, which should not be based on the grades nor age of the users. They said:

TE: Now, it could also be done, I don't know. But this is to fall into [a curriculum-like organisation], yes. If it is atypical, to fall into this, is like doing, like levels, as levels are made of degrees of difficulty, something like that ?

SE: OK, yes

TM: Not of grades, not to say of levels, [like] from this age to the other. Rather of, you already played here, and now you can go to another level". It doesn't matter if you [the user] are in third or fourth grade. That's it ?

TE: Yes, it could be so, or it could, I don't know, be with degrees of difficulty some times, I don't know. Also, based on the scope of the subject.

TE started with the idea of organising the software based on levels of difficulty and TM complemented this idea by saying that it should not be based on school grades nor on the age of the pupils, rather, the organisation should enable the pupil to 'jump'

[17] At least this was the situation in Chile at the time this research was carried out. In 1997 a new strategy was developed by the Ministry of Education where schools will be able to design their own curriculum, given a common framework of aims and contents that should be included.

to different levels of the software without restrictions imposed by the grade, with TE adding only by the degrees of difficulty.

This way of organising the contents incorporated the user in its design, rather than looking at the contents as a set of subjects that needed to be structured and presented through the software, which was the way the Software Engineer and Psychologist viewed the organisation.

The data suggests that the teachers' ideas were not understood by the Software Engineer and the Psychologist and they continued designing the organisation of the contents without realising their differences. In the next dialogue an example of this situation is presented. In order to understand the dialogue this misunderstanding must be kept in mind and also the fact that they seemed not to be aware of this. In order to clarify the dialogue comments are included separating the main ideas. This dialogue occurred in the same session as the previous one (i.e. Session 2):

TM: Then it would be complete. If the child can be alone in front of the software, and if he also could have, should have. Look, that way, the chance, alone, [or] in teams, [to browse] there downwards, [going through a] first phase, second, third, and there downwards <gesture with the hand>.

PY: That is, the software starts immediately divided in two paths, in group and individual.

TM: OK.

PY: and then, it is divided into subjects, in subjects and these same subjects into levels [grades].

SE: no, look, the topic should be shared there, based on the subject, we should see the way to avoid repeating all the things [the subjects].

First, TM described an interaction with the software in which users would be playing and advancing through phases of the software, referring to levels of difficulty like in a video game. Then, the Psychologist expressing his agreement, said that the software would be divided depending on the type of use (individually or in groups), implying a separated set of contents in each case. After the agreement of TM, he continued explaining the structure, now dividing it into subjects and each subject in levels. Until here, it was possible to imagine the tree structure proposed by the Psychologist, where divisions were made depending on choices like subject or grades. At the end the Software Engineer argued that the subject should be shared, addressing the need to repeat the contents in different areas (sections) of the software.

TM: Subject in levels [of difficulty], and the other things also are the levels and in each level a subject. It could be.

SE: In each level [grade] one subject.

TE: No, look, one subject.

SE: Or the same subjects for each level [grade].

TM: Only one subject and that's it.
<noise>

SE: No, not in one level [grade], three subjects for example. But at the same time the three subjects are deeper treated.

PY: For example, here the basic [subjects], here are Natural Science, Social Science, to give an example. These are the most typical, Spanish, Maths and some arts, true?

TE: No, but not here.
<laugh>

TM: Don't curricularise me, don't curricularise me.

PY: It is because of the area, I don't know how to define the areas [of the curriculum], you can define these better.

TM: Areas(?) of expression.

After this initial explanation given by the Psychologist, TM argued that the subjects should be divided into levels, but also that in each level, there should be a subject. Here she was talking about the degrees of difficulty and not about the levels of the school (grades). Then the confusion persisted, on the one hand trying to organise subjects in levels of the school (grades) and on the other hand trying to say that the different levels (of difficulty) would contain different subjects. Afterwards, the Psychologist tried to explain it using the idea of curriculum subjects, but TM refused to identify the subjects on this basis and she changed to 'areas' that correspond to cross curriculum contents, which, again, seemed not to be understood by the Psychologist.

In these two examples, it would appear that teachers showed a more 'developmental' view of the subject, in that while designing the organisation, they imagined the user playing with the software, so they incorporated this dynamic dimension in the design. The Software Engineer and the Psychologist on the other hand, had a more structural view of it, they referred to subjects and levels for each subject, dividing the contents into hierarchical categories.

Later on, in Session 6, TE expressed what she really wanted. She referred something her colleague had said earlier and confirmed that they were talking about the same organisation, which was to have a story where the pupils did not realise that there were subjects, they would just be following a story. The dialogue was:

<first TE is saying that she and TM have similar approaches>

TE: You see, we can say perhaps completely different sentences. That is, she can say blue and I red, but we know that, you see, that it is a matter of minimal approach. That it is about details only, which would be, for example, I don't agree, I don't know if you understood what I said to her. I don't agree in doing specific contents for 2º grade for 1º. OK. I want only one story, that is what we want with TM. One story where you go through, and there it isn't, there is a curriculum crossing, that you don't note that this is [the subject] history. That, that's why the idea of SE is good, of doing a story, I like it, of doing a story perhaps a piece of software that is done right in a story where you can find, or I don't know, a story but where you can find the stages.

TE also said that they wanted to organise the information in a more integrated (cross curricular) way and that they wanted the pupil to 'go through' the story and not have separate 'chunks' of contents.

This 'breakdown' between the teachers, the Software Engineer and the Psychologist throws light on the different ways of approaching the organisation of the contents of the software. The teachers' way of organising the contents was closer to the 'traditional' way of organising lessons described by Hammersley (1990), that is to say: "pitched at a certain level of 'difficulty' according to the co-ordinate position of the class in relation to age and ability".

They wished to organise the contents so that the pupils could look for information within different degrees of difficulty. The fact that they insisted in having degrees of difficulty rather than sections of the subjects can be explained through this type of lesson design.

7.3.5 Content Organisation - Medium

The discussion at this level was about the way in which the different contents should be organised in the software, that is how to structure the 'sequence' of the content in the software. There were two interesting issues here, the first was related to the way in which teachers organised the specific content of the software based on the progress of the pupil while using the software. In this case, evidence was found of the different teaching strategies used by the teachers. The second issue was about the two different abstract organisations of the contents that came out from the design team.

- **Sequence of contents**

Related to different content organisation and keeping in mind the different points of view on how to organise the contents (subject areas versus degree of difficulty), the following dialogue presents the next level of detail. The dialogue occurred in Session 7, after defining the general idea of the way in which contents should be organised. In the dialogue, TM exemplifies the way in which the user would follow the story and thereby she designs the content organisation:

TM: Then I, my idea was that a story that starts with something, and we said that it was very simple, and the child goes through <gesture expression 'without problem'> OK?. And there he does little things. Then, [he] goes on to the next page of the story, which has different things, but [that are] related to the previous one, and he relates concepts that are there, that were there [in the previous page]. He relates them, he has to remember that he saw them, for example, OK. And [then,] we go to a third [page] and we keep developing basic skills in this, that are rehearsed with these [exercises] and here I present two or three new [concepts] that are rehearsed with [these exercises], reinforcing those from there [previous page], let's say five, to say something, OK?. And he finishes the story <while talking she was drawing the levels on a piece of paper >

PY: OK, the difficulty level would be ascending [while reading the story]

TM: Ascending, yes. But at the same time, reinforcing the previous [concepts] and remembering things, for example, here there should be elements from there.

In this case TM proposed an organisation based on ascending degrees of difficulty, but inside this general schemata she proposed to repeat the contents in different 'pages' of the software. That is, she was repeating part of the content in the next page with the aim that the pupil would remember the previous contents and relate them to the present ones.

Later on in the same session, TE expressed her view about the way in which the contents should be organised. She described basically the same type of organisation, but she changed some of the contents of each illustration, not including the previous contents:

TE: It is for example, like TM says, I don't know if you thought the same, but I, for example, TM says that a story that has like 6 or 7 illustrations [could be used], OK?, and that each of these stories has like, like. What she apparently tried to say was that it should [be designed for] one specific skill, the illustration [story]. Then, here, for example, we can go deepening this [skill], you see, here with an increased level of difficulty, the same as there [in the next illustration]. What was shown in the initial illustration [should

be followed], and here [in the next illustration] deeper [more difficult], because you can graduate the same exercises in terms of difficulty. I don't know if you follow me SE. And, for example, here, she said, here, we do another thing [exercise], a little more difficult, also in this way, [similar as] that in the same time that the contents [were presented]. Let's say that here it is discrimination or conservation OK? we are talking about conservation, a very simple conservation here OK? For example, the body parts that the child puts in order [of correspondence], disarranged and then he should arrange them. There you have a conservation at the level of age 4 or 5. Afterwards, a conservation, could be, the one about water, and then another one more complex, then, but that here at the initial story, and that this could also be <she interrupts her explanation>. That's why I said that at the end we could build like a story but well structured, that is what I said, like a strong story.

Here, TE is arguing that each story would develop a different basic skill and that the illustrations of each story should be organised based on increasing levels of difficulty at each time. She used examples to illustrate her point. The difference of TE's description from that of TM was in the reinforcing aspect. TM stressed the fact that the pupil should be reinforcing the previous knowledge and TE said that the content should evolve in terms of difficulty, and she did not include the reinforcing aspect.

This difference might appear to be rather insignificant, but it could reflect the teaching strategy used by each teacher. TM is repeating the cycle of contents, question and feedback described in the literature (for example by Hammersley, 1990), whereas TE did not include the previous content in the next level of difficulty.

- **General organisation of contents**

The second issue here was related to the abstract organisation of the contents in the software.

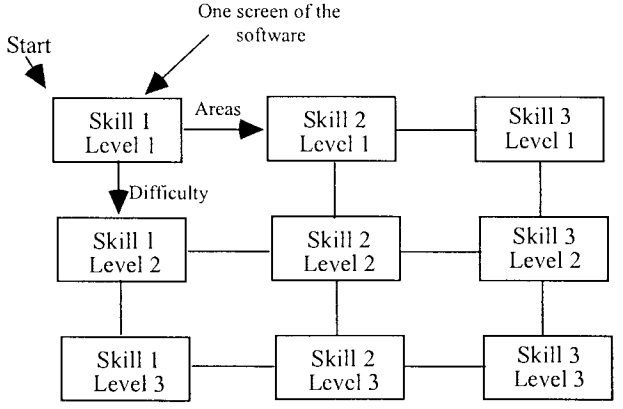

Figure 7.3. Teachers' abstract organisation of the contents in the software.

Figure 7.3 represents how the teachers proposed that the content should be organised. This is, they proposed that the content organisation should be like a matrix. The idea was that the user could browse through the software following a story that is composed by several illustrations (screens), each illustration would present exercises about some particular skills at a particular level.

The specific dialogue where teachers proposed this structure was in Session 7, where they said:

TE: If we do a story, for example. We, as we understand each other better <she refers to the other teacher>, if we build a story, for example, with ten pages, in one we do visual discrimination, in another language

TM: <interrupts> OK, perfect, good

TE: Then, but the story is the story, and downwards [,in a column of the matrix,] language is being graduated, the discrimination is being graduated, it [the level of difficulty] is being graduated

TM: OK, yes

TE: True ?

TM: Yes

PY: Suppose that in the second scene [,or column of the matrix,] we will see norms,

TE: To call it somehow

TM: Yes

PY: To call it somehow, in the third [column of the matrix] language, on the fourth whatever it is

TM: OK.

TE is proposing developing a story that had increasing levels of difficulty 'downwards' (referring to the vertical axis of the matrix) and where the scenes of the software are the different areas of the content of the subject (referring to the horizontal axis of the matrix).

On the other hand, figure 7.4 presents how the Software Engineer and Psychologist thought that the content should be organised. In this case the contents were organised accordingly to the different school grades, then each grade was divided into the way the software would be used (individual or group based activities) and finally the different subjects and areas of each subject correspond to the leaves of the tree.

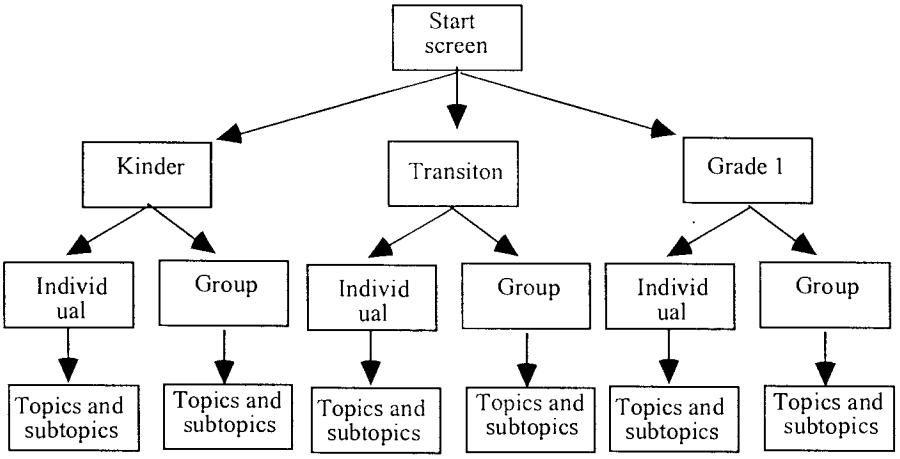

Figure 7.4. Software Engineer's and Psychologist's abstract organisation of the contents in the software.

The following dialogue from Session 7, shows the discussion of both ideas, and the decision to follow the teachers' proposition. In the dialogue, the Psychologist was arguing that with their proposition the level of difficulty would be well organised, because the pupils would be using the area of the software designed for them and so they would be able to finish the story. TE argued that with their organisation pupils would also finish the story because the same row of the matrix would be simple enough to finish. The dialogue was :

PY: Then, that was the idea. And then so. Because, what happens with this, is that it wouldn't have a deepening along the story, because if that were the case, if it had a deepening along the story, it could happen that the child of Kindergarten, supposing, would not be able to finish the story

TE: Yes

PY: On the other hand, if it is how we explained it, the level of requirement would be a little less for the kids, then they could finish the story anyway

TE: OK, but anyway with ours he would also finish it [following the first row only], because in the first illustration, OK, he would have simple things

TM: In our idea

SE: We are, yes we are exactly the same

TE: But, perhaps what you say is simpler. What we want is that, if the idea is present, yes, what we want is that it [our idea] is present, that there is, for example, here [a simple path to follow], so the kid could finish the story

PY: Anyway, that is, there wouldn't be a level of deepening along

TM: No,

SE: No, no, no, it is downwards.

PY: It would be always downwards.

TE: Yes.

SE: Ours is also downwards, but <he interrupts himself>

PY: No, because in the beginning you said that there would be an advancing.

TE: Eh, eh, in the story she said, but not in what will be taught as strategic in the story.

PY: That is, it wouldn't be more difficult across [the row].

TM: It will be changing

TE: Changing

As mentioned before, this difference in the way of organising the contents of the software reveals some of these teachers' teaching strategies, the matrix based structure enables the pupil to 'negotiate' with the computer at which level to interact, in this sense it was closer to a style of teaching defined as 'Interactive' by Cooper & McIntyre, (1995). In the other tree-like structure, the computer offers the pupils a specific subject and level to be learned and in this sense it was closer to a style of teaching defined as 'Reactive' by Cooper & McIntyre (1995) in which the pupil decides where to go, without mediation of the 'teacher' (computer in this case).

Also, the argument in which they based the discussion about the organisation of the contents reveals some of the teacher's craft knowledge, that is, they were concerned about the affective outcome of the use of the software, "pupils should finish the story" in order to avoid their eventual frustrations and to encourage their sense of achievement (as Woods & Jeffrey 1996, describe that primary teachers do during their normal lessons).

7.3.6 Content Organisation - Concrete

At this level they discussed the way in which basic skills should be presented in each screen of the software. They started organising the software so that in each screen they included one basic skill to be developed, but then they realised that it was not possible. The dialogue where they realised this occurred in Session 9, where TE expressed her concern about being able to have only one basic skill per illustration. She argued that normally the skills are evaluated in groups that belong to the same area and not individually. The Psychologist seconded this idea arguing that one isolated skill would not be enough to give the feeling of the story's context, therefore it should be integrated in the story. The dialogue was:

TE: That is, at the beginning I saw this as, as very simple, as SE said. That is very logical, here sensory motor co-ordination, here <she interrupts herself>. But then, after thinking about a little, [I found that] here there is something that I couldn't understand you the other day <telling it to PY>, that it is impossible to put things this way, that is, [one skill per illustration]. That's why, that's why, I think that perhaps, I have the intuition, I am not

sure if it is so, I am inferring that perhaps, that's why they evaluate the kids like in areas, because it is very difficult, it is too fine the separation [of basic skills] they do, you see?

TM: That's why, teachers work integrating [all the skills], they do a 'salad', but they evaluate [separating them].

TE: <interrupts> That's why, that's why we can't [separate them].

TM: To know where it is wrong and exercise it, build that

TE: That is, we would make a mistake if we develop [each skill] separately, you see? And in some place, in fact the story is already integral, because the story will have to include dialogue, it [will] have to include [different] elements.

PY: Sure, because that, exactly, that's the other point, we are working with, in relation to, a story. That is, we are doing an activity which is relatively playful, in which we need to integrate [different skills].

TE: <interrupts> We would need to produce the integration

PY: On the other hand, it is highly probable that, perhaps, we could reinforce a specific area based on a particular illustration, but we would have to stick to it, and forget about the context of the rest of the story.

The argument of TE in this dialogue was based on what teachers normally do in the classroom, that is her colleagues' experience and she did not draw on theoretical considerations. This suggests that TE was using her experience and practical knowledge to contribute to the design rather than focusing on learning theories as means-ends relations (as Marton 1994, reports that teachers do).

Later on, in Session 10, they discussed this issue again, but this time they gave different arguments to integrate many skills in one illustration. In these dialogues TE highlighted the fact that the software would be different from text books or a multimedia copy of them, because it would be integrated, and they (teachers) would be able to choose which basic skills to work with and the pupils would be reinforcing several skills in parallel. She complemented this argument by saying that this way of presenting the exercise would be closer to 'real life'. The dialogue was:

TE: I think that the difference, for example, of all texts available, with what we will do, the nice thing about our work, is that it wont be simple electronic illustrations with voice, sound and colour. Ours will be something integrated where we will be able in each basic skill [to] choose some categories with which we are interested to work, and with which you can go into a software, because not in all of them you can, as TM and PY said. Then, that's the nice thing about our software, because you will always be able to integrate, perhaps in one illustration, you will be able to see two or three skills. What we were talking about during the last session, for example, while giving one instruction

where we force the child to think, true ? That is, he will be using thinking skills, he will be reinforcing the language, it will be happening, that is, the difference is that we would be reinforcing something else.

TM: More complete.

TE: Sure, more of a total like kids' real life is really. That is, not illustration after illustration, which is what we do in Kindergarten or in grade one, where children develop much manual movement, true? and they fail, and it is hard for them to see the complete scenario, because they start, exactly, working illustration after illustration, then, I believe, that that's the nice thing and the innovative part that this work would have.

In the dialogue TE rejected what they normally did in the classroom, arguing that because of the types of exercises pupils did, they failed to gain a broad view of the subject. In this sense, she argues, the software would integrate different skills and therefore would be closer to real life (out of school) experience.

Given the decision that they would include several basic skills in one illustration, the next decision they faced was how many skills would be included in each illustration and what kind or level of skills. They decided to include three to four per illustration and that they should have an increasing level of difficulty, but they would be related to each other, they would correspond to the same type of basic skill. The reason for this selection as they said in Session 13, was: "it is not reasonable to teach to add, multiply, subtract and divide in the same class, you need several sessions to do it. The software ought to be the same".

Finally, they discussed how to give the instructions to the user. The choices were, to give all instructions at once, that is, indicating the required exercises for all the skills to be rehearsed in the screen, or to give instructions one at the time, before each exercise. The choice made was to give separate instructions to the user, because each of the exercises contained in one illustration would be about a different basic skill, so it would not be appropriate to give them all at once.

With these final decisions they had already defined several dimensions of the content organisation of the software, that is, the contents (basic skills), the software's metaphor (set of illustrations linked by a story), the content of each illustration (or screen) of the software and finally the dialogues with the user (in this case these were instructions). If these elements were compared with the design of a book, there would be only one difference, which is, the possibility of reading the story following different paths depending on the user's performance. In this sense, the computer would be reacting to the user's requirements and 'negotiating' the level of difficulty with them.

7.3.7 Browsing - Abstract

In this category the previous discussion about the way in which to organise the contents was also present and they discussed the way in which the pupil should progress through the software. To present this discussion there are only a few more dialogues included, because the evidence presented in the previous section illustrated this category as well.

As described above, the teachers decided that the browsing strategy should be designed based on levels of difficulty and that the progress criteria should be based on achieved behaviour (successful answer to a question or exercise). The Software Engineer and Psychologist proposed to review a specific subject at each time. The reason used by the teachers to argue this was that in one classroom children of similar age have different levels of cognitive development, so, they should be able to browse to reach different levels of the software. This idea was expressed in Session 4:

TE: That is, to call the levels so, because, to call it one for Kindergarten, other for grade one, for grade two

TM: Sure, rather, [it should be] based on achieved behaviour

TE: By age rather than by maturity

TM: Rather than by chronological age. Because that's the problem we have, and also that we have the children divided by chronological age which are not their cognitive age. That is, they do not have the same level true? and you can clearly see this, there are very immature children

...

TM: I like this level, that is, the little ones to stimulate basic skills and without levels [grades]. That is, like a game in which he ascends, it becomes a little more complicated each time, and I wouldn't do it using levels of [depending on] grades, absolutely, rather for the user group, say between ages 4 and 8 , that's what I would do.

TM argued that it should be based on 'achieved behaviour' rather than age, because, she argues, their chronological age is not their cognitive age, so they would have different achievements. Her argument is the same that Hammersley (1990) uses to explain the reasons for the way teachers structure their lessons, he says:

> This pre-setting is designed not only to ensure that pupils are taught something 'new', that they 'keep moving', but also that they have the resources to understand what the teacher is to teach.
>
> (Hammersley 1990, p.47)

The Psychologist and the Software Engineer, on the other hand, proposed that the user should browse through the different subjects and grades (these dialogues were presented in the group of categories 'Contents Organisation' above).

This decision supports the arguments presented in the previous category, that is, teachers were designing a mechanism in which the pupil and the computer would be able to negotiate the level of difficulty while browsing, so pupils would have access to the information that suits their needs.

Coherent with the previous arguments, the teachers suggested that the user should be able to go directly to different sections of the software, using the same argument about having pupils with different levels of development (knowledge) in the same classroom and therefore pupils should be able to select their appropriate level of difficulty.

7.3.8 Browsing - Medium

At this level the teachers, the Software Engineer and the Psychologist continued with their different points of view on the way in which browsing should be designed. This time they argued using concrete examples of what the software should present and what the user should do there. In this category, teachers engaged in an additional discussion with the Software Engineer and the Psychologist. This discussion was related to the definition of the 'end' of the software.

- **The browsing strategy**

While discussing the browsing strategy, there was an additional issue that appeared at this level of abstraction, based on a proposition made by the Software Engineer and the Psychologist which the teachers did not understand. This was that the Engineer and Psychologist proposed separating the software into two different sections, one called 'learning' and the other called 'advancing'. In the former the pupils would be learning different skills while doing exercises because the computer would tell them the right answer if they did it wrong. In the latter the computer would not tell the pupils the right answer, so they would need to go to the former section to learn it and then return to the 'advancing' section to continue[18].

[18] The dialogues that illustrate this point are several pages long and are about the explanation of a schemata that the Engineer and the Psychologist draw on the blackboard of the meeting room. So it is not included.

This view of the browsing in the software separating the 'learning' from 'advancing' separated the teaching into two dimensions, one with playful interaction, where pupils used a game-like software and a second dimension where pupils would learn because the computer tells them the right answer. The fact that these teachers did not understand this point could be a consequence of their teaching experience in which this division is meaningless. In fact, dialogues between teacher and the pupils follow a different pattern, which is closer to having cycles of Question, Answer, Feedback (as described by Hammersley, 1990), rather than presenting a set of questions and then offering an explanation only if pupils do not give the right answer.

In later sessions the Software Engineer and the Psychologist changed the scheme to match the teachers' expectations, proposing a structure with multiple browsing alternatives, like the one showed in figure 7.5 (this structure was described and illustrated in Session 9).

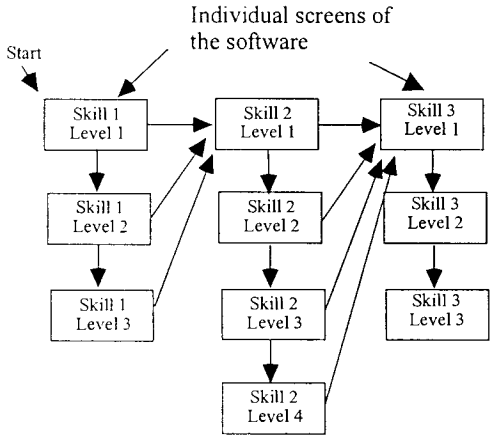

Figure 7.5. Browsing structure of the software.

In this structure, users would be able to browse through different levels of difficulty and skills, accordingly to their achievement. For example, if users could solve exercises only at difficulty 'Level 1', then they would be able to see three illustrations only (i.e. 'Skill 1', 'Skill 2' and 'Skill 3' in row 'Level 1'). If users could solve all the problems, then they would be able to browse through the ten illustrations.

From the teachers' point of view the rationale was that the user should be able to browse through this story using different paths, but all of them should constitute a coherent story. This illustrates the argument presented before: they were using their knowledge about lesson design principles to implement the software, that is, the timing of the lesson to enable all pupils finish the story and so that all of them could reach an end, described by Woods & Jeffrey, (1996).

- **The end of the software or of the story**

Another issue at this level, was to define if the story would have an end or not. The Psychologist explained the browsing path of the software and at one point he said "and this is the end". The teachers disagreed with this claim, arguing that the story should not have an end, so that pupils could imagine different endings. The point here was that apparently, the Psychologist was speaking about the end of the software (at some point the user should arrive to the last screen of the software, or simply quit from it), but the teachers were speaking about the logical end of the story, not the end of the software. This misunderstanding was resolved by deciding that the software would have a set of situations that would be linked together by the context. This linkage was designed in order to avoid the feeling of being lost in a set of disconnected situations and it was supported using arguments about the classroom experience of the pupils. The dialogues concerning the misunderstanding lasted several sessions and they are difficult to understand if they are presented in pieces, therefore they were not included here. The dialogue where they came to an agreement occurred in Session 10, it is:

SE: Or it is not a key factor, [the one] about the child knowing where he is standing ? Without knowing where he is in, or what he is doing ? That is more important than knowing, say, where he is standing in the software.

TE: I think that that depends, we should see really in concrete what happens but ...

TM: I think that what will happen is that he will feel like strange because in general he is accustomed to be confronted by structured situations with a start and an end.

SE: OK

TM: Then, perhaps this will provoke him, a, a, a [disorientation] yes, but I don't know if this could be negative for the child

...

TM: To confront him/her with a new situation, that is, in general one says, OK here he started, here is the middle, and here we end. Not this which is different

Here they started considering the pupils perception of the activity (they should know where they are in the software) and they supposed that if they were in an disconnected situation they would feel strange or lost, and therefore the story should be coherent and have a start and an end (this is the argument of TM).

This separation of the path of the story from the path of the software highlights the image that these teachers share while designing the software, and how it is different from the ones of the Software Engineer and the Psychologist. Again, the teachers are trying to design a complete and coherent activity for the pupil.

On the one hand, the teachers were thinking about the user in the classroom context, they imagined the pupil browsing through the software and enjoying the experience. On the other hand, the Software Engineer and the Psychologist were thinking about the software as an artefact, as something that should be developed. These different layers of software design were a consequence of their different professional backgrounds and in fact, the teachers, brought their actual professional practice into the design. The Psychologist and the Software Engineer designed at the layer of developers, referring always to the particular software that they designed, ignoring the vision of what would actual happen when the software was used in the classroom. This situation is rather common while developing software and as Winograd, (1995) points out, new environments for software design are needed that enable to share the ideas and conceptions about the artefact that is being designed.

7.3.9 Browsing - Concrete

At a concrete level they designed the browsing strategy through the different illustrations of the software, they discussed the possibility of repeating one illustration, that is, if the user would be allowed to go two times through the same screen or not. For example, in Session 9 they discussed this possibility, arguing that if screens were repeated, the user would not realise how to browse to a different screen. The dialogue was:

PY: Yes, but as SE says it is not necessary. Because the idea, I was telling this only in order to avoid repeating the same screen, and then, he would not know how to quit. OK. I did this and how do I quit from here ?

TM: And if they want to play it again ?

The same decision with similar arguments was presented in other dialogues that is, whether the user should pass or not through the same illustration twice (i.e. Sessions 9 and 11).

One discussion that lasted several sessions was about what happens if the user did one or two exercises wrong in one illustration that had, for example, four exercises. The decision here was about how should the software react? Two possibilities were considered: leave the user at the same level of difficulty in a different area or progress to the next level of difficulty. This question was addressed in the group of categories 'Teaching Strategy', because in order to decide it, they started to talk about teaching strategies.

Another interesting issue here was the decision made by the teachers that the pupil should not know what would happen next in the software, that is, they decided to 'hide' the strategy from the user. The following dialogue presents the discussion when the Psychologist was exemplifying one scene of the software and said that at the beginning of the software a character should tell the user what would happen next. TE reacted to this proposition and argued that this character should motivate pupils to go forward in the software, but it should not tell them what they would find next. The dialogue was in Session 10:

PY: Look, here he will play in this thing [screen], and the little animal can speak this way, this [a stone] can be moved here or there and can tell him all the story, that is, the possibilities, or not?

TE: I think that no, I don't know to which extend you say 'tell', because, where is the fun if it tells him beforehand what will happen

PY: No, no that it tells him the possibilities

TE: Here, I think that in this illustration where the elements are, there should be a friend, tell, that is, tell, rather, inviting more than anything, in some way. Not tell him. That is, motivating, that I think is what you want to say, or not?

...

TE: Motivation should be at the start, but motivation. That is, an invitation, something funny, so that he engages to get into it. But not knowledge of what will happen.

As shown, TE reacted to the Psychologist's comment. First, expressing thoughts about hiding what would come next and then expressing a conviction about it. Her argument about providing motivation but not certainty of what would happen next, could be understood as a teaching strategy described by Edwards & Mercer (1987) as: "Never defining (for the children) the full agenda of any activity or lesson in advance".

7.3.10 Interaction - Abstract

In this category, they defined the interaction style that the user would have with the software. The teachers defined the metaphor that the software would present to the user. The way in which they designed this metaphor presupposed an adventure for the pupils, that is, they imagined what they would feel and think when they interacted with the software. They used expressions like "they will feel like in a dream", "they will go into the room", "they will be afraid". While designing this interaction they placed the user as a protagonist of the story, creating a fantastic world where the pupils would interact with the objects in it.

The decision about where to place the user was a choice between two options: inside the computer or outside it. In the former the user would identify himself with one character of the software. In the latter, the software would give the feeling that the user is manipulating the elements of the software, in this case the screen would act as an extension of the eyes, and the mouse pointer should act as the arm and hand of the user. An example of the discussion is presented in the following dialogue of Session 14, where the Graphic Designer proposed that the user should be physically represented inside the screen. TM argued against it, because the image of the character would not, necessarily, resemble the user's appearance.

GD: Why not. In my opinion, one of the characters that you can see there should be the pupil. That is, that the pupil sees himself coming in through the door, that they can see the pupil watching, that is not like you say, that the pupil will be the user, he should be the character.

SE: But there are scenes like.

TM: And what if I don't feel represented through the one that is there [in the screen] ? If I am, for example, <referring to the PY> blond, thin and with white skin <whereas the PY has dark hair and skin>.

After this argument they decided to place the user outside the computer, providing enough visual stimulus so that the user would 'feel' manipulating the objects and participating in the experience.

This design of the interaction could be interpreted by saying that these teachers were "planning their teaching to include activities which would give children direct, concrete experience, and which would require them to act, not just listen, read or write" (described by Edwards & Mercer, 1987 as a principle that 'progressive' teachers follow). Such a view might be supported by the argument that they placed the user directly interacting with the objects and characters of the software, providing direct and concrete experiences for the users. An alternative interpretation for this design could be that these teachers were trying to stimulate the pupils' fantasy, providing realistic and relevant experiences that would encourage their sense of involvement (described by Woods & Jeffrey, 1996 as a common tactic of primary teachers).

At this level they started with the definition of the main character of the software, that is, the 'person' that would give instructions to the users and that would guide them through the different stories. The instructions that this character would give to the users, were designed like guessing games and question and answer sessions.

7.3.11 Interaction - Medium

At this level they designed the general interaction with the computer following a well known pattern. That is, the computer gives instructions to the user, the user follows these instructions (interacts with the computer) and the computer provides some feedback for him. During the design process they enumerate several different styles of instructions, interactions or feedback. Following are some examples found in the transcripts[19].

- **Instructions**

The instructions for the user were presented in one of several ways:

- A specific exercise to be done, for example a sequence of points to be linked (Session 3), a path that should be followed (Sessions 3 and 6) or to erase a figure in the screen (Session 3).
- Written or oral instructions (Sessions 2 and 5) that could have the form of a direct order to the user, for example: "place the stone in front of the tree", "now move forward" (Sessions 7, 12 and 14) or it could be presented as a problem to be solved, for example: "the child wants to go down, could you help him ?" (Session 11).
- An open place with things to discover (Sessions 5 and 6). In this case there were no instructions, but they assumed that the pupil would try to click everywhere to discover all sounds and effects.

Their final decision about how to give the instructions to the user included the last two types of instructions, that is, they presented to the users the problems to be solved and provided a rich scenario with different elements to discover.

- **Interaction**

The interactions of the user with the software were basically of three kinds:

- To draw something on the computer screen, in this case the idea was that the user should be able to 'create' something that has a consequence on the screen, for example, to draw lines or to erase a figure.
- To manipulate objects on the screen (Sessions 3, 5, 8, 11 and 12).

[19] In order to present the different ways in which they design the interaction some examples from the data are provided, but not the transcripts themselves, because these are generally long sets of examples of the kind of interactions designed and therefore add little to the description presented here.

- Free interaction, basically point and click (Sessions 5, 6 and 13).

Their final decision was that users could manipulate objects on the screen and that all the objects on the screen would respond to clicks (i.e. free interaction).

- **Feedback**

The computer provides feedback for the actions of the user as:

- Sounds, noise and movements (Session 1, 6, 8 and 16).
- Explanations to the user, for example, telling the pupils what elements they were clicking on (Session 6).
- Positive feedback to the user, for example. "good, you made it" or applause (Session 7).
- Teacher of the users (Session 8) or guiding them in a given task (Session 3). This dimension is analysed in Section 7.7.

They decided that the computer should give positive feedback to the users if they solved the problem presented, and it should repeat a modified version of the instruction if they failed. Also, they included noise and special effects for the different objects on the screen.

The interaction designed by the teachers had two dimensions, the first related to the learning activities that the user would carry out (instructions, manipulation of objects and rehearsal), these were designed as question and answer dialogues between the character in the software and the user. The second type of interaction designed was a 'playful interaction', where the user could discover things on the screen and receive feedback such as sounds or movements. This latter interaction had no specific learning goal, it was designed basically to achieve an emotional goal, related to the surprise and motivation of the pupils.

7.3.12 Interaction - Concrete

At a concrete level they designed the specific actions that the user could do within the software and the feedback for these actions.

There is one general issue here to be noted, which was that they designed the reactions of the different objects on the screen based on the everyday world of the child's experience. For example the lamp could turn on and off, the radio could play music, the shoes could move. These elements were not part of the 'learning' activities

of the software, they were 'toys' designed for the users, but they spent lots of time designing the interaction with these specific elements and they wanted every element in the software to give feedback to the user.

This principle of realistic behaviour was also applied to the specific exercises that they designed. For example, they designed an exercise where the user should paint a wall and they designed the interaction based on realistic movements ("moving the brush sideways only": Session 17). The type of objects to be manipulated by the users were from their environment and should have a realistic response to the user's actions.

The way in which they designed these interactions was by first defining the exercise that the user would perform and then designing the environment and the objects that would be manipulated in this environment. For example, they needed a river or lagoon to place fish inside (the user was asked to order them from smallest to biggest) and also to put ducks on (the user was asked to order them from the biggest to the smallest). In this sense, the interaction with the software was designed as a consequence of the learning activity that the pupils should perform. It would seem that they saw a necessary connection between the design of the users' actions should and the software interaction design.

7.3.13 Interface Element - Abstract

At this level they defined the general shape of the interface, they discussed the colours to be used and the general appearance of the different scenarios. In this sense, this general level of design of the interface elements was the implementation of the decision to use a story and it included the selection of the different scenarios. These illustrations were designed corresponding to three different scenarios in which the stories occur. One was the farm, the other the neighbourhood near the school and the third one was the school.

The interface design task was delegated to the Graphic Designer, who produced paper drafts of the different illustrations that were then critiqued by the teachers.

7.3.14 Interface Element - Medium

At a medium level they designed the different objects of each screen of the story, such as rooms, surroundings and the artefacts to be found in each room and they also designed the character that would play the role of 'guide' in the software.

- **Realistic scenarios**

The teachers designed an environment for the users, they created an interactive experience where they would browse through a set of rooms and places (the farm, neighbourhood, school) experiencing different situations. In this sense, the interface of the software was transformed into a 'window' to this fantastic land and the human-computer interface elements were the objects that would look and behave as if in a fantastic but real world. In fact, the teachers argued repeatedly that the elements should be like the ones found in their environment. The graphic style used was supposed to be for children of age 4 to 6, so it was somewhat fantastical.

Perhaps surprisingly, the more traditional elements of a human-computer interface, such as buttons, dialogue boxes, menu bars, icons, etc. were not designed. The design and implementation of these elements was assumed by the Software Engineer.

- **The principal character**

The other main issue at this level was the definition of the principal character of the software, who would guide the user 'through the learning experience'. That is to say, at the beginning the character was defined as a sort of narrator for the user. In the following dialogue, found in Session 7, TE described some of its characteristics:

TE: But, I thought that we must have in the software a narrator. A fixed narrator in charge of telling [the story], a narrator that would engage with the children, that would be very meaningful for them, very contextualised and that this person, animal or so, was the one who had to [guide the user].

...

TE: Or a character that is typical, like a pet, I don't know

As TE said, it ought to be meaningful for the pupils, it should be a special person able to engage children in the story and also provoke feelings of sympathy (like a pet). This character was considered as the central character in the software (its description and specific design lasted several sessions, particularly Sessions 14, 15, 17 and 18). This character was the one that would tell the users what to do, and would provide them feedback when required.

The teachers defined this character as: a friend of the user, wise, saint, a good person, patient, audacious, helpful and with a friendly voice (female if possible). Its appearance had to be fantastic (not a real person-like character) and its description ranged from a clown or a joker to a teddy bear or an alien.

As mentioned previously, the role of this character was to guide the users through the software and show them the different possibilities, behaving like a partner during their learning adventure. They implicitly defined the feelings that this character should provoke in the users as sympathy, trust, admiration, etc. Although there was no explicit reference in the data to this character as assuming the role of the teacher, the description of the affective relation with the users was very similar to what is argued to be sort of relation described between pupils and teacher in the primary classroom by Cooper & McIntyre, (1995).

7.3.15 Interface Element - Concrete

At this level they defined the look and appearance of the different scenarios that they had designed before. They reviewed sketches that the Graphic Designer proposed and critiqued them.

What came up clearly in the data was that they wanted each environment, and all the elements in them, to be copied from their real environment. They argued against including foreign scenarios (like Africa or the north of Chile[20]). To show the idea, in the following dialogue the Graphic Designer proposed including some illustrations of the north of Chile (desert), but TE rejected the idea and proposed something closer to their own context. The dialogue was found in Session 15:

GD: For example, if it were the North [of Chile], it could be the desert, a cactus, I don't know, colours of the desert. Also, that he should take the opportunity to try the basic skills and also to know it.

TE: Look that, with these trips, we are taking it [the software] out of the context, a little [out] of what we though at the beginning. That is, it could be like a trip also, but on the other hand, it would be like any neighbourhood <pause> a bus.

GD: Something more

TE: Something more within context, closer [to the user].

Here TE was looking for images that were from the local context of the user, when she referred to the 'neighbourhood' she was referring to the school's surroundings.

They spent a lot of time designing the different elements of each scenario and they did not design 'typical' software interface elements like buttons, dialogue boxes, menus, etc. Their focus was to design the scenarios and objects in them.

[20] The city where the school is located is in the south of Chile.

7.4. ACTIONS

As explained in a previous chapter, the group of categories 'Actions' refers to the activities to be performed by the teachers or pupils as individuals or in the classroom setting, interacting either with the software or with the pupils.

The categories of this group are separated into two layers, the classroom setting and the individual setting. The former is related to the roles of the teachers or pupils, considering the classroom as a place where the teacher is in front of the class interacting with the pupils or the software and the pupils interact as a group with other pupils or the software. The latter is related to the teachers' or pupils' role when they interact with pupils or software carrying out an activity that is essentially individual (for example if a group of pupils are supposed to watch the screen of the computer and think about strategies to solve a problem, they would be doing an individual action interacting with the computer).

7.4.1 Actions of the teacher in the classroom with the software

As mentioned before, this category dealt with data describing the actions that teachers could perform in the classroom context using the software. It will be shown that teachers rejected the idea of designing interactions of the teacher with the software, arguing rather that children should be able to learn by themselves and that the teacher should guide them in this process without necessarily manipulating the software.

- **No interaction with the software**

To present the idea, there was one interesting discussion involving the teachers, the Psychologist and the Software Engineer about the need of a role for them. The Psychologist and the Software Engineer proposed that the teacher should interact with the software assigning the different levels of difficulty of the software, taking into consideration the particular level of each pupil. That is, they proposed that the teacher take the role of strategic manager or task setter for the lesson, tuning the software to the particular needs of the user. To exemplify the type of design, we present what the Psychologist said in Sessions 6 and 7:

PY: Also, we, as I told you before, this screen will be for the teacher's use, that is the teacher will choose where to work. Now, it is probable that it isn't so, it is probable that you leave the child to choose where to go into, I don't know

.....

PY: That is, the teacher, he will appear here in the presentation, for example, [assigning level] number 1, supposing 6 levels of deepness, then, but this will not be presented to the child, the child will not know about it, he will not know that there are different levels of deepness, instead the teacher will say 'look, this child, that is in these conditions, assign him a level 2'. Then he does a control 2, the teacher, without the student noticing it and his story will be the same, exactly the same, but what [the software] will ask him or what the computer will demand from him to do will be of a higher range.

In Session 6 (first paragraph) the Psychologist explains that a certain screen of the software will be for the teacher to decide where the pupil should work and in the second paragraph (Session 7) he gives an example of the teacher assigning levels of the software based on the pupil's actual state.

The teachers, on the other hand, rejected this idea, by arguing that the design should contemplate the possibility of the pupil interacting with the software alone, without the need of tuning the software before. They described the teacher's actions in the classroom as interacting with the pupils, rather than with the software.

The argument for this position is presented in the following dialogue between the Software Engineer and Teacher M found in Session 7. Here, the Software Engineer argued that the software should be a tool for the teacher and that they were not trying to replace the teacher with the computer. TM argues against that, she says that although the idea that teachers could feel displaced existed, they wanted to be replaced in some tasks, because the children should learn by doing. Finally, she gives the possibility of including both types of uses, alone or 'in the classroom' (with the teacher).

SE: What I had clear until today is that the software is a tool for the class and it is not a teacher. That is what I don't, what makes me think: Until now we are building a tool for the teacher, there are problems with those children and this is another tool. We will not bring this thing here and put a hut on the computer and [expect that] the teacher goes home. We are giving the option that if the child has the teacher, that he can offer <noise>

TM: Look, to realise that it should be the contrary. That is, we should be discussing the possibility that we are required [necessary], but it isn't so SE, because at no moment, that is, for example, I know that at some point Lucio said, the teachers perhaps could feel a little that they will be displaced, [but] no, he is absolutely wrong, then no, we never thought this.

SE: Do you know where I am going ?

TM: It is the contrary, we think that now that learning by doing is very important, and that the child is constructor of his own learning and we are guides in this.

SE: I am thinking which is

TM: No, but I am telling you which is the position of thinking that the teacher is not required, that is, maybe it could be better giving the option that it could be worked in class or that the child could interact alone at any moment.

This dialogue shows that the underlying reason for rejecting the idea of the teacher interacting with the software is related to the teachers' conception of learning. That is, they draw on methodological arguments that justify their design decision. She argues that pupils would 'construct' their own knowledge, drawing on constructivist theories of learning.

7.4.2 Actions of the teacher in the classroom with the pupils

In this category, the teachers described their role as helpers for the pupils in the use of the software, but not in the learning activity, and as classroom managers.

- **Helper with the software**

In Session 5, TM described the role of the teacher while using software, separating what is done in the 'classroom' from what is done in the computer lab:

TM: The role of the teacher with the software is nothing else than being there. Helping [the pupils] to get in or out of the software, because the rest of the work she will do it in the classroom, and I think that's the idea and that's the way it should be, because, we cannot think that it effectively will help us to cover subjects, because that's impossible.

Although TM mentioned the teacher's role "with the software", she was situating the context of her action in the computer lab and then she exemplifies her role as "helping [the pupils] to go in or out of the software". This claim was relevant because she stressed the fact that the actions of the teacher were limited to help in the use of the software and also because she clearly specifies that her role as teacher would occur in a traditional classroom.

- **Classroom manager**

In a similar vein, in Session 7, TE argued that in the computer lab the teacher must do "other things" (other than in the classroom), and that the pupils must "develop the software" meaning "following the lesson" that was built into the software. The

teacher's role in the computer lab, she says, is to ensure that pupils behave properly and that they do what they are supposed to do. The paragraph is:

TE: Yes, because the ideal is that the teacher here does like other things. That is, being the guide means that she has to worry about, to start with, that it is done what should be done, [this is,] using the software; that children are rotating; that many other things are happening; that it is happening the fact that, that they have equal chance, all the children, that is, he [the teacher,] is doing a lot of other things. That is, at no time can he go home, because it is likely that the computers will burn, or that someone goes for water and pours it over the computers or turns them over the desk. That is, teachers cannot be absent, but I refer that , could be better, I don't know. But this seems to be well structured and it points also to something that I have seen in few pieces of software, I haven't seen that many anyway, but I have seen few, so as, well structured for a lesson, you see ? <pause>. Not like the encyclopaedias, that when you look for something , you have to wait.

Here TE described her role as managing the classroom, not as scaffolding, tutoring or helping pupils to follow the learning activities of the software. She also argued that there is a lack of software structured for a lesson, meaning a piece of software designed to support a lesson.

7.4.3 Actions of the teacher individually with the software

This category includes data describing actions that were designed to be performed by the teacher interacting alone with the software. In this piece of software actions designed for the teacher but that imply some classroom management were excluded (for example the discussion about the need for the teacher tuning the software of section 7.4.1) .

There were two descriptions of teachers' actions in this category. The first one concerned the possibility that the teacher could control the speed of display for each screen, arguing that in this way the child could try to discover the image. The second one is that the teacher could interact with the software after the pupils use it in order to evaluate their progress. Both ideas were rejected or dismissed during the design process. The data shows that these teachers considered the idea of interacting with the software and that they explicitly rejected this interaction.

7.4.4 Actions of the teacher individually with the pupils

This category includes data concerning the interaction between the teacher and the pupil that was not specifically designed as situated in a classroom. Nevertheless, to understand and interpret what the teachers said here, it is necessary to take into consideration what they described as their actions in the classroom context. The data in this category adds additional support to the idea that they designed the role of the teacher as helper of the pupils with respect to the use of the software and not the understanding of the contents to be learned.

Following on from the descriptions given in the previous category from Sessions 6 and 7 in Session 9 these teachers continued with the description of their actions, arguing again that they would interact with the pupils only to help them with the software. In this paragraph TE described the teacher's role through the example of a child that has problems. At the beginning of the paragraph it is possible to think that the child could have a problem with the contents or similar, but then she clarifies the type of problem by copying the child's expression: "I quit from the software" and it is there that the teacher should have a role:

TE: What happened when a child had problems, he couldn't go through, it isn't the computer, after thinking, it isn't the computer the one who should guide him. It is the teacher, yes or no ? The teacher is the one that should come close to him in this minute or when the child calls her and so on. The children, when they quit, [for example] the ones from Kindergarten from the software, when they are working they say to you 'aunt I quit', and then, see, the teacher must have a role there. That is, telling him to try again, or give him an instruction, I don't know.

This claim is consistent with the previous arguments about the teacher having a role of helper in the use of software and managing the classroom. Although in this case they did not explicitly reject the possibility of helping the children with the contents, given the examples of the previous categories, it is possible to conclude that they do insist on this exclusive role.

7.4.5 Actions of the pupil in the classroom with the software

There was very little data that were categorised as the design of actions of pupils interacting in the classroom with the software. What was expected here was activities designed where the software would be part of an activity that is performed in the classroom. This would correspond to the category of 'teaching centred software'

described in the literature review (Section 2.2.3 in Chapter II). The absence of data categorised here shows that teachers did not engage with this aspect of design.

7.4.6 Actions of the pupil in the classroom with other pupils

This category includes data categorised as the design of the interaction between pupils in the classroom. In this sense, the data illustrates that the teachers did talk about pupils working in groups or teams in front of the computer, similar to reports found in Crook (1987).

In Session 2 the teachers proposed that the software should allow work in groups, because the school has few computers so they can not accommodate all the pupils working in the lab. They also said that working in groups would contribute to solving the problems of pupils with learning difficulties. Later on, in Session 5, they clarified the concept, arguing that the software should not be designed in such a way that you need a group to interact with it, but it should be designed so that you can work as a group with the software.

The teachers' aim seems to have been that the pupils should feel that they belong to the team so that they were motivated to work towards a common goal. This idea was presented by TE in Session 5, where she starts by talking about her experience using the computer lab with several children. Based on this experience, where she needed another teacher to control the pupils, she argues that group work does solve the problem, because children are aimed towards a common goal and they engage in being part of the group.

TE: The only thing that I can talk about is my experience that I am two years giving lessons of [with] IT trying to use the pieces of software that come here and with the small children, with the ones of Kindergarten, and there what happens ? We come with the k-teacher trying to do a lesson with them, but the older, the more you can increase the number of pupils in a group, that is, with luck and two teachers you can work with groups of five, of 30 or 25 pupils. But with the younger ones, you can't, because the child is jumping or he goes home. Opens the door and he's gone, he has a minute and he didn't touch a computer and he goes and "I go home because I didn't touch no computer". We saw even these ones, that's why with the [number of] computers we have, which is our reality [because] there are no more [computers], that allows to play in teams, that they inscribe me [the child] or I choose a character that represents me as a player, I don't know, that's an idea that occurs to me now, because they can't write, the elders of grade 2 could write their names and play in a sort of team

...

TE: It is incredible, but if you don't identify your self it does not give you a sense of belonging, that is, it permits you, that the child says: I can't go out there while I am in the lesson, because its my team I must, at least that he says to you that it is not necessary that he drives the computer, because that's not the goal, true ? That is, that he keeps saying, hey, you have to drag it there, don't do that, no don't do it so fast, so that the others can

Through these claims it is shown that teachers had the idea of designing the software for team work, but the underlying reason for this was that if the pupils feel compromised, they will keep doing what they are supposed to be doing, instead of going out of the room or making a noise. In this sense, they were talking about classroom management and they consider the work in teams as a strategy to cope with disciplinary problems. Also, it appears clear that she was not designing collaborative work necessarily, she only explained the reasons for having team work. Anyway, this idea of designing the software for team work was not continued after these initial dialogues, in fact they did not design any special features in the software to enable or encourage this type of use.

7.4.7 Actions of the pupil individually with the software

This category includes data concerning the design of actions for the pupil interacting directly as individuals with the software. The type of actions designed were consistent with the group of categories 'Interaction' described before (sections 7.3.10 to 7.3.12). There are additional data in this category which supports what was previously described as what pupils would be doing in the lesson with the software. The interaction with the software from the pupil's point of view was described through examples of what they should do or feel while using the software.

The actions of the user were defined across the sessions in a way that seemed to closely follow a well known pattern in teaching, that is to say: "the use of techniques like the 'guessing game' question and answer sessions, to elicit 'key ideas' from children rather than informing them of these directly" (Edwards & Mercer, 1987). In fact, in Session 2, TM described the type of interaction using examples of known 'guessing games', and she described pupil's reactions to this prompting:

TM: For example, with guessing, with questions, [for example,] this way: 'how many ducks are in a basket ?', 'if there are so many ducks and so many', and <moves her hands like keyboarding> and then he goes into another thing. In order to have mental agility, because you get sleepy only by watching them, few light. But the thing should be so well done that it goes on changing so that he does not memorise it

...

TM: That one could change something in it. For example, at one time guessing, afterwards riddles, afterwards to complete, a proverb: To a given horse, and that he puts, you don't look at its teeth, OK, So

TM describes a user who would be engaged in solving these type of questions and she was concerned about having a sufficient variety of such games. In Session 5, TE continued with similar examples, and she contributed to the justification that in this way children would not realise that they are learning:

TE: Sure, he doesn't know, he doesn't know that he is learning sensory motor co-ordination or anything, he is playing something, in the [text] book the first [chapter] one appears, to say, for example, 'A' comes sensory motor co-ordination, 'B' comes an other thing, 'C 'another. That is, it isn't necessary, the child [that] works in one sheet three or four [basic] skills [at once], without problems, that is, there isn't this fine thread

While arguing that the children would engage in a playful interaction, without realising that they were learning, she was 'hiding' the learning goals and activities from the children (as Edwards & Mercer, 1987 describe that 'progressive' teachers do). In fact, in Session 8 TM argued in a similar vein, that for the children these activities would be games only:

TM: Clear, it is for rehearsal and that he practices them, I think that the goal is that he plays with these things that at the end they have no idea for what they are, because, for the children these are games only, ...

In this case TM added the comment that the children would be rehearsing the contents while playing with the software.

In summary, the specific actions they design were those of the pupil responding to the software's prompting or instructions. That is, the computer is questioning the pupil about the material to be learned and the pupils' role is to answer these questions.

The fact that the teachers visualise the pupils' responses to the software's prompting in the same way as the literature describes pupil's responses to the teacher's prompting (cf. Hammersley, 1990), supports the argument that they were designing the way in which pupils would be learning while using the software.

7.4.8 Actions of the pupil individually with other pupils

This category includes data illustrating conversations in which teachers visualised what would happen among the pupils. As mentioned before, these conversations were not planned, but they realised that these interactions would happen. In the early sessions they talked about competition between pupils and about collaboration. The former appeared as conversations between pupils about the advances (moving forward) in the software, the latter, appeared as a sort of scaffolding between peers.

In Session 2 TE and the Software Engineer talked about the type of dialogues that they imagined would happen amongst the pupils. TE reacts to the Software Engineer's example regretting the use of the curriculum subjects, but she agrees in the style of conversation:

SE: But, there will come a moment, I imagine, where they will sit and say: 'in natural science I am in the 6th [level], in social I am in the 2nd'.

TE: No, but it shouldn't be, that's why it shouldn't be subject oriented.

SE: that's what I refer to

TE: Because, for example, the discussion should be different, the discussion with his partner should be: 'in ecology I am in this stage, in which are you ?'

The main change that TE made to what the Software Engineer said was to change the name of the subjects. Both were imagining that this type of dialogue would happen between pupils, but they did not include explicit design elements that would provoke this to happen.

The actions between pupils that they described were similar to a particular kind of scaffolding, that is, one pupil helping the other in a certain exercise. In the following dialogue found in Session 8, TM speaks about her experience as a mother with two of her children. She explains that the older one helps the younger in some exercises and she generalises this experience to other children. The Psychologist asks about their age and she answers that they have different levels of knowledge.

TM: All what gives you the quantity in mathematics in pre-calculus, for example, ..., there are two children they help each other, you know. Because, because my little children when they work together they help each other, the older says look who has more, more is asking you, you see. They reinforce each other, that's why I have the experience with the children and know that they reinforce each other. Now I don't know if it corresponds to the psychological level that it is done so, I got no idea, but I know that children reinforce each other.

PY: Of the same age ?

TM: Yes, because they don't have the level of experience and the degree of learning exactly the same. Therefore one will always know more, except if you sit two that are very similar in front of the computer.

Again, they imagined that children would co-operate within the groups, helping each other in areas where they have different levels of knowledge, but they did not design special features that could trigger these interactions. They saw it as 'naturally' happening when children work in groups.

7.5. AIM

This group of categories includes the data that were categorised as expressing the aims that the group assigned to the software or computer regarding the pupil or the teacher. For the purposes of this analysis, claims about software or computers were differentiated by the type of reference they make about it. That is, if they spoke about a generic artefact that could be any software, it was classified as claims about computers. If they spoke about any specific software product or 'idea ware', it was classified as software.

7.5.1 Aim of the software for the teachers

This category includes the claims made by the group about the aims of the software for the teachers. That is, the purpose that the software should serve for them. In this sense, it will illustrate that they defined the software's role as rehearsal or as a complement to the contents of some lessons. That they proposed the aim that the software should save the teacher time.

In the following dialogue found in Session 5 TE expresses the idea of the software as a time saver, and as a result of questions from the researcher (RE), she explains in more detail what she was trying to say and presents arguments that it is effective in that role:

TE: It is supposed that all what comes out of the software, the teacher will go through it in the classroom. What happens is that perhaps now, instead of being ten days covering the subject, [she] will do one session [in the classroom] and will bring them here [to the lab]. She will put the software in the equipment and its over, you see. It will save a lot of time and effort to her.

RE: But this results so, has it happened to you with other software ?

TE: Yes, of course.

TM: How,, lets see ?

RE: For example, if you went through a subject about biology ,for example, during four weeks, and now you do a general view of the human body and the child comes here and learns ? Comes back, or with other software ?

TE: It is the same as if you view a movie, for example, it has happen to me with the human body, with the [software] one about indigenous [cultures], with some slide shows that I downloaded about some topic. Because it isn't the same as the blackboard, with the chalk and the papers with colour pencils, it isn't the same that something that has sound, it's different.

RE: But is it effective ?

TE: It is effective, the fact even to reinforce a session with a video is very effective, it gets you closer to reality.

TM: The computer in it self is a stimulus, not perhaps so much value in the contents or the way to expose the content. Because, in this sense, one uses it more as a complement of [the classroom activities], in the case of delivering subjects. In this case it will be the same.

In this case TE's beliefs about the potential of the software as a complement for the lesson's contents, was based on the possibility of having multimedia software. TM, supported this claim confirming that it would be a complement for the teacher (she speaks about the computer, but because of the context of the dialogue is included here). In this sense the software's role is to expand the resources that teachers have to teach certain subjects.

7.5.2 Aim of the computer for the teacher

The data categorised here will illustrate that the teachers argued that the computer could serve as a motivational tool for them. That is, they could use the computer to motivate the pupils to do certain activities that with other type of resources pupils would not do. Also, it will show that they said that the computer should not act as a teacher.

- **Computer as a control tool**

The following dialogue found in Session 2 exemplifies this claim. In this case TE answers a question of the researcher (RE) about the role of the computer compared with other resources. Her answer takes the form of another question, but she is clear about the role of the computer as a motivational tool. Although she recognises other roles (tool for work) for it, the primary role is motivation:

RE: Which would be the role of the computer, [compare it] if you have 6 boxes with nice slides and children take them ?

TE: And you believe that the boxes engage the children the same as the computer ? Children couldn't care less about the boxes.

PY: These are for motivation only.

TE: Yes, but no, should give <pause> and what other thing is a computer but a tool that helps to motivate ?

PY: Don't know.

TE: It is a tool for work, well but the boss says that for him it is not like any other resource only.

TM: It has a lot of [other] things.

TE's definition of the role of the computer seems to be based on her experience and therefore she had this conviction. The idea about the computer serving as a motivation tool for the teacher was expanded in the following paragraph of Session 4. Here TE argues that the real power of the computer is that it is considered to be a 'treasure' by the children, before this paragraph they were talking about the design of a game and the Psychologist was explaining that the teacher could organise the players, but TE reacts saying that the computer can be used as a reward if the children behave well:

TE: The point is that it is not the same PY. You know why? Because the teacher is a teacher, you see? But the computer is not, if you don't respect what it is telling there [the computer], you missed the opportunity of working with the computer. You cannot work the game. It is part of the game. You loose the most loved, which is to work with the equipment. That is, the computer its true, it can not replace the teacher, but in this minute it is very motivating for the children, then it is like the engagement tool of this.

In this argument TE was explaining the role of the computer as an instrument for the teacher to control the pupils, she says that children would follow the instructions because they wanted to use the computer.

With different arguments but meaning the same, in Session 4, TM argues that the computer is useful because it controls students with behavioural problems:

TM: Like you said TE, think in a piece of software for 'Macana', for the 'Masos', for 'Trabol', something that keeps them busy. But we, [in order] to win, [we] need more space, so that he can practice. Because at what time should I put him in [front of the computer] ?

The examples they gave here ('Macana', 'Maso' and 'Trabol') were children that presented difficult behaviour in the school and she would like to use the computer as

a tool to "keep them busy". Underlying is the same argument of motivation, that is, they would be wishing to use the computer and that is the reason they will "keep busy".

Also in Session 5, TM argued that the computer is a stimulus and that there may be less value in the contents:

TM: The computer in it self is a stimulus, there is not perhaps so much value in the contents or the way to expose the content. Because, in this sense, one uses it more as a complement of, in the case of delivering subjects. In this case it will be the same.

She talks here about the computer, but because of the context in which this was said, it was interpreted as expressing an aim for the software rather than the computer.

Through the previous examples it is possible to see that the computer was seen as a tool for the teachers that solve a control problem. That is, they recognise that for the pupils the computers are fascinating artefacts and they use this characteristic as an instrument to control them.

- **Not a teacher**

Another aim or role that the teachers define for the computer was that it should not act as a teacher. This argument was presented while they were discussing the feedback that the software should give to the users if they did a mistake. In Session 9 the Software Engineer argues that the computer will tell the user exactly how to solve the problem. The teachers argue against that and deny the role of teaching for the computer, they say:

SE: But, it isn't necessary, because when I spoke about this, I meant that after the child tries two or three times, the computer would not tell him no, afterwards the computer tells him exactly the way it should be done.

TE: But, do you know ? That is exactly what I was thinking about, then we would fall into a pedagogical issue. That is, we can not let the computer teach anything, that is, the computer can not teach. What the computer is supposed to do is to give the instructions, take the child, guide him, OK? not tell him straight forward for example, put the Apple into the basket.

TM: [Because] it could happen that he doesn't even try to solve it [the exercise] in order to hear the computer [telling him what to do].

Here TE was clear about the fact that the computer should not teach. The same type of argument is repeated in Session 13, where TE says that the computer should not solve the problem instead of the pupils, that the computer should guide them only:

TE: No, in what I disagree, was about that what we talked once. It is that the computer does the exercise for the child. The computer cannot do any exercise for the children. That is, it can show them, but not tell him, 'look you know that you should do it this way', no he will realise, he will look to his friend.

In these arguments the teachers are not really denying the fact that the computer would act as a teacher, rather they deny that it should provide the correct answer. In this sense, they were applying some teaching strategies to the way the computer should 'guide' the pupils. In fact, by saying that the computer should "give the instructions, take the child, guide him" they were replicating their behaviour as teachers and when they say that "the computer cannot do any exercise for the children, but it can show them how, without telling them", again, they were transferring their own teaching strategy to the computer.

7.5.3 Aim of the software for the pupil

The data in this category will illustrate that the aim of the software for the pupils (from the teacher's point of view) was described as a rehearsal, as a set of activities that would help the student to train in certain areas. The teachers also define the software's aim as a game in which pupils would train certain skills without realising that they were learning. These ideas were part of several dialogues found in the data and some of them were included in previous sections, therefore they will not be included here.

The interesting point in this case is to realise that the teachers defined the aim of the software for the pupil as a learning aim (or part of a learning cycle). They explicitly said that pupils would be rehearsing what was taught in the classroom. In this sense, they designed the software's aims for the pupil as a complement of what is taught in a 'normal lesson'.

7.5.4 Aim of the computer for the pupil

The data categorised here will illustrate that the aim of the computer for the pupil from the teacher's point of view was defined as an attractive and stimulating artefact. They referred to the 'potential' of the computer as a stimulating and motivational resource, as they said in Session 2:

RE: Which would be the role of the computer, [compare it] if you have 6 boxes with nice slides and children take them ?

TE: And you believe that the boxes engage the children the same as the computer ? Children couldn't care less about the boxes.

PY: These are for motivation only.

TE: Yes, but no, should give <pause> and what other thing is a computer but a tool that helps to motivate ?

PY: Don't know.

TE: It is a tool for work, well but the boss says that for him it is not like any other resource only.

TM: It has a lot of [other] things.

As it was commented in a previous category were the same dialogue was used, TE reacts to the comparison of the computer with other resources (like boxes with illustrations), defending the motivation that it provokes in the children. Later in the same session TE continued arguing about the challenge that the computer poses to the pupils:

TE: OK, they win against the computer, but they have to win. I don't know why. It is not like with the boxes, with the boxes they will not care if they don't win. But against the computer, they need to win. Then, they, even if they have problems, but those who do have problems, of [learning]. This can be solved with team work.

In this sense the teachers defined the aim of the computer for the pupil as a motivator, which is consistent with some other findings that say that it "increases challenge, control, curiosity and fantasy that allows for personalisation of one's work" (Schofield, 1995, p. 196). This powerful role of the computer is used by the teachers in implementing the rehearsal activities in the computer.

7.6. USER

As mentioned in a previous chapter, this group of categories contains data regarding the characteristics of the users (teacher, pupil or groups). While designing the categories of the data (the systemic network) it was assumed that they would describe characteristics of each user, but the evidence found was mostly related to pupils. Only a few paragraphs were found where they referred to characteristics of a group of students or to the teacher as user of the software.

Also, in the data there were few explicit references to the user which could be classified in this category, but it is necessary to say that there were several other implicit references to the user in other categories. For example, when they define the human-computer interface elements, they hypothesise what users might find attractive, they design what they believe will motivate them and so on.

In the data categorised here there were two types of user descriptions, one about the target user of the software and the other about characteristics of the pupils of the school.

The target user was defined to be of age 4 to 6, which corresponds to Chilean school levels K to 2. They decided to focus on users with learning difficulties (Sessions 2, 3, 5 and 8) but they also wanted the software to be useful for all the users, i.e. with and without learning difficulties (Sessions 3 and 8).

They characterised the pupils as ones with low achievement, with reading problems and poor vocabulary compared to other children of similar age but from different schools (private schools). Describing the pupils in general, they described them as ones that can sit and follow instructions if the teacher has the control of the situation, that is if the teacher is dictatorial, but they agree that this is not the right system to teach (or at least not the contemporary one). In general, they say, they exhibit bad behaviour in the classroom and that teachers have difficulty in controlling them.

They described the social background of the user as one with low income and with lots of family problems, such as low parental control, low levels of positive feedback, and belonging to families that were not well constituted. This description was supported by demographic data derived from other studies, indicating the social levels of the pupils that attend this school.

The teachers described the psychological characteristics of the target users in the following ways:

TM: They follow instructions, without necessarily understanding them (Session 6)

TE: They have low control over their body (Session 8)

PY: They are egocentric (Session 6)

TM: They have low tolerance to frustration (Session 14).

TM and TE: They say that at this age they engage with stories (Session 6), they like animals (Session 15) and certain colours (Session 16).

TM and TE: They like that the computer gives feedback (Session 1) if it does not, they get bored (Session 3)

The only explicit reference to the user as a group was found in Session 2, where they argued that groups of users should be of not more than four children each.

Analysing this information, it is possible to see that these teachers had a certain stereotype of the user and that this image is used in the design of the software. What is more interesting perhaps is what they did not say. That is, they did not refer to the teacher as a software user, they draw heavily on their own experience, without explicitly describing the type of teacher that would use this software or the composition of the groups of pupils.

Another issue here was that although there are not too many explicit references to the user, they refer to him in several other categories in a more implicit way. They assumed that they knew their pupils and therefore it was not necessary to describe them.

7.7. TEACHING STRATEGY

As discussed in a previous chapter, this group of categories includes data illustrating discussions about the behaviour of the software that revealed some of the teachers' conceptions about teaching. For example, the decision of what kind of feedback should the software provide to the user, or its reaction to a wrong answer from the user.

In the transcripts there were three interesting issues. The first was about the computer's response to a mistake of the users: should it teach them or not?. The second was about the user's progress through the software, when should users step into a higher level of difficulty? and what kind of feedback should be provided if they did a sequence wrong?. The third was about the software's response if the user starts to play instead of following the instructions. Following is a description of each of these issues.

- **Computer's feedback**

The first issue was about the type of feedback that the user should receive in case of a wrong answer, for example, should the computer give the right answer in such a case. That is, should the computer teach the user about the concept or not.

The Psychologist proposed that the computer should teach the child. The teachers argued against this, and proposed that the computer should provide an example, but

never tell the right answer. The teachers said that the software should let the users know that they did something wrong (by a simple sound or a visual message), and provide guidelines or help to show them the concept, and then the user should do the exercise again. This example or help should be provided by the principal character in the software, the one that guides the user.

The teacher's argument was presumably based on what a teacher does in the classroom, and as is explained in the discussion about the group of categories 'Aim', they argued that the computer should not teach, but it should guide the user, give instructions and help. In this sense, they were using the same arguments that they used to describe their behaviour as teachers applying constructivist methods. They designed the number and type of successive helps and the way in which the computer should guide the user.

This shows how these teachers transferred their teaching methods to the computer while designing the feedback. They were following what they did as teachers in the classroom. The type of arguments were:

TE: Then it [the computer] would be doing what a teacher does when they [the pupils] don't understand (Session 5).

TM: Yes, that's what we are doing as teachers in this moment without the software, (Session 9),

TE: That the computer gives help to him, that the computer guides them, as you do, that is, you don't tell them look, you have to do this (Session 9).

TM: As we do it as teachers, if we give him everything done, then he does not think about (Session 9).

In these examples it is possible to see that they replicated their behaviour as teachers while deciding about the computer's feedback. This type of feedback is similar to the type of feedback described by Mayer (1995) as "feedback in academic learning tasks" that "refers to information concerning the correctness of students' performance" (p. 249).

- **Users' progress**

The design of the user's progress was another issue, that is, given the structure of the software (as a matrix) and that each screen of the software would contain three or four exercises, they had to decide what to do if the users did one or two of three exercises only. Should they browse to a next level of difficulty or stay in the same level but in another skill? The teachers argued that if one of the exercises was wrong, the users

should browse to another area of a similar level of difficulty, without letting them know that they could not advance to a higher level of difficulty. This is, after doing the exercises of one screen, the software should change to the next screen automatically, without asking the users or letting them know of their overall performance.

The argument for this decision was given in Session 13 and was based on the motivation of the user. In the dialogue TE is arguing that children should have positive inputs only, they should live happy, but the Psychologist is concerned about reinforcing bad habits when the users are doing something incorrectly wrong and can continue playing.

TE: But don't forget that children are children OK. They don't think like us that they don't care, sometimes for them the game is [everything]. And specially with this kind of children [of these schools]. There can't be more negative things, like the ones you saw at the school's door <two children fighting>.

PY: Sure, but we could fall into the dilemma of reinforcing [bad habits].

TE: But the computer should be something beautiful. If in the classroom one is always telling them look you went out of the margin, you draw it wrong, you see?, that's one of the nice things about computers.

PY: What I say is that we will be reinforcing a behaviour. That is, a wrong learning, because if the child does this wrong, I don't know, he keeps playing.

TE: But it doesn't matter, the game is not a competition for us.

TE's argument about avoiding negative feedback, even if they did some exercises wrong, is consistent with the affective outcomes described by Cooper & McIntyre (1995) that teachers look for and is not consistent with the descriptions given by Hammersley (1990), of the way in which a teacher would react in case of wrong answer in a classroom. This may well relate to the specifics of this population of students where the affective dimension is very important.

This decision is interesting because, on the one hand they designed the feedback for the actions in each screen (for each exercise) so that it would provide the required information for the users (first a message indicating that they did something incorrect and then the help) but on the other hand, they designed the browsing through the different screens of the software it such a way that it would not let the users realise that they had made a mistake.

In this design there are two levels of feedback, the first is related to the particular interaction during one exercise, and the other is related to the overall browsing

design. In the former they design a way in which the user should be able to learn how to do each exercise. In the latter, the idea was that the computer would provide a nice experience to the users, without disappointing them.

Underlying this issue, it appeared in the data that teachers assumed that the user would be able to do all the exercises and that these were simple enough to be completed. That is, they had high expectations and confidence in pupils' abilities to meet them, as reported by Woods & Jeffrey (1996) this is usual in the classroom atmosphere.

- **Users' general behaviour**

The third issue was the discussion about the behaviour of the users in front of the screen, what happens if they started to play instead of doing the exercises ? or if the users did not understand the instructions or if they can not click in the given area of the screen. They decided that there should be written instructions on the screen and that there should be the possibility of repeating the instruction (by clicking somewhere). They did not address this issue further, and they assumed that the teacher would have a role in this in so far as the computer would never 'notice' this behaviour. That is, they transferred this decision to the 'classroom management' arena and they decided that the teacher would have this role.

7.8. CONCLUSIONS

This analysis showed two main dimensions of what these teachers believed about educational software and computers. First it showed what they believed the characteristics a piece of software should incorporate; and second, what they believed about their roles and actions while using computers and educational software in the classroom.

The discussion of these findings will be presented in the Discussion and Implications chapter of this report. The conclusions that are included here will serve as an introduction to the discussion of this research that will incorporate the results of the other analyses as well.

- **Subject areas and interface elements**

It was shown that these teachers decided that the subject area of the software should be about 'basic skills' and that they embedded this subject into a playful story that children would read while doing exercises. This design decision can be understood

looking at reported experiences about what teachers do in the classroom, for example, Woods & Jeffrey (1996) say that teachers use stories as a way to 'bring home' to the child the contents of the curriculum, and establish 'common knowledge' (referencing Edwards & Mercer, 1987). The concrete subjects (contents) of the software were drawn from an external source (a textbook) that contained exercises for 'basic skills' development. These exercises were adapted by the Psychologist in order to be included in the software.

In order to implement the story they designed the human-computer interface elements of the software as they were designing the elements of the story rather than 'technical' human-computer-interface elements (like buttons, dialogue boxes, menu bars). That is, they designed backgrounds, characters and several elements that were part of the scenarios in which the story would happen.

One particular element that they designed was the principal character of the story, which would guide and tell the user what to do. In this sense, they implemented the atmosphere of the story that would be 'told' during the lesson (which is similar to the atmosphere in a normal classroom described by Woods & Jeffrey 1996, and reported in section 2.6.2 of this thesis). This atmosphere would be 'orchestrated' by the principal character who appears to assume the role of 'story teller'. While designing these elements, they stressed the need for representing their environment through the software, that is, the interface elements should behave like 'real' elements and should be similar to the ones found in their neighbourhood, trying to place instruction within 'authentic' context that mirror real-life problem-solving situations (as Brown, Collins, & Duguid 1989, argue that a situated approach to teaching and learning should consider).

In the next chapter, it will be shown that the design of the subject areas and the human-computer interface elements of the software serve the same purpose, this is, to provide the classroom atmosphere for the lesson.

- **Content organisation, browsing and teaching strategy**

These teachers organised the content of the subject area in the story dividing it into levels of difficulty and type of basic skill to be developed and they designed the browsing possibilities of the software so that pupils could follow a sequence of contents accordingly to their performance. In other words, they were designing the way in which pupils would follow the lesson. In particular, the way they would progress through the contents and the feedback that they would require in order to be able to finish one story. While designing these elements, the teachers drew on their

experience in the classroom to define and justify their decisions. In the arguments they gave, they assumed that the computer would take their role, in that what they did in the classroom as teachers, served as a 'model' of the way the computer should behave (for example: to present the contents organised as a matrix, based on levels of difficulty; to let the users progress based on their achievement; to provide guidelines as feedback for wrong answers; to ensure positive feedback during the lesson).

These discussions were presented across these three categories and, if combined, they show one particular dimension of the design of this piece of software which is related to the pedagogy embedded in the software. This pedagogy could be understood by thinking that these teachers were designing the contents structure and availability not only to ensure that pupils are taught something 'new', that they 'keep moving', but also that they have the resources to understand what the software is presenting (similar to the descriptions found in Hammersley (1990) about what teachers do during a lesson). Further on, it could be thought that they tried to implement a style of teaching that was defined as 'Interactive' by Cooper & McIntyre (1995) where the computer and pupil 'negotiate' the learning aims.

- **Interaction and pupils' actions**

The interaction with the software was based on exercises that pupils would do while browsing through the story. This interaction was designed as sequences of questions (exercises prompted by the software), answers of the user (actions) and feedback that the computer would give to the user. On the other hand, they described the pupils' actions as rehearsing while answering the questions prompted by the software, without explicitly designing other type of actions. This type of dialogue is usually recognised as a teacher-student dialogue (for example, by Hammersley, 1990) and the particular decisions that these teachers made about the way the software should interact with the pupils follows some of the principles that make up what can be described as 'progressive teaching' (Edwards & Mercer, 1987).

- **Actions and aims**

The development team talked about two types of teacher actions: first, helping the pupils to use the software, solving technical problems and second, keeping the discipline and management of the classroom. These roles are found in other reports (for example: in Olson, 1988 and Sandholtz, et al., 1997) and in this sense could be expected. But what is not commonly found is that they rejected the idea of designing particular interactions of the teacher with the software arguing that the children should be able to learn by themselves and that the teacher should guide them in this

process. Further on, they argued that in the computer lab the teachers would act as 'guides', whereas in the classroom they would teach the subjects. They did not consider scaffolding, tutoring or coaching to be part of their role in the computer lab. In this sense they differentiated their role as teachers in the computer lab from their teaching role in the classroom.

These findings are supported by the discussions related to the aims of the software where they indicated that the main aim was to save time as a result of its multimedia capacity. This is, they said that they could go through the contents much faster because the computer was used as a rehearsal tool that could serve as a complement to what is taught in the classroom. This is, the aim of the computer for the teacher was a teaching resource that was used by the pupils as a rehearsal tool, not as a source for new learning or as a cognitive tool[21].

Consequently with the descriptions found in the reports by Lepper & Malone (1987) and Schofield (1995), these teachers said that the aim of the software for the pupils was as a game, that while engaging in a playful interaction children would not realise that they were 'learning' or rehearsing. They recognised the motivational power of the computer and they decided to use it as a resource that engages children in learning and becomes part of the punishment -reward system of the classroom. This is, they were using the computer as a control tool, similar to the descriptions found in Olson (1988).

[21] Cognitive tools are defined by Lajoie, (1993) as software that enhance metacognitive strategies, share cognitive loads, expand the cognitive possibilities of the learner and allow them to generate and test hypothesis. The software this Development Team developed was not of this kind, it did not require the same degree of complexity, for its aim was simply to reinforce what the teacher had already taught in the classroom.

VIII. DISCUSSION AND IMPLICATIONS

8.1. INTRODUCTION

This section integrates the findings from the three analyses reported in chapters V, VI and VII; and discusses them using different theoretical frameworks. The overall aim is to try to characterise the model of understanding of information technology that these teachers demonstrated during the 19 sessions of the software design process.

This chapter concentrates on those aspects of the previous analyses which throw light upon the model, and a number of other interesting insights are not included.. For example, some previous chapters have shown some interesting data about the development process, the breakdowns between the teachers and the Software Engineer and the Psychologist and the particular teaching style of these teachers, these issues were used to focus the discussion on relevant issues, but are not taken up here as issues in themselves. The rationale behind this decision is that both the process whereby the different characteristics of the software were defined and the identification of the particular teaching style of these teachers, helped to identify the relevant dimensions of the model, showing what these teachers thought about computers and educational software and how they embedded their teaching practice into the software.

The discussion starts by presenting a theoretical perspective that examines two dimensions of what the two teachers involved in this study believed (section 8.2). Firstly, what they believed about the use of computers for teaching, and secondly, what they believed about the characteristics that a piece of educational software should have.

The former dimension shows that these teachers conceived of the computer as a resource that could replace them in specific roles and in particular that it could be used by the pupils as a rehearsal tool. The latter dimension presents the teachers' model of educational software and shows the particular elements of their teaching strategies that were embedded in the characteristics of the software. These two dimensions together constitute what is called here a model of understanding of information technology in education.

In order to provide a critical view of the way in which this model was elicited, the methodology used during the research process is also discussed (section 8.3). This section starts by analysing the definition of the present research as a case study and then, it examines the way in which the data were structured, that is, the systemic

network used. Finally, the analyses of the data are discussed, focusing on the combination of qualitative and quantitative methods. The implications of this discussion are related to the process of construction of the systemic network and to the value added to the overall analysis by the combination of quantitative and qualitative methods.

These three sections of the discussion converge in the last section of this chapter, which corresponds to the implications of this study (section 8.4). As mentioned before, the main implications are represented through the model of understanding of information technology in education that considers firstly a strategy to use computers as an aid for teaching and secondly a framework to design educational software that incorporates teaching strategies into particular characteristics of the software.

Before this discussion starts and as a general framework for the scope of the analysis it is important to remember two things. Firstly, these teachers decided to design a piece of software for children of age 4 to 6. Clearly one of the chief interest of these teachers was the use of computers with young children. The data collected across 19 meetings in which the teachers were designing a piece of educational software does give us a wider view of the teachers views, since they had to call on their general perspectives on information technology in education in order to enter into the design process.

The second consideration is that these beliefs are certainly influenced by their working context and previous experience. These teachers had three years of experience in the use of computers and they were teaching poor kids in a rather average city school with a few but not many computers.

8.2. THEORETICAL PERSPECTIVES

This section discusses the beliefs of the teachers about the use of computers for teaching and the characteristics that a piece of educational software should have.

8.2.1 The use of computers for teaching

This section discusses the findings related to these teachers' beliefs about the use of computers and software in teaching. The discussion is organised around three questions:

- What did these teachers declare as the aims of the computer or software? Why ?
- What was the role of the teacher with respect to the computer in the software design? Why?
- What activities would the pupil do with the computer in the software design? Why?

As it will be shown, the teachers defined the aims of the computer as a tool for pupils' rehearsal and as a control tool for classroom management. These aims were then implemented through the actions designed for the teacher and the pupils. They decided that, in the computer lab, teachers would act as computer helpers and classroom managers, without interacting with the computer. The pupils actions were designed as interacting with the computer only. The following sections describe these roles and discuss the rationales that could explain these decisions.

The transcripts that will be included here are used to clarify the arguments, and were already presented in the 'Contents Analysis' (Chapter VII) and correspond mainly to the sections 7.4 ('Actions') and 7.5 ('Aim').

8.2.1.1 Aims of the computer or software

During the design process the development team spoke very little about aims, in fact only 4% of the data were classified under this group of categories. In the 'Participation Analysis' it was shown that the contributions and profiles of participation of the teachers indicated that they were using their professional background during the design of these categories. In the 'Sequences Analysis', the group of categories 'Aim' were mostly related to data about the pupils as users (i.e. the group of categories 'User'). And in the 'Contents Analysis', it was found that in this group of categories teachers defined the aim of the software as a rehearsal of what they do in the classroom, not as a source for new learning. A subsidiary aim found was to manage discipline problems of the pupils.

- **Means to an end: Computer as a tool for pupils' rehearsal**

The conception of the computer as a rehearsal tool has been largely described in the literature, in fact, considering that drill and practice software are the most used type of software (Cuban, 1997; Evans-Andris, 1995), it could be argued that rehearsal is the most common aim that teachers have in mind when deciding to use computers and the fact that they use drill and practice software is because it can be easily accommodated to such aim.

Considering this argument about the teachers' aim to use the computer as a rehearsal tool and recalling some of the teachers' statements:

TE: It is supposed that all what comes out of the software, the teacher will go through it in the classroom. What happens is that perhaps now, instead of being ten days covering the subject, [she] will do one session [in the classroom] and will bring them here [to the lab]. She will put the software in the equipment and its over, you see. It will save a lot of time and effort to her.

...

TM: The computer in it self is a stimulus, not perhaps so much value in the contents or the way to expose the content. Because, in this sense, one uses it more as a complement of [the classroom activities], in the case of delivering subjects. In this case it will be the same.

It is possible to argue that, in fact, these teachers explicitly decided to use it as an aid for efficiency, for performing their teaching better.

Similar aims for computer use were classified by Sandholtz, et al. (1997) as belonging to an adaptation level of instructional evolution. They report that maths teachers, for example, relied on computer homework for arithmetic and spent school time on problem solving. Also, they report one of their teachers saying: "... we've been using the software as a backup for each of the objectives" (p. 70). Further on, they report that teachers "developed strategies for increasing the amount of material they could cover during the school day" (p. 70).

Based on this evidence, the fact that this particular piece of software could appear to be just drill and practice software does not reflect their real aims, which appear to be to use the computer to support their teaching strategies and thereby to be more efficient. It implies that these teachers identified a certain need of the user (pupil) to rehearse the contents taught in the classroom and they decided that the software could satisfy this need. On the other hand, they realised that they needed 'more time to teach' and decided that this type of software could satisfy this need, providing them with additional time to teach. In both cases, they considered it as a means to an end.

The question that remains is what sort of rehearsal tool did these teachers have in mind while designing the software. To answer this question the particular characteristics of the software are discussed in section 8.2.2. Nevertheless, the interesting fact is that the 'Sequences Analysis' showed that these teachers related the group of categories 'Aim' with the groups of categories 'Content Organisation' and

'Actions'[22] only two times during the development process and no further relations were found. This shows that when they referred to the aims of the computer, they spoke at an abstract layer, without explicitly transferring their arguments to the design of the software's characteristics nor to the actions that would be performed in the classroom.

This finding, together with the results shown in the 'Participation Analysis' that the teachers did use their professional background while speaking about aims, draws an interesting picture of these teachers beliefs. First, it indicates that they conceived of the computer as a professional resource (in so far they used their professional background to define its aims) and secondly, that this consideration was done at an 'abstract' layer, without explicit connections to the 'real' classroom practice. This picture is coherent with the findings of Alexanderson (1994) and Marton (1994), that show that teachers are not necessarily aware of the strategies and tactics that they use while teaching, so it is reasonable to think that they do not transfer their abstract aims (or strategic planning) to the classroom practice and therefore, accordingly to Yinger & Hendriks-Lee (1995), both, planning and classroom interaction are responsive, compositional and situated (i.e. contextualised). This picture could explain why the design of educational software has failed to incorporate the teaching dimension, in so as far teachers would not have explicit awareness of these issues while participating in a software design process.

Another finding is that, considering the overall picture, it can be argued that they did not consider that the computer could have different aims depending on the software that is being used or the context in which it is used. For example, it could be a communication device (e-mail), a physics lab (simulation software) or a word processor. This finding has its own implications, in so far it reduces the scope of possible aims of the computer and confines it to particular roles during teaching, in this case, rehearsal. In this sense, what is needed is to expand the teachers' conceptions of computers, including the different uses of the machine that are supported by different types of software.

- **Means to an end: Computer as a control tool for classroom management**

The use of the computer as a control tool for classroom management has been reported in the literature before. For example Olson (1988), describing one of his teacher's activities, says:

[22] The actions that were found in these sequences were about the teacher interacting with the pupils and the aims were about the software for the pupil.

> He adopted a rota system, in which access to the computer was part of the reward and punishment structure of his classroom
>
> (Olson, 1988, p. 29)

And, as Olson (1988) says, one reason that could explain this use of computers is that:

> The teachers were interested in the motivating power of computers: 'they excite students, just the fact they're in the school'
>
> (Olson, 1988, p. 109)

Similar descriptions were also reported as a strategy that teachers use at the adoption level of instructional evolution (Sandholtz, et al., 1997) and have been reported in other studies as well (for example, by Schofield, 1995). The reason for designing the computer's aim for the teachers as a control tool, has been related to the high degree of motivation that pupils have to use it (Lepper & Malone, 1987; Sandholtz, et al., 1997), and so teachers use this characteristic as an aid to maintain discipline in the classroom.

What is particular to these teachers is the fact that they recognised this characteristic of the computer and they explicitly decided to use it for their advantage. For example, Teacher E says that:

TE: ... Because the teacher is a teacher, you see? But the computer is not, if you don't respect what it is telling there [the computer], you missed the opportunity of working with the computer. You cannot work the game. It is part of the game. You loose the most loved, which is to work with the equipment. That is, the computer its true, it can not replace the teacher, but in this minute it is very motivating for the children, then it is like the engagement tool of this.

This type of awareness was considered by Sandholtz, et al. (1997) as a characteristic of teachers that are at the adaptation level of instructional evolution, they write:

> Teachers now were able to use technology to enhance student motivation and interest while decreasing the number of discipline problems.
>
> (Sandholtz, et al., 1997, p. 71)

In this sense, these teachers saw a classroom management tool in the computer and therefore considered it as a control tool. In this case the computer is used as an instrument for this purpose, as opposed to considering the computer as a source of new classroom management requirements (for example, technical problems).

This fact in itself is not new, it has been reported previously, but teachers' awareness of this characteristic and their consideration of it during the software design process enables us to argue the need for considering this issue while speaking about classroom management during teacher training. This is a neglected area, Sandholtz, et al. (1997) point out that, in the descriptions about classroom management found in the 'Handbook of Research on Teacher Education' (Jones, 1996) there is no reference to computers, and that in the report on information technology in teacher education (Willis & Mehlinger, 1996) or concepts of educational technology (Eraut, 1996), there is no reference to classroom management issues.

- **Means to an end: Computer as an instrumental or expressive tool**

Olson (1988) identified additional aims of computers for teachers, he describes the computer as an instrumental and as an expressive tool for teachers. In the former he identifies two groups of uses, first, the ones where the computer was used as an instrument to teach the subject. And second, the ones where the computer was used to teach computer literacy together with the particular subject. In this case it was shown that the computer was used in several instrumental dimensions that correspond mainly to the first group identified by Olson (control tool, efficiency tool), but no data were found that could support the claim that these teachers intended to use it to teach computer literacy (second group).

Two reasons for this could be drawn, first that because of the technological evolution (easy-to-use software) and the availability of computer-like games at home, the focus of learning to use computers (or similar machines) is no longer considered to be an issue for these teachers. They assume that children will know how to use them or will learn quickly. Second, the software was designed for early stage users and therefore the need to learn to use particular pieces of software based on vocational arguments (Squires, 1996) was not relevant at this stage.

About the latter aim, the computer being an expressive tool, Olson (1988) argues that by using computers, teachers express something about how they want to be seen as teachers. As he says:

> Mr Heiburg [one of his teachers] is not only interested in having a computer in the classroom as an instrument for promoting computer literacy ... but he is also using the computer as a way of expressing something about himself as a teacher - about his interest in being modern and up to date.
>
> (Olson 1988, p.15-16, our brackets)

And later on he says that:

> He [the teacher] constructed the using of a computer as a way of being 'modern', of expressing something about the kind of teacher he is.
> (Olson 1988, p. 28, our brackets)

Olson's description of the computer as an expressive tool can also be understood in terms of thinking about the computer as a catalyst for innovation (Hawkridge, et al., 1990; McDonald & Ingvarson, 1997), in so far as Olson's teachers, who had accepted the use computers in their teaching, were engaging in a process of innovation where they wanted to experiment with new teaching methods and the computer provided them with tools to implement a particular innovation (Olson, 1988, describes how in his studies the goal of the innovation itself changed in the implementation process). In this sense it is possible to argue that for them the innovation itself was used as an occasion to project themselves as being avant garde, and the computer was a catalyst to do so.

From this latter perspective, the two teachers participating in this research were also engaged in a process of innovation, in so far as they were designing a piece of software to help in their actual teaching practices. In this sense, they were trying to use the computer to do their teaching better and thereby to support their innovation. Within this perspective, the resulting design was not a catalyst for change, but a support for the ongoing changes prompted by their needs. It is then possible to argue that the catalyst for innovation in this case was the research process itself.

Although it is possible to find this dimension of expressiveness in this research, no evidence was found that these aims where explicit in their design, so it can not be argued that these are means to any explicit end.

8.2.1.2 Teachers actions with the computer

In general terms, the teachers spoke very little about teachers' actions during the design process, in fact, as presented in the 'Participation Analysis', only 2% of the units of analysis were classified under this category and 49% of these units where spoken by the teachers (i.e. 1% of the total). In the 'Sequence Analysis' it was shown that they did not relate the design of actions for the teachers with the design of the interaction with the computer. Rather, these teachers assumed that only the pupils would be interacting with the machine. In doing so, they dissociated their role as teachers from the use of the computers.

While speaking about their actions, these teachers designed two types of actions in the computer lab: helping the students with the software and ensuring that the students would follow the instructions given by the teacher (for example, that the groups were rotating). They also rejected their role as 'subject teachers' in the computer lab.

- **Computer helpers and classroom managers**

An example of 'helping the students with the software' would be:

TM: The role of the teacher with the software is nothing else than being there. Helping [the pupils] to get in or out of the software, because the rest of the work she will do it in the classroom, and I think that's the idea and that's the way it should be, because, we cannot think that it effectively will help us to cover subjects, because that's impossible.

It seems that Teacher M is expressing two issues, one is that in the computer lab the teacher needs to worry about the use of the software, and the second is that her role as subject teacher is in the classroom.

The other role, managing the classroom, is shown here:

TE: Yes, because the ideal is that the teacher here does like other things. That is, being the guide means that she has to worry about, to start with, that it is done what should be done, [this is,] using the software; that children are rotating; that many other things are happening; that it is happening the fact that, that they have equal chance, all the children, that is, he [the teacher,] is doing a lot of other things. That is, at no time can he go home, because it is likely that the computers will burn, or that someone goes for water and pours it over the computers or turns them over the desk. That is, teachers cannot be absent,...

She is expressing the need for keeping control of the computer lab, that is, she needs to be worrying about the pupils rotating and doing their work and not 'destroying' the hardware. She is worried about classroom management issues that are different from the traditional management requirements described by Jones (1996).

Similar teachers' roles, as classroom managers and helpers in the use of computers, have been described in the literature. For example, Olson (1988), describes what one teacher was doing, by saying:

> Seat work does take some of the pressure off her as it engages attention, leaving her free to deal with computer problems.
>
> (Olson 1988, p. 104)

In a similar vein, Sandholtz, et al. (1997) describe these type of classroom management issues as happening at the entry level of adoption of the technology. They say that at this stage teacher's concerns are related to students misbehaviour and attitudes, physical environment, technical problems (hardware and software) and dynamics of the classroom environment.

These findings, as well as other reports, highlight the need for considering computers as sources of new demands of management in the classroom (for example, technical problems, source of misbehaviour, etc.). Therefore, there is a need for considering classroom management issues while designing computer based teaching strategies (Sandholtz, et al., 1997) and that is precisely what these teachers did.

- **Teachers interacting with the computer**

There are few reports where teachers manipulate the computer while teaching (one example is Fraser, et al., 1991) and where teachers indirectly interact with the computer while developing an activity in the classroom (for example: Dockterman, 1991). Generally, teachers' actions are described as guiding the pupils while using a piece of software (as it is described by Olson, 1988 and Sandholtz, et al., 1997). In this latter type of activity three different teachers' actions are usually described: helping the pupils to solve computer problems, helping the students to solve subject matter problems and dealing with discipline problems. Categorising these actions described in the literature it is possible to say that while using computers in a classroom a teacher will be doing some of the following actions:

a) Directly interacting with the computer
b) Indirectly interacting with the computer
c) Solving computer problems
d) Helping the pupils to solve subject matter problems
e) Controlling the classroom discipline

As presented in the 'Contents Analysis' chapter (section 7.4.1) while describing their actions, these teachers explicitly rejected the idea of direct interaction with the computer and focused on actions (c) and (e). That is, they did not consider that they should get involved with the contents in any particular way.

In fact, the 'Sequences Analysis' showed that the design of the teachers' actions were not related to the characteristics of the software nor to the teaching strategies. Further, these teachers rejected the idea of designing other actions for the teacher with the

software. They discussed this issue with the Psychologist and the Software Engineer and decided to release the teacher from the interaction with the software. These teachers said: "we want to be displaced by the computer", they did not want to be required as teachers. They supported this claim arguing about the new pedagogic tendencies, saying:

TM: ... we think that now that learning by doing is very important, and that the child is constructor of his own learning and we are guides in this.

Here it is possible to see that, in this context, they saw themselves as 'guides' while they saw the pupils as 'constructing' their own knowledge. Examining their conception of 'guide' it can be shown that they define guide as 'classroom manager', that is, to be worried about the discipline and lesson flow. Teacher E defines her 'guiding' role as:

TE: Being the guide means that she has to worry about, to start with, that it is done what should be done, [this is,] using the software; that children are rotating; that many other things are happening; that it is happening the fact that, that they have equal chance, all the children, that is, he [the teacher,] is doing a lot of other things.

This is an important consideration while trying to understand these teachers' intentions, because it shows what they understood as 'progressive' teaching methods (Edwards & Mercer, 1987). This is, they believed that their role in the computer lab, while using 'constructivist' methods, was to manage the classroom and they did not see their actions as scaffolding, counselling or tutoring the pupils.

This picture of teachers' beliefs could be a valuable source for further research in this area, in so far as it shows a possible misconception about their roles while using constructivist methods. This conception could mislead them to adopt rather passive roles while implementing these new teaching methods and therefore undermine their eventual effectiveness. The consequences of such a conception could also undermine the possibilities of success of innovation programs because teaching would be understood as classroom management only.

Considering that these teachers believed that pupils would be constructing their knowledge while interacting with the computer without requiring any teachers' mediation, and that they defined their role as to control the physical and technical environment (helping with the discipline and computer problems), two questions about the real implementation of such concepts arise:

The first question is whether their understanding of constructivism leads to such beliefs or their beliefs lead them to understand constructivism in this way. In other words, are their actions a consequence of the new learning theories or a response to these new theories?. Some reports indicate that what teachers normally do are both things, that is they accommodate their old practices to the new tendencies and also change (when possible) some practices to follow these new theories (Edwards & Mercer, 1987). Nevertheless, it must be remembered that this finding relates to what these teachers said they would do and not to what they really would do in the classroom or computer lab.

The second question is would they really only act as classroom managers or is it the case that they simply did not express all the actions that they might take part in during the lesson. This question can be examined based on some research about teachers' practice, for example, in their report, Yinger & Hendriks-Lee (1995), say that planning and classroom interaction are responsive, compositional and situated (i.e. contextualised). In this case, these teachers were, in some way, planning a lesson using the computer and designing software that could be used during such a lesson. So, it is possible to say that these teachers, while using the software, would in fact do other actions that depend on the context. Nevertheless, based on these findings, it can only be argued that at a level of beliefs, their main concern was acting as guides and managing the computer lab.

If we combine the finding that these teachers rejected the role of manipulating the computer with the findings of the 'Participation Analysis', where it was shown that these teachers expressed aims for the computer but did not design actions for the teacher with the computer, then we can see something about their beliefs about computers. This is, they considered the computer to be a resource, like a video that pupils 'watch' (or use) during the lesson and they did not consider it to be a tool like a blackboard that is used by the teacher. In this case the resource is interactive, in so far as computers are regarded as interactive tools par excellence, but they believed that this interaction should be exercised by the pupils only. This interaction was carefully designed as to be a model of what teachers do in the classroom. For example, as shown in Section 7.7, the teachers designed the feedback based on their teaching practice, transferring their behaviour to the software. This places the use of computers (from the teacher's point of view) close to the use of other media (like video or television) and thereby it defines the computer as a resource and not as a tool for the teacher.

Nevertheless, this resource was considered to be very special and, even, close to being a 'real person', a 'colleague'. This would justify their trust in the computer taking over some of their teaching roles (but not all of them!).

One possible explanation for this consideration could be that individuals' interactions with computers are fundamentally social and natural, just like interactions in real life (Reeves & Nass, 1996). Considering this idea, it is possible to argue that these teachers treated computers as they treat people, in particular, as colleagues. This argument is consistent with the overall picture presented and could throw light on the reasons why these teachers believed that the computer could take over some of their roles as it were a colleague.

Another possible reason for this belief is mentioned by Olson (1988), he stresses the need of the teacher to be in control of the classroom and in this sense to be able to master the technology. The fact that these teachers rejected the possibility of manipulating the computer could be a consequence of their fear of not being able to be in control of the situation due to possible hardware or software problems and this would be added to the normal classroom requirements.

- **Classroom and computer lab**

The fact that these teachers argued that the subject contents should be taught in the classroom and not in the computer lab differs from other reports, where the teacher uses the computer while teaching the subject. For example, Fraser, et al., (1991) describe the different roles that the teacher, students and the computer take up while teaching with technology. The difference with this case is that in Fraser's report the computer was brought into the classroom and it was operated by the teacher, who was orchestrating the activity.

Also, in Olson (1988)'s report, some teachers taught the subject in the computer lab, adopting the role of tutors. Quoting one of his teachers, he writes:

> I like the idea of being able to sit down with a small group of students who are working on a particular task while others are at the computer, or even being at the computer and helping them individually. You are acting as a tutor rather than a teacher at the front of the class. I don't want them to see me as the ogre at the front of the class... I can offer them more of me than in a traditional role.
>
> (Olson 1988, p. 47)

Considering these examples and other reports (for example: Sandholtz, et al., 1997; Schofield, 1995 and Watson, 1990), in both scenarios (the computer in the classroom

and the computer lab) there is evidence that teachers adopt different teaching strategies while using computers to teach and that in both scenarios they do teach the subjects. So, what the teachers in this case study designed corresponds to an extreme position of considering the computer lab as a rehearsal room.

This distinction leads us to go one step further, differentiating the classroom and the computer room as places with two different teaching and learning aims. For these teachers, the aims in the classroom were that pupils learn new concepts and that they teach these concepts. The aims in the computer lab were that pupils rehearse these concepts and that teachers act as classroom managers only. These conceptions will be analysed next.

- **Means to an end: Class room and rehearsal room**

This conception of the computer lab as a rehearsal room could explain the general tendency described in the literature that software products that are most frequently used in school are based on drill and practice activities (Cuban, 1997; Evans-Andris, 1995). In this sense, these teachers divided their tasks as teachers in two stages, first teaching the subject and then rehearse the subjects that were taught. The first stage of this strategy would be carried out by the teacher in the classroom and the second stage would be carried out by the computer in the computer lab. In both stages classroom management would be carried out by the teacher.

This description of their teaching strategy reveals that these teachers were not simply transferring their instructional role to the computer nor adapting it to their established curricular and pedagogical preferences as they might be at the adoption level of instructional evolution (Sandholtz, et al., 1997). Rather, they designed the software to use it effortlessly as a tool to accomplish real work as described by Sandholtz, et al., (1997) to happen at the appropriation level of instructional evolution. In fact, they separated their roles as subject teachers, as rehearsal teachers and as classroom managers and then designed the software so that the computer should take the role of rehearsal teacher, while they would keep the roles of teaching the subjects and controlling the classroom.

The teaching strategy that these teachers decided to use is very interesting because they separated their teaching methods into two stages, one that could be interpreted as being more constructivist and which would be used by them in the classroom and the other, which could be interpreted as being more instructional (rehearsal), and would be taken over by the computer in the lab. In this sense, it could be argued that these teachers did what some other professionals do, that is, delegate the more labour

intensive work to the computer (for example, doing calculations). The difference in this case is that this more labour intensive activity not only involves the computer, but also the pupils. This key difference lead us to question the necessity of having a 'rehearsal room' where pupils carry on this more instructional activity.

Another question that this type of design raises is whether software designers should acknowledge this role of the computer and use it as an advantage to design software that could then be used as rehearsal, or should they try to place the computer into the classroom and use it as a Trojan horse as defined by Olson (1988):

> Good software is a Trojan horse- an appealing new package inside of which is the germ and challenge of innovative practice.
>
> (Olson 1988, p. 115)

Answers to these questions are not clear yet, so far research reviews do not support the proposition that any one particular teaching style is more effective than another (Sammons, Hillman, & Mortimore, 1995). This implies that if software is used as a Trojan horse to change 'the teaching methods' in general, it is not possible to be sure that the new style will be better than the old one. These findings show a more focused possibility, that is, if software is designed to be used as a well conceived rehearsal tool, that could help teachers to perform some of their specific teaching routines. In this sense, from an innovation perspective, such a tool would be 'grafting the new on the old' (Huberman, 1992), in so far it would be designed for similar purposes (rehearsal) but using different methods.

Leaving such hypothesis to be investigated in further research, what can be observed here is the fact that these teachers defined a particular role for the computer and that they think that the computer could help them in this role. Then, it could be argued that during the design process, these teachers were trying to answer the following question:

> Given that I am expected to maintain order and get students to learn essential skills, knowledge, and values, how will these machines help or hinder my mission?
>
> (Cuban, 1997, p. xii)

And they decided that the computer could help them in the rehearsal role during their teaching. In this sense, these teachers tried to give one answer to this particular question.

8.2.1.3 Pupils activities in the software

Pupils' actions were designed in this case as respondents to the software's questions, similar to some pupils' actions described as happening in a classroom without computers (for example: Hammersley, 1990; Woods & Jeffrey, 1996). The 'Contents Analysis' showed that the main type of action designed for the pupil was to answer the questions of the software and, further, the 'Sequences Analysis' showed that during the design process the pupils' actions category was frequently related with the software interaction categories.

In order to analyse this design, the software interaction and the classroom activities can be separated. For example, looking at some reports found in the literature about alternative actions of the pupils or teachers while using computers in the classroom, it is possible to find that activities are described in these ways:

- where the computer is used to organise and guide the activity but pupils are not necessarily interacting with it (Dockterman, 1991)
- where the teacher uses the computer while pupils watch the software and participate in an activity taking different roles (Fraser, et al., 1991),
- where the pupils take part in a project in which the computer is used as a tool to analyse information and produce reports (Sandholtz, et al., 1997),
- where the computer is used as a source of information that can be analysed (Olson, 1988),
- where the computer is used by the pupils as a tool to construct ideas (Laborde, 1995).

These examples refer to descriptions of classroom activities rather than to particular types of software interaction. The activities in the classroom are not necessarily conditioned by the particular design of the software, and the same piece of software could be used to implement different types of classroom activities.

Two different issues can also be identified in this research and need to be considered. First, the fact that these teachers did not design any other type of interaction with the software, and second, the type of activities designed for the lesson in which they would use the software.

Related to the former, it appears clear that they wanted the pupils to spend their time responding to the software's prompting. Therefore it could be argued that they wanted the pupil to interact with a drill-and-practice like software. Looking at this issue from a broader perspective, the rationale for this design can be understood as a

consequence of the their conception of the use of the computer as a rehearsal tool for the pupils that was discussed in the previous point.

Related to the latter, it is possible that they were designing a piece of software only and they did not mention the other classroom activities that would happen while using the software. So, based on this evidence, it is not possible to say that these are the only activities that pupils would be carrying on during the lesson. On the contrary, in some dialogues they refer to group based activities (reported in Section 7.4.8), pupils' rotations and other activities that could be used to argue that pupils would be doing other things as well. Nevertheless, these activities were not described during the design process.

It is possible to say that what these teachers were doing was to design a tool for pupil's rehearsal only, and therefore, pupils actions described in this study allow us to draw the false image of 40 pupils watching the computer screen and responding to the software. This was not actually possible anyway because the computer lab of the school where these teachers worked at the time of the study had only 7 computers for 40 children that would take part in a lesson.

Analysing the overall design of pupils' actions, it was possible to see that both teachers wanted the pupils to construct their knowledge, they said "we think that now learning by doing is very important, and that the child is constructor of his own learning". In this sense it is possible to argue that they believed that knowledge is "something that children must construct for themselves and less as something that can be transferred intact" (Sandholtz, et al. 1997, p.47) describes this perspective as corresponding to the invention stage of instructional evolution in the use of computers in education.

The apparent contradiction here is the way in which these teachers wanted to implement their beliefs, which is closer to instructional methods. But, as it was mentioned previously, from these teachers' point of view, this implementation is coherent with their operational definition of constructivism. This highlights the need for considering that there are at least two different perspectives of analysis: what these teachers believed including their interpretation of certain theoretical propositions, and the theoretical propositions themselves, which could have different operational definitions.

- **Summary of the discussion**

In summary, the evidence suggests that the teachers held the following beliefs:

- That the classroom and computer lab have different teaching aims. The former is to teach new concepts and the latter to rehearse these concepts.
- That computers would help them to perform particular tasks of their overall teaching strategy (in this case rehearsal) and therefore computers are considered to be an aid for efficiency.
- That computers bring new challenges to the normal classroom management requirements, and they can be used to control the discipline (it enters the reward system of the class).
- That while using computers they need to concentrate on helping students to use the software and controlling the discipline, and they do not interact with the software. This is, they perform a role of classroom mangers only.
- That pupil's would act as respondents of the software's questions, but this does not imply that they could do other activities during the lesson.
- That innovation can be supported using computers as resources for teaching which are tools for the pupils.

Some of these beliefs were previously reported in the literature, but the fact that they are presented together as a model of what these teachers believed about the use of computers enables to have a complete picture of the reasons why they constructed such use of the computer. It is not argued here that this is the 'right' way to use computers to teach, but the interesting issue is that the computer is being used as a professional tool, perhaps poorly used, but the challenge of improving such use is for both, the educational software designers and the teachers.

These findings constitute a rich source of information for the areas of software design and software evaluation. For the former area, it enables the designer to position the computer in a certain teaching strategy that teachers are likely to implement and therefore use the software.

For the latter it throws some light on what teachers believe about the software that they will use and enables us to understand the perspective of analysis that teachers would be using while evaluating or selecting a piece of software.

A comparison with the stages of instructional evolution in technology rich classrooms presented by Sandholtz, et al., (1997) shows that these beliefs have correspondences with beliefs at each of the four levels of entry, adoption, adaptation and invention.

This configuration of their beliefs does not correspond to observed uses of the technology in the classroom. So, it is not argued that these teachers would use the computer in the same way as they believed it could be used. But, it can be argued that these beliefs can be used to understand the underlying reasons why a teacher would perform particular actions in the classroom, for example the ones described by Sandholtz, et al., (1997). In this sense, it could be said that these teachers' beliefs show a 'potential state of instructional evolution'. In order to analyse their 'real' state, a different piece of research would be needed (i.e. classroom/computer lab observation).

Finally, this opportunity to characterise what these teachers believed could be very useful for teacher trainers, in so far using this information they could be able to understand what teachers need to expand or change and thereby facilitate their instructional evolution towards an 'appropriate' use of Information Technology[23].

8.2.2 Issues about the characteristics of the software designed

This section is aimed at understanding the reasons that could explain the characteristics of the software that these teachers designed. In this framework, the description of the particular software designed is not the focus of this analysis, in so far this research was not aimed at producing a piece of software neither was it intended that this piece of software could serve as a model of the type of educational software that should be designed. Rather, the focus of the analysis here is what these teachers believed about educational software that lead them to design such characteristics and thereby to understand some educational software dimensions that they considered to be special.

The 'Contents Analysis' showed that, in general terms, these teachers designed the software focusing on issues that were closer to the design of a lesson and without much consideration of technical issues like human-computer interface design (for example: Laurel, 1990) (dialogue boxes, menus, icons, etc.) or special functions (printing, calculating, navigational aids, etc.). This finding is coherent with the overall picture presented until now, that is, teachers were planning a lesson that could be carried on primarily by the software.

The 'Participation Analysis' showed that the units corresponding to the group of categories of the branch 'Characteristics of the software' were the most frequently spoken units during the design process. In fact, 78% of the units of analysis found

[23] It could also be used to characterise the 'Zone of Proximal Development', or their current state of internalisation of Information Technology (Wertsch, 1985).

were classified in these categories and these two teachers spoke 50 % of these units. The group of categories that contains the most frequently spoken units was 'Interface Element', it represents 26% of what was spoken during the overall software design process. This fact by itself shows the time invested in designing the characteristics of the software, particularly what they considered to be the interface elements of the software.

The 'Sequences Analysis' showed that some of the groups of categories were frequently related to each other for example: 'Browsing' and 'Teaching Strategy' (996 sequences of these two units were found), 'Interaction' and 'Actions' (1029 sequences of these two units were found) and the three unit sequences 'Interactions', 'Actions' and 'Subject Areas' (63 sequences of these three units were found). Also it showed that the group of categories 'Content Organisation' was related to two other groups of categories ('Subject Areas' and 'Browsing) acting as a link between these categories.

As shown, analysing these findings as isolated pieces of information is limited in so far it enables to see just one dimension of the data and does not provide an answer to questions like:

- what were the teachers' intentions while designing the characteristics of the software ?
- why did these teachers speak so much about 'Interface Elements' ?
- why did these teachers speak so little about 'Subject Areas' ?
- why did they combine the groups of categories 'Browsing' and 'Teaching Strategy'?
- why did they combine the groups of categories 'Interaction', 'Actions' and 'Subject Areas'?

In order to answer such questions the three analyses together were needed to build up different layers of information. That is, combining the quantitative findings ('Participation' and 'Sequences' analyses) with the qualitative findings ('Contents Analysis'), it was possible to construct new meanings that provide a deeper understanding of what these teachers were designing. These meanings are:

- **The classroom atmosphere in the software**. Looking at the meanings of the groups of categories 'Interface Elements' and 'Content Organisation' described in the 'Contents Analysis' it was found that they contain the descriptions of the way in which these teachers intended that the contents should be presented to the pupils, that is, the design of a story that has the subject areas embedded. In this

sense, the atmosphere of the classroom acts as the link between the other dimensions of the lesson (pedagogy and learning).

- **The pedagogy in the software**: Looking at the meanings of the groups of categories 'Browsing' and 'Teaching Strategy' described in the 'Contents Analysis', it was found that they present the way in which these teachers implemented the pace of the lesson, that is, the way in which the software enables the pupil to follow the contents.
- **The learning dimension in the software**: Combining the meanings of the groups of categories 'Interactions', 'Actions' and 'Subject Areas' it was found that in these categories they were designing the way in which pupils would learn while using the software.

These new meanings provide a basis for understanding what these teachers were really designing and the reasons why they designed the software as they did. In this sense, they show the links that these teachers constructed between their teaching strategies and the characteristics of the software designed.

As mentioned before, these findings should be interpreted as what these teachers believed about educational software that would be used by children aged 4 to 6. That is, the particular characteristics of the software were designed for children that have such age and therefore that have particular learning potentials and requirements. Therefore the relevant dimensions of these findings are not the particular characteristics of the software, but the links that these teachers defined between the software's characteristics and the particular dimensions of teaching.

8.2.2.1 The classroom atmosphere in the software

The teachers decided to embed the curriculum contents in a playful story, where pupils could browse through this story, following the instructions given by a character in the software. They organised the contents dividing them into levels of difficulty and differentiating the types of contents (in this case different basic skills). The resulting design was represented as a matrix in which the rows organised the progression of different levels of difficulty and the columns organised the different types of contents. In this matrix, users could browse through the cells accordingly to their achievement and each possible path would constitute a coherent story for the pupil.

This section will analyse the reasons why these teachers decided to present the curriculum contents through a story and why they decided to implement these particular characteristics in the story. While doing this, it will draw on other studies

that show that it is a common strategy used by primary teachers, and that in doing so these teachers transferred their role as 'story tellers' to the software.

The fact that these teachers decided to use a story to embed the contents is largely described in the literature as a common strategy used by teachers in the classroom as a way to 'bring home' to the child the contents of the curriculum (Woods & Jeffrey, 1996), and establish 'common knowledge' (Edwards & Mercer, 1987). As presented in Section 7.3.2, the particular dialogue were they proposed to use this strategy was:

TM: Hey, and what if we invent a story and the child develops basic skills through the narrative of the story ?

...

TM: For example, the little Red Riding Hood, and there she goes [to visit her grandmother], because almost all children know little Red Riding Hood, then he [the user] has to put apples into the basket [of little Red Riding Hood], and she goes walking to her grand mother's house, and she goes into the house.

Here Teacher M is proposing to use a known story ("almost all children know...") and that the exercise ("putting apples into the basket") is embedded into the story's flow. In this sense, these teachers decided to use the same strategy in the software and designed it so that users should be immersed in the story and placed as protagonists of the activities, being able to move and touch all the elements presented in the different scenarios of the interface and having a dialogue with the character that prompted exercises and guided them through the software. The difference with teachers' traditional practice is that, in this case, the story would be told by the computer, not by the teacher.

While using a story, these teachers used the software as a way to create a 'world' in which the user is immersed[24], rather than providing only a model of the world that the user can manipulate. In the former users get cognitively involved in the story because of its motivation (as described by Lepper & Malone, 1987), in the latter they are outside observers and have access to this world, but are not 'inside' it, they manipulate the model but do not interact within the model.

The story was designed as to happen in a similar environment to the one known by the pupils. In fact, during the design process these teachers emphasised that the scenarios should resemble their real social and geographical environment of the

[24] In this sense what teachers designed follows the spirit of Virtual Reality environments, but without the technical features that enable the user to really 'live' the situation.

children, they rejected showing animals or landscapes from other countries (from Africa for example). This emphasis, on using known images and realistic situations, could have several explanations, such as:

- These teachers saw an opportunity to bring their neighbourhood into the computer, perhaps expecting that the children would feel 'proud' of seeing their own world in the computer in contrast to images of foreign scenarios that are represented in imported software.

- These teachers tried to copy the social environment in which the software is supposed to be used, placing instruction within 'authentic' context that mirror real-life problem-solving situations. In this case these teachers would have been trying to build a model of a situated cognition approach (Brown, et al., 1989) in the software, trying to give the children direct and concrete experiences that resemble real world situations, just as described by Edwards & Mercer, (1987) that 'progressive' teacher do.

- These teachers tried to build a shared mental context or image, and thereby enable the pupils to create common knowledge or shared understandings between the children and software (similar to the definition found in Edwards & Mercer, 1987).

It is not argued here that this software is the right implementation of these intentions but it is possible to argue that these teachers may have had in mind a combination of these ideas while designing the story in the software. In fact, these teachers spent large amounts of time speaking about the characteristics of each scenario that was proposed by the graphic designer. Their permanent critique was related to the fact that the images should be realistic and be a representation of their environment.

The progression through the story was designed considering the possibility of different navigational paths as a result of the pupils' levels of achievements. That is, using a matrix structure, they designed the software so that the user would follow different paths depending on their performance, and each of these paths would constitute a coherent story. This organisation of the contents was discussed at length during the design process and these teachers emphasised each time that the contents should be organised as a story with multiple paths, as opposed to more curriculum oriented organisations that were proposed by the Psychologist and the Software Engineer.

Similar software design principles are described in the literature as the possibility that technology gives to individualise instruction (Sandholtz, et al., 1997) where students work through the material at their own pace, and in this case, also depending on their own performance. Comparing this design with traditional drill and practice software, the difference in this case is that the software was organised in such a way that each pupil should be able to finish with a coherent story, in doing so, these teachers were worried about the time available for the lesson and pupils' sense of achievement (similar to what Woods & Jeffrey, 1996, describe as a common strategy used by primary teachers).

Looking at the design of the character in the story, it could be said that it provides the possibility of a more 'traditional' dialogue between the computer and the pupils, creating a sense of involvement of the user in the situation (similar to some primary teachers' strategies described by Woods & Jeffrey, 1996). Further on, this character was designed to provoke sympathy in the user, looking for an affective relation (as Cooper & McIntyre, (1995) describe that teachers also relate to their pupils). This can be seen while Teacher E describes the need of a narrator saying:

TE: But, I thought that we must have in the software a narrator. A fixed narrator in charge of telling [the story], a narrator that would engage with the children, that would be very meaningful for them, very contextualised ...

In this description it is possible to interpret that while designing this story these teachers intended to replicate a 'classroom atmosphere' in the software, transferring their role of 'story tellers' to the character of the software.

In a way, this type of design follows the principles of software described as tutor (Graesser, Person, & Huber, 1993; Reusser, 1993) or as adaptive (Laurillard, 1993), that accommodates its next action to the user's previous answer, giving meaningful feedback. In this case, if the matrix that organises the contents and the navigation possibilities had more cells, the software could have been classified under this category.

As mentioned in the previous section, the intention of these teachers could be seen as providing a tutor (or 'colleague') for the students in the software, that could implement the specific behaviours and actions necessary for a learning-teaching exchange to take place (in this case rehearsal), thus delegating to the software what Leinhardt, et al. (1987) call the 'support routines'.

8.2.2.2 The pedagogy in the software

The 'Participation Analysis' showed that the teachers did use their professional background during the design of the groups of categories 'Browsing'. The 'Sequences Analysis' showed that while designing the software these teachers consistently related their teaching strategies with the design of the browsing characteristics of the software. This section discusses this relation and will show that these teachers saw the navigation of the software as the main 'pedagogic' dimension of the software.

To start this discussion it must be remembered that in this research the group of categories 'Browsing' was defined as the dynamic sequence of contents that will be displayed while the user is advancing through the different stages of the software. In this sense, browsing can be understood as the contents' presentation strategy embedded in the software. On the other side, the group of categories 'Teaching Strategy' was defined as discussions about behaviours of the software that relate to the teaching conceptions of these teachers.

In the qualitative analysis it was shown that during the design of the browsing strategy the teachers had a breakdown with the other members of the development team. After several discussions, the teachers' model of browsing was accepted, and so the final browsing strategy was (same as figure 7.5):

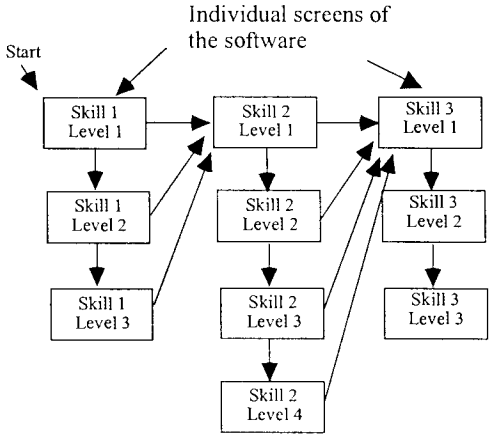

Figure 8.1. The final browsing structure of the software

Looking at what was designed as the browsing structure of the software, it is possible to say that while designing it, the teachers were trying to design the sequence of contents that each user would be able to learn or rehearse. They were planning a teaching sequence, and determining the circumstances under which certain contents should be taught or not. It was not simply a matter of organising the contents and enabling the users to be able review all of them. On the contrary, they designed it

taking into account that the user would be learning and therefore the browsing strategy was associated with teaching strategies.

In fact, the resulting design can be understood in terms of some reports about the way in which teachers normally organise their teaching (without computers). For example, Hammersley, (1990) writes:

> The lesson as presented by the teacher is pitched at a certain level of 'difficulty' according to the co-ordinate position of the class in relation to age and ability.
>
> (Hammersley 1990, p.47)

Transferring this lesson design to the software, it could be said that it was designed as to have the possibility to be 'pitched' at several levels of difficulty (at each cell of the matrix) and these levels of difficulty would be accessed by the users accordingly to their performance (the pupil would work on a 'cell of the matrix'), responding automatically to the 'co-ordinate position' of each pupil. In relation to this type of classroom organisation, Hammersley (1990) argues that:

> This pre-setting is designed not only to ensure that pupils are taught something 'new', that they 'keep moving', but also that they have the resources to understand what the teacher is to teach.
>
> (Hammersley 1990, p.47)

These teachers solved some of these problems by designing a special browsing strategy in the software, but they still had to ensure that the pupils would be able to browse through this structure. Therefore they had to design the way in which the software should 'guide' the pupils through it. In Hammersley (1990)'s words:

> However, despite this pre-setting, teachers have continually to check that the pupils do understand what is being taught. The teacher's control and development of lesson-topic constitutes the most important source of clues to what he is asking for.
>
> (Hammersley 1990, p.47)

In order to ensure this condition, they addressed the following issues which were classified under the group of categories 'Teaching Strategy' and reported in section 7.7:

- Checking that pupils did understand, and what to do if the did not. In this case, the software's response to a wrong user's answer. They discussed whether the computer should teach the users the right answer or give them clues and ask again. That is the type and content of the feedback for the user.

- The development of the 'lesson': In this case: conditions for the user's progress through the software, that is, under which conditions should users step into a higher level of difficulty.

- The 'control' of the lesson. The software's response if the user starts to play instead of following the instructions.

This is, in the group of categories 'Teaching Strategy' it is possible to find additional teaching strategies that complement the design of the browsing strategy of the software.

The way in which these teachers designed the software to implement these dimensions is not what constitutes the interesting finding, in so far they continued applying their 'normal' teaching strategies to solve them. What is interesting is the fact that these dimensions of teaching were considered to be part of the browsing characteristics of the software. That is, they embedded their teaching patterns into the way in which the software guides the user through the contents.

This shows not only that these teachers believed that the computer could in fact guide the user and perform some of their roles, but also the relevance that the browsing strategy has while designing educational software. In this sense, these teachers implicitly identified this dimension and designed it to copy their behaviour or 'expertise'.

About the type of role delegated to the software through the browsing strategy, in this case the role is closer to what is defined by Leinhardt, et al. (1987) as the exchange routines, that is, the way teachers enter into dialogue with the pupils, the way teachers pose questions to the students or give feedback to them. Looking at this issue from a different theoretical perspective, the teachers' design of the browsing strategy could be seen as a style of teaching where the software negotiates the learning outcomes with the pupils, that is, the software would let them go at a higher level of difficulty only if they gave the right answer to the software's prompting, thereby controlling what could be learned next. This style of teaching is similar to what Cooper & McIntyre, (1995) defined as an 'Interactive' style of teaching. This interactive style is contrasted by Cooper & McIntyre (1995) with a model of teaching where pupils are in control of what they want to learn, which they call the 'Reactive' style of teaching.

8.2.2.3 The learning dimension in the software

The 'Participation Analysis' showed that the teachers did use their professional background for the design of the groups of categories 'Interaction' and 'Actions-Pupil'. Coincidentally, during the 'Sequence Analysis' it was shown that the categories 'Interaction' and 'Actions' were frequently related to each other, that is, they were often referred to consecutively in the data. Further, the actions designed were mostly of the pupils interacting with the computer. The fact that these two categories were found frequently to be consecutive is not therefore surprising. In the same analysis it was also found that they related these two categories with the group of categories 'Subject Areas', which configures an interesting picture in so far as it indicates that these teachers designed the interactions and the actions of the pupils while focused on the contents to be learned.

In general terms, these teachers designed that the pupil should act as a respondent to the software's questions, that is, they assumed that the pupils would participate in a playful interaction with the computer that had the pattern of a guessing game (as Edwards & Mercer, 1987 describe that 'progressive' teachers plan their lessons) and that the dialogue with the computer would be based on questions, answers and feedback (as described by Hammersley, 1990 to happen in a normal classroom). The type of questioning that they designed in the software was closer to the sort of recitation questions (as defined by Gall & Artero-Boname, 1995) in so far as the pupils were required to 'execute' exactly what the software was asking them to do.

The fact that these actions (groups of categories 'Actions' and 'Interaction') were combined with the group of categories 'Subject Areas' as shown in the 'Sequences Analysis', enables us to say that these teachers were designing the way in which the contents should be presented to the user. That is, they designed the way in which the pupils should interact with the contents and eventually learn them.

From this perspective, it could be said that these teachers believed that the process of learning, or rehearsing in this case, would take place during the pupil's answering of the software's questions.

Looking at the type of interaction designed and the actions of the pupils it is possible to understand the learning procedure that these teachers had in mind while designing the software, that is, the way in which they believed that learning takes place. Some evidence that shows the general framework of this model was already presented in previous sections of this chapter, particularly when teachers explained that "learning by doing is very important, and that the child is constructor of his own learning and

we are guides in this". It is therefore possible to argue that teachers did not have a behaviourist model of teaching and learning in mind, at least they did not believe that they were using a behaviourist model.

It is possible to identify some characteristics implemented in the software that lead us to say that these teachers had in mind:

> a version of what is popularly called 'progressive' [teaching] - an approach normally contrasted with the 'traditional' one, which relies heavily on didactic methods and formal procedures
>
> (Edwards & Mercer, 1987, p. 35, our brackets)

The principles of this 'progressive' approach are described by Edwards and Mercer (1987) as:

1. Setting up conditions which they believe would allow children to discover things for themselves.
2. Planning their teaching to include activities which would give children direct, concrete experience, and which would require them to act, not just listen, read or write.
3. Attempting to refer to children's wider out-of-school experience when planning curriculum topics (in the sense of 'general knowledge', but hardly ever by reference to the particular life experience of any one child in the group).
4. By the use of techniques like the 'guessing game' question and answer sessions, to elicit 'key' ideas from children rather than informing them of these directly.
5. Never defining (for the children) the full agenda of any activity or lesson in advance.
6. Not defining explicitly (for the children) the criteria for successful learning which would eventually be applied to what they had done.

(partial transcript of Edwards and Mercer, 1987, pp. 33-34)

This confirms that they were acting as teachers and modelling what they thought they did as teachers[25]. In itself this is not an issue, but the fact that they tried to apply these principles to the design of the software (with greater or lesser success) implies first that they believed that the software should teach following these principles, and second, that in fact learning could take place during this interaction.

- **Summary of the discussion**

In this section the characteristics of the software were analysed and it was shown how these teachers constructed the software to resemble a lesson. This discussion leads us to suggest that these teachers had the following beliefs about the software:

[25] Given the data in this research and the period in which it was obtained it is not possible to know if what they described during the design process was what they really did in the classroom.

- That the software is able to provide the required atmosphere for the teaching and learning process to take place, using a story that engage the pupils and has the required balance between fantasy and realism to provide 'common knowledge' (Edwards & Mercer, 1987) between pupils and software.

- That the software should have a browsing structure that enables it to individualise instruction, providing self 'pitching' mechanisms and the required feedback for the pupil. So, the pedagogy would be determined by the browsing of the software and each pupil would be able to do what (s)he should and could do.

- That the learning occurs during the interaction with the software and that this interaction should be designed following accepted principles and practices of teaching.

While discussing these beliefs it was shown that these teachers designed the software applying their strategies of teaching and embedding their learning conceptions. In doing so, they were prompting the need for considering such dimensions during the design of educational software. Further, through this discussion it was possible to identify in which dimension of the software such elements should be implemented.

Finally, considering these three dimensions it is possible to build the model of educational software that these teachers had and also it is possible to understand the reasons why they build such a model.

8.3. METHODOLOGICAL PERSPECTIVES

From a methodological perspective, there are three areas to discuss, the definition of this research as a case study, the systemic network defined to create the categories of analysis and the combination of qualitative and quantitative methods of analysis.

8.3.1 The case study

The discussion about the methodological framework of this research will be guided by the description of the critiques to case studies reported by Yin, (1994) (see section 2.2 of the methodological chapter for details).

In the first place there are three general critiques of case studies: (i) the lack of rigor, (ii) the provision of little basis for scientific generalisation, and (iii) they take a too long time period and produce massive unreadable documents.

About the first critique, the relevant data for this research was a consequence of the software development process, not the process itself. Considering the process, it was carried out avoiding interventions of the researcher or of any other known source of bias. In this sense, this piece of research can be defined as an observation of a natural setting. About the resulting data, the process of transcription, coding and analysis was done considering all the required additional processes (multiple revisions, re-coding, validation, etc.) to be able ensure reliable data, and where it was not possible to do so, the reader has been informed about the possible deviation (see the coding validation and the description of the systemic network for evidence). Finally, about the possible bias in the interpretation of the data, three different methods of analysis were used in order to have a sort of triangulation of the conclusions to be drawn. Nevertheless, despite the procedures taken to avoid subjectivity, while interpreting the findings the researcher's conceptions and beliefs would necessarily be present, as Eisner, (1993), says:

> The relativity of my views pertain to the belief that knowledge is always constructed relative to a framework, to a form of representation, to a cultural code, and to a personal biography.
>
> (Eisner, 1993, p. 54)

In the light of this inquiring paradigm (Guba & Lincoln, 1994), these findings and implications depend on the questions that the researcher asked and therefore constitute one possible answer that needs to be contrasted with other findings in order to be generalised.

In fact, related to the possibilities of generalisation, in this case it is not possible to generalise starting from the data itself, but some general claims can be made by drawing on theoretical evidence that support these findings. For example: the studies reported by Olson (1988); Sandholtz, et al. (1997) and Schofield (1995), the teaching principles described by Edwards & Mercer (1987), including additional evidence from Hammersley (1990) and Woods & Jeffrey (1996), the teaching practices described by Cooper & McIntyre (1995); Jones (1996); Leinhardt, et al. (1987) and Mayer (1995). These theoretical frameworks give the basis for defining a model of understanding of educational software.

As regards the criticism regarding the long time and massive documents, this depends on the analysis technique used, in this case the time period and the size of the end document is about that of an average thesis.

From a different perspective, there are four areas to be addressed while discussing the methodology: construct validity, internal validity, external validity and reliability. With respect to the construct validity, in this case one possible source of bias were the teachers involved in the process, therefore they were not informed about the final purpose of the activity that they were carrying out. The raw data consisted of the transcripts of the software design sessions which were reviewed and checked several times. At a different level the evidence for this case were the data categorised in the systemic network which was analysed from three different perspectives. In this sense, it could be argued that the conclusions are based on different sources of information as recommended by Yin, (1994).

About the internal validity of this case study, although this is intended to concern only causal (or explanatory) case studies, where the investigator is trying to determine whether event x led to event y, it is possible to argue that the comparison and contrast of findings in the different methods of analysis provides evidence to argue that the ideas and conceptions expressed are internally coherent.

The external validity of this case study has just been addressed and the possible generalisations are based on other theoretical frameworks from different areas that inform these findings. Because the model proposed here is sustained by these theories, it is possible to argue that it can be generalise to some degree. Nevertheless, it must be kept in mind that this research reports only one case involving only two teachers.

The reliability of this case study has two dimensions, on the one hand it is not possible to repeat the process that constituted the source of raw data. Even if the participants would agree to repeat the 19 sessions of software design, they have changed during time and because of their experience in this research they would behave and (probably) think different. But on the other hand, if someone starts from the raw data gathered (videos, sound records or the transcripts in case of starting at a higher level) it is possible to replicate the analysis, so far all the data are present and available on request. Further, the systemic network which guided the analysis, is described and also available. Finally, the different analyses that were carried on are presented as well as the evidence from which conclusion were drawn. Hence, at least, this research can be considered reliable.

8.3.2 The systemic network

As Bliss & Ogborn, (1979) suggest, a systemic network was used to give structure to the data gathered and thereby to be able to analyse and discuss the codes used.

Several aspects of this network were already discussed in section 4.4, and there is one further dimension that will be discussed here, this is, the construction process of the network.

About the process of defining the systemic network, similar reports can be found in the literature (for example: Bliss, et al., 1983), but the examples shown refer, generally, to a process of convergence only (simplification of the network). In this case, the process of definition of the network started using a rather theoretically driven proposition which was then refined using an iterative process of coding samples of the data and revising the consistency of the categories defined in the network. As shown in the Systemic Network Chapter, the process of refinement had two dimensions: (i) the qualitative one (that considers the concepts expressed through the network, including the internal consistency and the general coherence) and (ii) the quantitative one (expressed in terms of the number of end-nodes and level of deepness of the network). The former process showed a convergence by the redefinition and relocation of nodes and branches in the network. The latter, showed first an increment (divergence), increasing the number of end-nodes and the level of deepness and then the complexity decreased (convergence).

This consideration on the process of building systemic network could help other researchers to differentiate both dimensions (qualitative and quantitative) and to conduct the process of definition being aware of their existence.

8.3.3 Qualitative versus quantitative analysis

The analysis of the data gathered and organised through the systemic network consisted of three different processes:

- Participation analysis: Aimed at establishing individual participation profiles during the development process, based on the calculation of the frequencies and distributions of participation of each team member in each category.
- Sequences analysis: Aimed at establishing the inter-relations among the different components of the software, based on the analysis of the patterns of sequences of units in the data for the group and each team member.
- Contents analysis: Aimed at looking for the meanings expressed by the teachers about the different dimensions of a piece of educational software.

The 'Contents Analysis' was used as the main set of evidence for the implications drawn from this research, the 'Participation Analysis' was used to ensure all member's participation and to ensure that teachers' participation was significant and

thereby validate the process. It also served to draw a profile of each participating teacher, depending on his or her participation in different categories. That is, the 'Contents Analysis' showed the meaning and purpose of each category and the 'Participation Analysis' showed who was responsible for these claims, and therefore for these meanings. The 'Sequences Analysis' was used to define new semantic categories in the data, combining the ones initially defined in the network. These new categories were used to support the claims shown in the 'Contents Analysis'. That is, the three types of analysis were combined and cross referenced in order to give internal validity to the research.

The use of qualitative or quantitative methods alone would have resulted in probable misinterpretations or a lost of focus. This is, the participation analysis alone might have lead one to argue that the software design process was dominated by one of the teachers and the Software Engineer. This conclusion alone would have missed the fact that some of the claims made by the other teacher were immediately accepted by the group without discussion. The sequence analysis alone would not have been enough to understand the reasons why these teachers combine certain categories. Finally the meanings analysis alone would have failed to ensure that the claims included were representative of what was said during the design process and thereby the validity of the research might have been questioned.

The combination of qualitative and quantitative methods served a double purpose, on the one hand it helped to focus the research analysis to relevant issues and also it provided a way to triangulate the findings. In this sense, the combination of these methods may have produced a more reliable 'picture' of what happened during the software design process and therefore the claims and implications have greater validity (Wegerif & Mercer, 1997).

8.4. IMPLICATIONS

In this research I have started from the position that it is important to give consideration to the reality of use of educational software in order to design, develop and evaluate software that can be used to support an innovation process which engages the teacher and the school..

These teachers were in fact engaged in an innovation process because they were asked to design a piece of software. In this sense, their activity in designing the software can be understood as part of the design of an innovation that would be supported by the software implemented.

The discussion within the design activities supports a view that the teachers expressed some of their beliefs about information technology in so far as they wanted to use it in their classrooms, therefore it was possible to construct a model describing their understanding of information technology. This model describes only one instance of their understanding of information technology, in so far as it was based only on what they said and it seems reasonable to assume that they did not say everything what they believed about information technology (i.e. they could understand it differently, depending on the purpose or context).

This model of understanding of information technology has two dimensions, one related to their conception of the role of the computer and the other about their conception of software. These two dimensions have particular sets of implications, which are:

- First, that the computer could be understood as an aid for teaching. The actual environment in which most computers are used (computer lab) was considered to be a separate place from the classroom. In this place the computer was used as a control tool by the teacher and was conceived of as a resource for teaching that acts as a rehearsal tool for the pupils.

- Second, that a prescriptive model of software for teaching could exist, which considers at least three dimension of teaching practice: classroom atmosphere, teaching strategy and learning strategy. Each of these dimensions should be considered while designing the human-computer interface (scenarios and elements), the browsing strategies of the software and the interaction with the software, respectively.

This model of understanding of information technology is grounded in two sources of evidence, first, the data collected in this research and second the evidence reported in some other studies about common teachers' practices in the classroom, (i.e. their strategies, routines, roles and some beliefs). This latter source of evidence expands the theoretical framework of the thesis, covering not only the area of information technology in education, but also the area of pedagogy. In this sense, this model of understanding of information technology constitutes an interesting stand point for further research that would look into the design of educational software from such a perspective.

The combination of these two areas opens a third set of implications about the need for cross-references between these two areas and in particular about the need for considering such issues in teacher training programs.

Finally, it needs to be said that we draw implications and hence we do generalise, but only tentatively in the full recognition that this work is based on a small scale study. However the study does give rise to a number of research questions that were mentioned in the previous sections and which can only be answered by further research.

8.4.1 Computer aided teaching (CAT)

In discussing the role of the computer in the classroom, these teachers indicated how important for them the role of the computer as rehearsal tool is, it was the role that they naturally picked-up and expanded on. They designed this role by integrating it into a larger teaching strategy that separated teaching into two stages: learning new concepts and rehearsing these concepts. As they themselves said, the former would take place in the classroom, directed by the teacher and without the computer, and the latter would take place in the computer room where the software would act as a rehearsal environment and the teacher would take control of the discipline. This model of use of the computer is represented in figure 8.2.

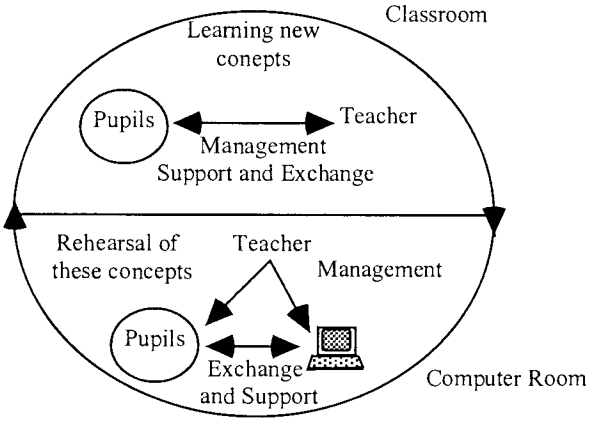

Figure 8.2. Model of the role of the computer

In this model the computer is acting as a resource for the teacher that helps him/her carry out more effectively some of their current tasks. Some ground rules in this model are that:

- teachers do not interact with the computers,
- the teacher will teach subjects in the classroom and not in the computer lab,
- the teacher guides the pupil's construction of knowledge keeping control of the students and solving computer problems, (s)he does not 'interfere' with the contents to be learned (when working in the computer room, not when working in the classroom)

- computers serve as tools for controlling the class that is they form part of the punishment-reward system of the classroom.

Some of the individual elements of this model are not new, but the fact that these teachers presented them as a coherent picture could imply two things: first, it could give a sound explanation for the actual use of computers as drill and practice machines and second it could be a starting point for re-considering the potential of computers not only as learning tools (running learning centred software), but also as teaching tools (running teaching centred software).

This case study provides evidence that at least these teachers consider that the computer could be used as an aid for efficiency while it takes over the role of conducting the rehearsal stage. So, it opens a window to consider the computer not only as a Trojan horse (Olson, 1988) that triggers innovation and change in terms of 'better teaching' or as a 'cognitive tool' that provides an environment that is mind-extending or as catalysing tools for intelligent and volitional learners (as defined by Orhun, 1995 and Reusser, 1993), but also as a teaching tool, that helps teachers to do their teaching better.

8.4.2 Teaching centred software vs. learning centred software

Our analysis of what the teachers said during the design of the software leads us to consider that a prescriptive model of 'teaching centred software' could be developed. Apart from considering the particular characteristics of the software they designed, what is relevant are the teaching dimensions that they implicitly or explicitly addressed during the design process. These dimensions were 'Classroom Atmosphere and Tone', 'Pedagogy', 'Learning Strategy' and 'Classroom Management'. This latter dimension was considered to be the teachers' exclusive role. Further, through the analysis of what they said, it was possible to identify the software characteristics to which these dimensions were correlated and therefore could be implemented. Table 8.1 summarises this correspondence.

The implication in this case is that at least these four dimensions should be considered while designing and implementing educational software. As a consequence, these findings open new areas of research that could help to identify further dimensions of the teaching practice that should be considered, and provide a starting point to design how such dimensions could be incorporated into educational software.

Incorporating these dimensions into the design of educational software could lead to implications for software evaluation, in so far as it could give structure and additional

theoretical frameworks for judging some characteristics of the software. Particularly it could enrich the 'Perspectives Interaction Paradigm' proposed by Squires & McDougall, (1994) for educational software evaluation, giving additional criteria for evaluating the student-teacher interaction, that is, the kind of classroom interactions and activities that might be fostered by the software.

Teaching Domain	Software Design Domain
Classroom Atmosphere and Tone	The human-computer interface elements (scenarios, backgrounds, characters and particular functionality of these elements) and the overall organisation of the subjects.
Pedagogy: Contents provision, lesson flow and control of the user's progress	The browsing strategy of the software and the response to certain situations (for example, feedback on errors)
Learning Strategies or Theories	Particular interaction with the software and user's actions.
Classroom Management routines	It was excluded from the software

Table 8.1. Correspondence between teaching and software design domains.

Even if the particular characteristics of the software designed could be considered to be badly designed or ill implemented, the fact that these links between two theoretical areas exist, constitutes an interesting finding of this research and as a case study, it provides enough evidence to justify and promote further research in this area that could help identify other links between teachers' roles and actions in the classroom and the roles that a piece of educational software could (or should) have.

IX. REFERENCES

Adler, P. A., & Adler, P. (1994). Observational techniques. In N. K. Denzin & Y. S. Lincoln (Eds.), Handbook of qualitative research (pp. 377-392). London: Sage.

Alexanderson, M. (1994). Focusing teacher consciousness: What do teachers direct their consciousness towards during their teaching ? In I. Carlagen, G. Handal, & S. Vaage (Eds.), Teachers' minds and actions: Research on teachers' thinking and practice (pp. 139-149). London: The Falmer Press.

Anderson, A., Tolmie, A., McAteer, E., & Demissie, A. (1993). Software style and interaction around the microcomputer. Computers and Education, 20(3), 235-250.

Ball, S. J. (1993). Self-doubt and soft data: Social and technical trajectories in ethnographic field work. In M. Hammersley (Ed.), Educational research: current issues (pp. 32-48). London: Paul Chapman & Open University.

Barret, M. (1995). Practical and ethical issues in planning research. In G. M. Breakwell, S. Hammond, & C. Fife-Shaw (Eds.), Research Methods in Psychology (pp. 16-35). London: Sage.

Baumgartner, P., & Payr, S. (1997). Methods and practice of software evaluation: The case of the European academic software award. In T. Müldner & C. T. Reeves (Eds.), World Conference on Educational Multimedia and Hypermedia. Calgary: Association for Advancement of Computing in Education.

Berliner, D. C. (1995). Teacher expertise. In L. W. Anderson (Ed.), International encyclopedia of teaching and teacher education (pp. 46-52). Oxford: Pergamon.

Bliss, J., Monk, M., & Ogborn, J. (1983). Qualitative data analysis for educational research: A guide to users of systemic networks. London: Croom Helm.

Bliss, J., & Ogborn, J. (1979). The analysis of qualitative data. European Journal of Science Education, 1(4), 427-440.

Boehm, B. W. (1988). A spiral model of software development and enhancement. IEEE Computer(May), 61-72.

Bostock, S. J. (1996). A critical review of Laurillard's classification of educational media. Instructional Science(24), 71-88.

Brooks, F. P. J. (1987). No silver bullet: Essence and accidents of software engineering. IEEE Computer(April), 80-89.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher(January-February), 32-42.

Bruce, B. C. (1997). Educational technology: Media for inquiry, communication, construction, and expression. Journal of Educational Computing Research, 17(1), 79-102.

Chandler, D. (1984). Young learners and the microcomputer. London: Milton Keynes, Open University.

Char, C., & Hawkins, J. (1986). Charring the course: involving teachers in the formative research and design of the voyage of the mimi. In R. D. Pea & K. Sheingold (Eds.), Mirrors of mind: Patterns of experience in educational computing (pp. 211-241). Norwood: Abelex Pub. Co.

Chen, M. (1995). A methodology for characterizing computer-based learning environments. Instructional Science, 23, 183-220.

Cohen, L., & Manion, L. (1994). Research methods in education (4th ed.). London: Routledge.

Cooper, P., & McIntyre, D. (1995). The crafts of the classroom: Teachers' and students' accounts of the knowledge underpinning effective teaching and learning in classrooms. Research Papers in Education, 10(2), 181-216.

Cox, M. J. (1997). Identification of the changes in attitude and pedagogical practices needed to enable teachers to use information technology in the school curriculum. In D. Passey & B. Samways (Eds.), Information technology: Supporting change through teacher education (pp. 87-100). London: Chapman & Hall.

Crook, C. (1987). Computers in the classroom: defining a social context. In J. Rutkowska & C. Crook (Eds.), Computers, cognition and development (pp. 35-53). Chichester: John Wiley & Sons.

Crook, C. (1991). Computers in the zone of proximal development: implications for evaluation. Computers in Education, 17(1), 81-91.

Crook, C. (1994). Computers and the collaborative experience of learning. London: Routledge.

Crook, C. (1998). Children as computer users: The case of collaborative learning. Computers and Education, 30(3/4), 237-247.

Cuban, L. (1997). Foreword. In H. J. Sandholtz, C. Ringstaff, & D. C. Dwyer (Eds.), Teaching with technology: Creating student centered classrooms. New York: Teachers College Press.

DeGrace, P., & Hulet Stahl, L. (1990). Wicked problems, righteous solutions: A catalogue of modern software engineering paradigms. New Jersey: Yourdon Press - Prentice Hall.

diSessa, A. A., Hoyles, C., & Noss, R. (Eds.). (1995). Computers and exploratory learning. London: Springer.

Dockterman, D. A. (Ed.). (1991). Great teaching in the one computer classroom. Tom Snyder Productions.

Dreyfus, H. L., & Dreyfus, S. E. (1986). Mind over machine: The power of human intuition and expertise in the era of the computer. Oxford: Blackwell.

Edwards, D., & Mercer, N. (1987). Common knowledge: Development of understanding in the classroom. London: Routledge.

Eisner, E. (1993). Objectivity in educational research. In M. Hammersley (Ed.), Educational research: current issues (pp. 49-56). London: Paul Chapman & Open University.

Elbaz, F. (1990). Knowledge and discourse: The evolution of research on teacher thinking. In C. Day, M. Pope, & P. Denicolo (Eds.), Insight into teachers' thinking and practice (pp. 15-42). London: The Falmer Press.

Eraut, M. (1994). Developing professional knowledge and competence. London: The Falmer Press.

Eraut, M. (1996). Concepts of educational technology. In A. C. Tujinman (Ed.), International encyclopedia of adult education and training Oxford: Pergamon - Elsevier Science.

Erickson, R., & Wilson, I. (1984). Making and using research documents of everyday life in schools. In Sights and sounds of life in school: A reference guide to film and videotape for research in education (pp. 39-63).

Evans-Andris, M. (1995). An examination of computing styles among teachers in elementary schools. Educational Technology Research and Development, 43(2), 15-31.

Evertson, C. M. (1995). Classroom rules and routines. In L. W. Anderson (Ed.), International encyclopedia of teaching and teacher education (pp. 215-218). Oxford: Pergamon.

Fatouros, C., Downes, T., & Blackwell, S. (1994). In control: Young children learning with computers. Wentworth Falls: NSW: Social Science Press.

Fitzgerald, G. E., Bauder, D. K., & Werner, J. G. (1992). Authoring cai lessons: Teachers as developers. Teaching Exceptional Children, Winter, 15-21.

Fontana, A., & Frey, J. (1994). Interviewing: The art of science. In N. Denzin & Y. Lincoln (Eds.), Handbook of qualitative research (pp. 361-376). London: Sage.

Fraser, R., Burkhardt, H., Coupland, J., Philips, R., Pimm, D., & Ridgway, J. (1991). Learning activities and classroom roles with and without the microcomputer. In O. Boyd-Barret & E. Scanlon (Eds.), Computers and learning Wokingham: Addison Wesley & Open University.

Fullan, M. (1982). The meaning of educational change. New York: Teachers College Press.

Fullan, M. (1992). Successful school improvement: The implementation perspective and beyond. Buckingham: Open University Press.

Fullan, M. (1993). Change forces: Probing the depth of educational reform. London: The Falmer Press.

Fullan, M. G. (1996). Implementation of innovation. In T. Plomp & D. E. Ely (Eds.), International encyclopedia of educational technology (pp. 273-281). Oxford: Elsevier Science - Pelgrum.

Fullan, M., & Stiegelbauer, S. (1991). The new meaning of educational change (2nd ed.). London: Cassel Educational Limited.

Gall, M. D., & Artero-Boname, M. T. (1995). Questioning. In L. W. Anderson (Ed.), International encyclopedia of teaching and teacher education Oxford: Pergamon.

Galvis, A. H. (1994). Ingeniería de software educativo. Santafé de Bogotá, Colombia: Ediciones Uniandes.

Gilb, T. (1988). Principles of software engineering management. New York: Addison Wesley.

Graesser, A. C., Person, N. K., & Huber, J. (1993). Question asking during tutoring and in the design of educational software. In M. Rabinowitz (Ed.), Cognitive science foundations of instruction (pp. 149-172). Hillsdale: Lawrence Erlbaum.

Grossman, P. L. (1995). Teachers' knowledge. In L. W. Anderson (Ed.), International encyclopedia of teaching and teacher education (pp. 20-24). Oxford: Pergamon.

Grunberg, J., & Summers, M. (1992). Computer innovation in schools: a review of selected research literature. Journal of Inforrmation Technology for Teacher Education, 1(2), 255-276.

Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. In N. Denzin & Y. S. Lincoln (Eds.), Handbook of qualitative research (pp. 105-117). London: Sage.

Hammersley, M. (1990). Classroom ethnography: empirical and methodological essays. Philadelphia: Open University Press - Milton Keynes.

Hammersley, M., & Atkinson, P. (1983). Ethnography: Principle in practice. London: Tavistock.

Hammond, K. R., & al., e. (1980). Human judgement and decision making. New York: Hemisphere.

Handler, M. (1993). Preparing new teachers to use computer technology: Perceptions and suggestions for teacher educators. Computers and Education, 20(2), 147-156.

Hawkins, J., & Kurland, M. D. (1986). Informing the design of software through context-based research. In R. D. Pea & K. Sheingold (Eds.), Mirrors of mind: Patterns of experience in educational computing (pp. 258-272). Norwood: Abelex Pub. Co.

Hawkridge, D., Joworosky, J., & McMohan, H. (1990). Computers into third-world schools: Examples, experiences and issues. London: McMillan.

Hepp, P., Laval, E., Moënne, G., & Ripoll, M. (1996). Monitoring the 'Enlaces' educational computer network. Education and Information Technologies, 1(1), 5-20.

Hepp, P., Rehbein, L., Hinostroza, E., Laval, E., Dreves, C., & Ripoll, M. (1994). Enlaces' A hypermedia based educational network. In ACM Multimedia: The Second International Conference on Multimedia, . San Francisco, California, USA:

Hinostroza, E., Hepp, P., & Straub, P. (1996). Un método de desarrollo de software educativo. Informática Educativa, 9(1), 9-32.

Hinostroza, E., Mellar, H., Rehbein, L., Hepp, P., & Preston, C. (1997). Diseño de software educativo o software escolar ? Informática Educativa, SIIE, Colombia, 10(1), 57-73.

Hoyles, C., Noss, R., & Sutherland, R. (1991). Final report of the microworlds project. London: Institute of Education, University of London.

Huberman, M. (1992). Critical introduction. In Successful school improvement: The implementation perspective and beyond. Buckingham: Open University Press.

Hughes, J., King, V., Rodden, T., & Andersen, H. (1995). The role of ethnography in interactive systems design. Interactions, 2(2), 56-65.

Hurst, P. (1983). Implementing educational change: A critical review of the literature (EDC Occasional Papers No. 5). University of London, Institute of Education.

Johnson, D. C., Cox, M. J., & Watson, D. M. (1994). Evaluating the impact of IT on pupils' achievements. Journal of Computer Assisted Learning(10), 138-156.

Jones, V. (1996). Classroom management. In J. Sikula (Ed.), Handbook of research on teacher education (pp. 503-521). New York: Macmillan.

Kemmis, S., Atkin, R., & Wright, E. (1977). How do students learn? (Working Papers on CAL No. Occasional Paper N° 5). Centre for Applied Research in Education, University of East Anglia, UK.

Koedinger, K. R., & Anderson, J. R. (1993). Reifying implicit planning in geometry: Guidelines for model-based intelligent tutoring systems design. In S. P. Lajoie & S. J. Derry (Eds.), Computers as cognitive tools (pp. 15-45). Hillsdale: Lawrence Erlbaum.

Laborde, J.-M. (Ed.). (1995). Intelligent Environments: The case of geometry. London: Springer.

Lajoie, S. P. (1993). Computer environments as cognitive tools for enhancing learning. In S. P. Lajoie & S. J. Derry (Eds.), Computers as cognitive tools (pp. 261-288). Hillsdale: Lawrence Erlbaum.

Lajoie, S. P., & Derry, S. J. (Eds.). (1993). Computers as cognitive tools. Hillsdale: Lawrence Erlbaum.

Laurel, B. (Ed.). (1990). The art of human interface design. New York: Addison Wesley.

Laurillard, D. (1990). Computers and the emancipation of students: giving control to the learner. In O. Boyd-Barret & E. Scanlon (Eds.), Computers and learning (pp. 64-80). Wokingham: Addison Wesley & The Open University.

Laurillard, D. (1993). Rethinking university teaching: A framework for the effective use of educational technology. London: Routledge.

Learning (1995). Learning with Software. In www: http://gwis2.circ.gwu.edu:80/~kearsley/: Open Learning Technology Corporation Limited.

Leinhardt, G., Weidman, C., & Hammond, K. M. (1987). Introduction and integration of classroom routines by expert teachers. Curriculum Inquiry, 17(2), 135-176.

Lepper, M. R., & Malone, T. W. (1987). Intrinsic motivation and instructional effectiveness in computer-based education. In R. E. Snow & M. J. Farr (Eds.), Aptitude, learning and instruction. Volume 3: Conative and affective process analyses (pp. 255-296). Hillsdale: Erlbaum.

Lowther, D., & Sullivan, H. J. (1994). Teacher and technologist believes about educational technology. Educational Technology Research and Development, 42(4), 73-87.

Mantovani, G. (1996). Social context in HCI: a new framework for mental models, cooperation and communication. Cognitive Science(20), 237-269.

Marton, F. (1994). On the structure of teacher's awareness. In I. Carlagen, G. Handal, & S. Vaage (Eds.), Teachers' minds and actions: Research on teachers' thinking and practice (pp. 28-42). London: The Falmer Press.

Mayer, R. E. (1995). Feedback. In L. W. Anderson (Ed.), International encyclopedia of teaching and teacher education (pp. 249-251). Oxford: Pergamon.

McConnel, D. (1994). Implementing computer supported cooperative learning. London: Kogan Page.

McDonald, H., & Ingvarson, L. (1997). Technology: A catalyst for educational change. Journal of Curriculum Studies, 29(5), 513-527.

McDougall, A., & Squires, D. (1995). An empirical study of a new paradigm for choosing educational software. Computers and Education, 25(3), 93-103.

McNamara, D. (1994). Classroom pedagogy and primary practice. London: Routledge.

Mellar, H., Bliss, J., Boohan, R., Ogborn, J., & Tompsett, C. (Eds.). (1994). Learning with artificial worlds: Computer based modelling in the curriculum. London: The Falmer Press.

Mercer, N. (1993). Computer-based activities in classroom contexts. In P. Scrimshaw (Ed.), Language, classrooms and computers (pp. 27-39). London: Routledge.

Mercer, N., & Scrimshaw, P. (1993). Researching the electronic classroom. In P. Scrimshaw (Ed.), Language, classrooms and computers (pp. 184-191). London: Routledge.

Olsen, N. C. (1993). The software rush hour. IEEE Software, 10(5), 29-37.

Olson, J. (1988). Schoolworlds/microworlds: Computers and the culture of the classroom. Oxford: Pergamon Press.

Olson, J. (1992). Understanding teaching. Philadelphia: Milton Keynes - Open University Press.

Orhun, E. (1995). Design of computer-based cognitive tools. In A. A. diSessa, C. Hoyles, & R. Noss (Eds.), Computers and exploratory learning (pp. 305-319). London: Springer.

Potashnik, M. (1996). Chile's learning network. Education and Technology Series 1 (2). Washington, D.C.: The World Bank.

Potashnik, M., Rawlings, L., Means, B., Alvarez, M. I., Roman, F., Dobles, M. C., Umaña, J., Zúñiga, M., & García, J. (1998). Computers in Schools: A qualitative study of Chile and Costa Rica. Education and Technology Series: Special Issue. Washington, D.C.: The World Bank.

Potts, C. (1993). A software engineering research revisited. IEEE Software(September), 19-28.

Reason, P. (1994). Three approaches to participative inquiry. In N. K. Denzin & Y. S. Lincoln (Eds.), Handbook of qualitative research (pp. 324-339). London: Sage.

Reeves, B., & Nass, C. (1996). The media equation: How people treat computers, television, and new media like real people and places. Cambridge: Cambridge University Press.

Reiser, R. A., & Dick, W. (1990). Evaluating instructional software. Educational Technology Research and Development, 38(3), 43-50.

Reiser, R. A., & Kegelman, H. W. (1996). Computer software evaluation. In T. Plomp & D. E. Ely (Eds.), International encyclopedia of educational technology (pp. 257-260) Oxford: Elsevier Science - Pelgrum.

Reusser, K. (1993). Tutoring systems and pedagogical theory: Representation tools for understanding, planning and reflection in problem solving. In S. P. Lajoie & S. J. Derry (Eds.), Computers as cognitive tools (pp. 143-177). Hillsdale: Lawrence Erlbaum.

Sammons, P., Hillman, J., & Mortimore, P. (1995). Key characteristics of effective schools: A review of school effectiveness research. London: Institute of Education, University of London.

Sandholtz, H. J., Ringstaff, C., & Dwyer, D. C. (1997). Teaching with technology: Creating student centered classrooms. New York: Teachers College Press.

Schofield, J. W. (1995). Computers and classroom culture. New York: Cambridge University Press.

Schön, D. (1983). The reflective practitioner. New York: Basic Books.

Schwartz, J. L. (1996). Motion toys for eye and mind. Communication of the ACM, 39(8), 94-96.

Schwartz, J. L., Yerushalmy, M., & Wilson, B. (Eds.). (1993). The geometric supposer: What is the case of? London: Erlbaum.

Self, J. (1985). Microcomputers in education: A critical appraisal of educational software. Brighton: The Harvest Press. Ltd.

Soloway, E., & Pryor, A. (1996). Using computational media to facilitate learning. Communications of the ACM, 39(8), 83-109.

Sommerville, I. (1989). Software engineering (3rd ed.). New York: Addison Wesley.

Squires, D. (1996). Production of educational software. In T. Plomp & D. E. Ely (Eds.), International encyclopedia of educational technology (pp. 217-221). Oxford: Elsevier Science - Pelgrum.

Squires, D., & McDougall, A. (1994). Choosing and using educational software: A teachers' guide. London: The Falmer Press.

Squires, D., & Preece, J. (1996). Usability and learning: evaluating the potential of educational software. Computers and Education, 27(1), 15-22.

Stake, R. (1994). Case studies. In N. Denzin & Y. Lincoln (Eds.), Handbook of qualitative research (pp. 236-247). London: Sage.

Stake, R. E., & Trumbull, D. J. (1983). Naturalistic generalisations. Review Journal of Philosophy and Social Science(7), 1-12.

Suchman, L. A. (1987). Plans and situated actions: The problem of human machine communication. Cambridge: Cambridge University Press.

Taylor, R. P. (Ed.). (1980). The Computer in the School: Tutor, Tool, Tutee. New York: Teacher College Press.

Thimbleby, H. (1990). User Interface Design. New York: Addison Wesley.

Tolhurst, D. (1992). A checklist for evaluating content-based hypertext computer software. Educational Technology(March), 17-21.

Watson, D. (1987). Developing CAL: Computers in the curriculum. London: Harper & Row Ltd.

Watson, D. M. (1990). The classroom vs. the computer room. Computers in education, 15(1-3), 33-37.

Watson, L. (1993). Appropriate tools ? IT in the primary classroom. In J. Beynon & H. Mackay (Eds.), Computers into classroom: More questions than answers (pp. 78-91). London: The Falmer Press.

Wegerif, R., & Mercer, N. (1997). Using computer-based text analysis to integrate qualitative and quantitative methods in research on collaborative learning. Language and Education, 11(4), 271-286.

Wertsch, J. V. (1985). Vygotsky and the social formation of mind. London: Harvard University Press.

Willis, J., & Mehlinger, H. (1996). Information technology and teacher education. In J. Sikula (Ed.), Handbook of research on teacher education (pp. 978-1029). New York: Macmillan.

Wilson, B., & Cole, P. (1991). A review of cognitive teaching models. Educational Technology Research and Development, 39(4), 47-64.

Winograd, T. (1995). From programming environments to environments for design. Communications of the ACM, 38(6), 65-74.

Winograd, T., & Flores, F. (1986). Understanding computers and cognition: A new foundation for design. New York: Addison-Wesley Pub. Co.

Winship, J. A. (1989). Information technology in education: the quest for quality software. Paris: Organisation for Economic Co-operation and development.

Woods, P., & Jeffrey, B. (1996). Teachable moments: The art of teaching in primary classroom. Buckingham: Open University Press.

Wright, A. (1987). The process of microtechnological innovation in two primary schools: A case study of teachers' thinking. Educational Review, 39(2), 107-115.

Yin, R. (1994). Case study research design and methods (2nd ed.). London: Sage.

Yinger, R. J., & Hendriks-Lee, M. S. (1995). Teacher planning. In L. W. Anderson (Ed.), International encyclopedia of teaching and teacher education (pp. 188-192). Oxford: Pergamon.

Zahner, J. E., Reiser, R. A., Dick, W., & Gill, B. (1992). Evaluating instructional software: A simplified model. Educational Technology Research and Development, 40(3), 55-62.

Zuboff, S. (1988). In the age of the smart machine: The future of work and power. Oxford: Heineman Prof. Pub.

X APPENDIX

A.1. INTERVIEW TO PRE-SELECTED SCHOOLS

These unstructured interviews were carried out in March 1996 and the following report was a personal communication to the supervisor.

a) Standard

School name: Standard
Participants:
Head of school: Ana Rieu and
Vice head: María Angélica Quintana
School population: 430 students.
e-mail: coordinador@standard.tco.plaza.cl

I told them about the research project and that I was looking for a piece of software that could support in a better way what they do in the classroom as teachers. I mentioned action research and explain them why it was useful.

They asked some things in order to understand better what I was telling and how many teachers should be involved.

They agreed with the need of better software design that could be used in the classroom.

The first reaction was that the project seems very interesting, they were thoughtful about the time it could involve and asked, laughing, what they would win with the project. I told them the piece of software and knowledge.

Afterwards, we spoke about innovation:

This school is seen as an innovative school, they have about 10 different projects running now, six related with computers (like involving the family of the students with the school reinforcing parent-child communication) and others like a radio station for the school.

When I asked them about the reasons that could make the school 'different' from others in terms of innovation they answered that they make participate the colleagues in the decision process and that the working environment (organisational climate) was

very good. That the colleagues do not reject what is being said from the 'upper level' (the head) because they are all friends. They do a lot of social activities together (birthdays celebrations, etc.). They make the colleagues fill engage with the new projects. In order to produce this engagement they 'create' workshops and activities in which the teachers are trained in new methods. They keep the colleagues informed of each project of the school.

In summary they told me that the two key elements for their successful innovations are: the friendship relations they have and that they make them (all colleagues) participate. In this sense, colleagues always believe that they are deciding the projects and innovations, without realising that the projects were defined by the director. That means that the innovation starts from a group of 3-4 teachers (head, vice-head and other) and the rest of the colleagues are 'induced' to participate.

Not all the teachers of the school participate actively in these innovations, because some of them are about to retire.

They say that the computers are not enough for the schools.

I asked them if the computers made any difference in terms of the projects they do. They told me that the time before they had computers, projects were mostly extra curricular activities (singing groups, etc.) but that the colleagues participated too. And when Enlaces appeared they started to do projects related with the curriculum and afterwards they started with Educational Improvement Projects [26]. The first project they had was Enlaces.

They say hat the project was effective because the students now are able to organise themselves to carry on activities (like playing a piece of theatre or doing a radio program). This is one way in which they evaluate their projects.

They use the computer to produce written material and also like a tool. This type of use, as a tool, is the emphasis of this year. The idea is that all the courses rotate using the computer room and browsing through the different pieces of software they have available and that in this process the teacher can see (evaluate) the material (s)he has available and will be able to design activities using the computer. Also, there will be activities with the teachers alone looking for material that could be useful for their teaching.

[26] This is the other important motivation for this new project oriented activities called the PME initiative (Education Improvement Project) of the Chilean MECE program. Through this initiative each school can apply for funds to carry on some specific project.

I asked them if these projects and innovations are also transferred to the classroom (teaching). They answered that the teachers in the school are realising that if they want to have a good discipline and motivated students they need to innovate, to do new things. That if the students are interested in the activities they do not interrupt the class.

One concrete effect of this is that students do not want to miss a class, because they will miss an activity. And if they are working in groups the group will suffer the absence and that is important.

They are doing a lot of group learning using collaborative projects. They are trying to induce all the colleagues to do collaborative work, facilitating materials and giving personal assistance (from the head) to each teacher.

The idea is that each student should be working because if not (s)he will bother the teacher.

They say that the colleagues have no fear to innovate. One task for the second half of the year for each teacher will be that each course designs a project, simple but including all the students and that the project incorporates the use of computers, visits and other resources they have (radio station, etc.). The idea behind is that the teachers design something that changes the traditional activities, because any change produces enthusiasm in the students.

I asked them why the emphasis in keeping the student s working and not bothering the teacher (referred to the difference with other schools that emphasise the learning and/or teaching methods).

They answered that they think that when the student is engaged with the activity (s)he is learning and that their students are very difficult.

This difficulty is because most of the social (cultural) level of the students. Most of the parents are divorced and have very low educational level. They told me some stories about aggressions from the students to teachers, use of drugs, students that are almost criminals, and suffer of home violence, etc.

This situation influences the vocation of the teachers and changes their behaviour, being more tolerant. To make this point more clear: when they ask the students why are they in this school, they answer because I like the school, and if they ask what do

you dislike of the school they answer the toilets, etc. And when they ask what do you like, they answer the teachers.

They want to build up useful citizens and good 'persons', they are not focused in the contents of the curriculum.

b) Vista Verde
(in English: Green View)

School name: Vista Verde
Participants:
Head of School: Amanda Deramond
School's population: 421 students in 1995.
e-mail: coordinador@vverde.tco.plaza.cl

I told her about the research project and that I was looking for a piece of software that could support in a better way what the teacher does in the classroom. I mentioned action research and the process of software development we would follow. I described the need of designing more appropriate pieces of software that support better the activities of the teacher.

She agreed with me and seemed to understand what I was looking for.

Her first reaction was that this is a new challenge and she explained to me that there is a lack of knowledge within the teachers about research methodologies. So this project would be very useful in this sense. And she said that if we could schedule the meetings in a way that they fit into the actual activities, not demanding extra time from the teachers it would be no problem.

We talked about the possible teachers that could participate in the project and designed a possible schedule. We left the decision to the possible candidates (two female rather young teachers that were students in the diploma course of Enlaces).

In order to speak about innovation I asked her about the projects they are carrying on.

They have two main projects:

- One Education Improvement Project for grade 1 to 8. The aim of the project is to improve written, oral and corporal expression of the students trough activities that aloud the to know all about the ninth region (political division of the Chilean

territory, equivalent to a county), including cultural, artistic environmental dimensions, among other. The activities planned are small.

- The other project is related to Enlaces and is focused on students from grade 1 to 3.

They also have small projects in each course, some of them with the parents, self esteem, involving the parents, etc.

I asked, how do you 'propagate' the projects among the other teachers ?

She told me that last year (1995) not all the teachers were involved in the projects (especially Enlaces), there was a group of 6 teachers that were very enthusiastic and did all the work with the computers. This year they left the responsibility of designing and writing the new project for Enlaces to the teachers that were not participating in it. So, now they fill owner of the new projects and they started to work in it enthusiastically.

With this strategy a 100% of the teachers are now involved in the projects (I think she meant only the Enlaces project).

Following the innovations, they are changing also the classrooms' layout, trying that all the students do collaborative projects in groups.

I ask her how did they train the group of teachers that were not 'innovators', and she told me that they do technical meetings once a week sharing experiences.

I asked if the computer had a special role in this process, she answered that the computer played the role of a challenge for the teachers. And now they transfer new methods from the computer lab to the classroom and from the classroom to the computer lab.

About the future, she told that in the past it was impossible to think about what they have now, and in the future they want to involve more the parents with the school. For example teaching the parents to read and write using the computer.

About the organisational climate, she said that they always have had a very good relation, there is no explicit authority. She thinks that the project will be useful to keep the colleagues together. But she mentioned that it is a very sensible area, that she must be very careful about the human relations. There is no one that owns the

computer lab, on the contrary, she said that they are reinforcing the autonomy of each teacher in the use of the computer. Each teacher is responsible for her time in the lab.

They are starting to use international communication with e-mail.
(I promised her to find schools in France to have key pals with the school.)

Comments:

Both schools are candidates to be part of the project.

In order to decide which one I could say:

- Standard school has a self concept of a very innovative school, but the way in which they 'propagate' the innovations is rather tricky. A small group decides the characteristics of the project and then they induce the rest of the colleagues to make it. On the other hand, Vista Verde has only a small group of teachers that are willing to innovate, but it seems to me that the strategies they use to involve other colleagues is more 'open'.

- I got the idea that in the Standard school there is more authoritarian present, in spite of what they say about friendship. I think the rest of the colleagues have a clear picture of who is the boss. In the Vista Verde, it seems that apparently the head plays her role of being the boss more straight forward, but she plays to be a reasonable and accessible boss.

- Some additional evidence is that the Vista Verde school showed immediately the disposition to participate and the head organised meetings with the candidates in order to ask them if they agree to be part of it. The Standard school was willing to participate, but they had to ask the teachers first and discuss this with other colleagues.

- The focus on the research training of Vista Verde worried me a little bit, but afterwards it became a low priority.

- In other aspects, both schools offer advantages and disadvantages, but I do not see another point that could help in the decision.

In this sense, it would be easier to work with Vista Verde, because of the organisational climate and the way in which they perceived the project. I have the felling that I would be better working with Vista Verde. So, if you have no objections

I would decide to work with Vista Verde. I need to tell the schools early next week (Wednesday the 17th).

I plan to start with the process the week of the 15th of April 1996.

A.2 PROBABILITIES OF UNITS AND SEQUENCES

The probability of a sequence (called P(Seq)) was defined as the multiplication of the probability of each unit that is included in the sequence. In the case that the units included in the sequence are composed by more than one category (for example 'Actions' has 6 categories, 'Aim' has four and 'Browsing' has 3), the probability of the unit was calculated as the addition of the individual categories. For example:

P(Aim) = P(Aim_Teacher_Software) + P(Aim_Teacher_Computer) + P(Aim_Pupil_Software) + P(Aim_Pupil_Computer)

or

P(Browsing) = P(Browsing_Abstract) + P(Browsing_Medium) + P(Browsing_Concrete)

and if the sequence found were composed by 'Aim' and 'Browsing', then its probability would be:

P(Seq) = P(Aim) * P(Browsing)

The probability of each category (e.g. P(Aim_Pupil_Software), P(Aim_Teacher_Computer), etc.) is calculated dividing the number of units found of that category by the total units spoken by each member of the development team and multiplied as explained before. The next table shows the probability of each group of categories:

	TE	TM	PY	SE	GD	Group
Actions	0.0937	0.0684	0.0549	0.0608	0.0049	0.0637
Aim	0.0648	0.0456	0.0428	0.0268	0.0000	0.0404
Browsing	0.1138	0.1027	0.1172	0.1397	0.0574	0.1136
Interface Element	0.2144	0.2664	0.2086	0.2330	0.5482	0.2593
Interaction	0.1340	0.2481	0.1822	0.2686	0.2833	0.2194
Content Org.	0.1271	0.0994	0.1404	0.1025	0.0672	0.1114
Subject Areas	0.1021	0.0729	0.1040	0.0612	0.0122	0.0770
Teaching Strategy	0.1070	0.0584	0.1103	0.0893	0.0244	0.0839
User	0.0432	0.0381	0.0396	0.0180	0.0024	0.0313

Table A.2.1 Probabilities of one unit of each category for each MDT

In table A.2.1 it is possible to see that in the given set of data the probability of finding a unit of analysis of , for example, Teacher E (TE) talking about 'Actions' is equal to: P(Actions) = 0.0937 and that the probability of finding a unit of analysis of the Software Engineer talking about 'Browsing' is equal to: P(Browsing) = 0.1397.

As explained, the probability of each sequence of units was calculated as the multiplication of the composing units. In the case of the 2-units long sequences, the probabilities are:

Sequences involving 2 different units	TE	TM	PY	SE	GD	Group	Total*
Interface Element and Interaction	0.0287	0.0661	0.0380	0.0626	0.1553	0.0569	0.3507
Browsing and Interface Element	0.0244	0.0274	0.0244	0.0326	0.0315	0.0295	0.1402
Interaction and Actions	0.0126	0.0170	0.0100	0.0163	0.0014	0.0140	0.0572
Browsing - Teaching Strategy	0.0122	0.0060	0.0129	0.0125	0.0014	0.0095	0.0450
Subject Areas and Content Org.	0.0130	0.0072	0.0146	0.0063	0.0008	0.0086	0.0419
Browsing and Content Org.	0.0145	0.0102	0.0165	0.0143	0.0039	0.0127	0.0593
Content Org. and Interface Element	0.0273	0.0265	0.0293	0.0239	0.0368	0.0289	0.1437
Content Org. and Interaction	0.0170	0.0247	0.0256	0.0275	0.0190	0.0244	0.1138
Aim - User	0.0028	0.0017	0.0017	0.0005	0.0000	0.0013	0.0067
Actions	0.0088	0.0047	0.0030	0.0037	0.0000	0.0041	0.0202
Prob. Of speaking any of the sequences	0.1612	0.1914	0.1760	0.2001	0.2501	0.1897	0.9788

*Total represents the probability of sequences spoken by any of the members of the development team, so it is the addition of these probabilities.

Table A.2.2 Probabilities of each two unit long sequence for each MDT

In table A.2.2 it is possible to see that the probability of finding a sequence combining, for example, units of the groups of categories 'Interface Element' and 'Interaction', spoken by TE is P=0.0287.

The distribution of probabilities of the three-units long sequences are presented in table A.2.3:

Seq. involving 3 Units	TE	TM	PY	SE	GD	Group	Tot*
Interaction, Subject Areas and Actions	0.0013	0.0012	0.0010	0.0010	0.0000	0.0011	0.0046
Browsing, Subject Areas and Content Org.	0.0018	0.0024	0.0028	0.0038	0.0011	0.0027	0.0119
Subject Areas, Interface Element and Interaction	0.0029	0.0048	0.0040	0.0038	0.0019	0.0044	0.0174
Interface Element, Subject Areas and Content Org.	0.0028	0.0019	0.0030	0.0015	0.0004	0.0022	0.0097
Interaction, Aim and Teaching Strategy	0.0009	0.0007	0.0009	0.0006	0.0000	0.0007	0.0031
Content Org., Aim and Actions	0.0008	0.0003	0.0003	0.0002	0.0000	0.0003	0.0016
Prob. of speaking any of the sequences	0,0105	0,0114	0,0120	0,0109	0,0034	0,0114	0,0483

*Total represents the probability of sequences spoken by any of the members of the development team, so it is the addition of these probabilities.

Table A.2.3 Probabilities of each three unit long sequence for each MDT

Due that the sequences found were combinations of repeated units (for example, 'Interaction'-'Browsing'-'Browsing'-'Interaction' for two units long sequences),the probability calculated for each sequence is the upper limit (maximum), because, if another unit is added, it should be the multiplication of the three unit's probabilities. Nevertheless, because of the consistency of the findings independently of the length of the sequence (table 6.2 and table 6.4), this factor should not interfere with the analysis.

- **Expected probability**

In order to be able to analyse these data, it is necessary to define a degree of meaningfulness, that is, a range in which we can consider a probability to be high, low or expected. In order to do this, it can be assumed that in the ideal case, all participants had the same chance to speak (uniform distribution) about any category. Starting from this premise, the average participation during the development process was 215 units per category, considering all the groups of categories defined for the sequences and the average number of units spoken by each member was 1933 (total number of units divided by five: 9663/5). So, the probability of an individual category for any member of the development team can be calculated dividing the average number of units spoken in each category by the average units spoken by each member, that is 215/1933= 0.111. This is, the expected probability to find any unit spoken by any member of the development team is **P(one unit)=0.111.** Then, the probability of repeating a unit two times is **P(two unit sequence)=0.0123** and three times is **P(three unit sequence)=0.0014**. Also, the probability of a sequence spoken by any of the members of the development team is five times the probability of a two units sequence **P(any member two unit sequence)= 0.0617** for two-units long sequences and **P(any member three unit sequence)=0.0069** for three-units long sequences. This number were used to define the limit of meaningfulness of the sequences found. Summarising:

- One unit: P(to find any unit spoken) = 0.111
  P(to find any unit spoken by any MDT) = 0.555
- 2 units-long sequences: P(to find any two units spoken by one MDT) = 0.0123
  P(to find any two units spoken by any MDT) = 0.0617
- 3 units long sequences: P(to find any three units spoken by one MDT)= 0.0014
  P(to find any three units spoken by any MDT)= 0.0069

- **Meaningful sequences**

Based on the previous calculation, the meaningful two unit long sequences found for each MDT ($p<0.0123$) and for any member of the development team ($p<0.0617$) are presented in table A.2.4.

The meaningful three unit long sequences found by each member of the development team ($p< 0.0014$) and of any member of the development team ($p<0.0069$) are presented in table A.2.5.

Sequences of 2 different units	TE	TM	PY	SE	GD	Group	Total*
Interface Element and Interaction	0.0287	0.0661	0.0380	0.0626	0.1553	0.0569	0.3507
Browsing and Interface Element	0.0244	0.0274	0.0244	0.0326	0.0315	0.0295	0.1402
Interaction and Actions	0.0126	0.0170	0.0100	0.0163	0.0014	0.0140	0.0572
Browsing – Teaching Strategy	0.0122	0.0060	0.0129	0.0125	0.0014	0.0095	0.0450
Subject Areas and Content Org.	0.0130	0.0072	0.0146	0.0063	0.0008	0.0086	0.0419
Browsing and Content Org.	0.0145	0.0102	0.0165	0.0143	0.0039	0.0127	0.0593
Content Org. and Interface Element	0.0273	0.0265	0.0293	0.0239	0.0368	0.0289	0.1437
Content Org. and Interaction	0.0170	0.0247	0.0256	0.0275	0.0190	0.0244	0.1138
Aim - User	0.0028	0.0017	0.0017	0.0005	0.0000	0.0013	0.0067
Actions	0.0088	0.0047	0.0030	0.0037	0.0000	0.0041	0.0202
Prob. of speaking any of the sequences	0.1612	0.1914	0.1760	0.2001	0.2501	0.1897	0.9788

*Total represents the probability of sequences spoken by any of the members of the development team, so it is the addition of these probabilities.

Table A.2.4 Meaningful probabilities of two unit long sequence for each MDT

equences involving 3 Units	TE	TM	PY	SE	GD	Group	Total*
nteraction, Subject Areas and Actions	0.0013	0.0012	0.0010	0.0010	0.0000	0.0011	0.0046
rowsing, Subject Areas and Content Org.	0.0018	0.0024	0.0028	0.0038	0.0011	0.0027	0.0119
ubject, Interface Element. and Interaction	0.0029	0.0048	0.0040	0.0038	0.0019	0.0044	0.0174
nterface Element, Subject Areas and Content Org.	0.0028	0.0019	0.0030	0.0015	0.0004	0.0022	0.0097
nteraction, Aim and Teaching Strategy	0.0009	0.0007	0.0009	0.0006	0.0000	0.0007	0.0031
rganisation, Aim and Actions	0.0008	0.0003	0.0003	0.0002	0.0000	0.0003	0.0016
rob. of speaking any sequence	0,0105	0,0114	0,0120	0,0109	0,0034	0,0114	0,0483

*Total represents the probability of sequences spoken by any of the members of the development team, so it is the addition of these probabilities.

Table A.2.5 Meaningful probabilities of three unit long sequence for each MDT

A.3 LIST OF SEQUENCES FOUND

This appendix presents a detailed list of the meaningful sequences found in each category. In order to present useful information (and avoiding endless lists), compacted repeated units in each sequence. That is, if a sequence had the form:

'aabccf'

it was transformed into:

'abcf'

In this way it was possible to analyse the relation between two different units. Sequences obtained this way that presented less than 10 occurrences were eliminated. This is the reason why the totals do not coincide with the previous tables.

A.3.1 Two unit sequences

In this section, for each sequence the specific units found in each data set is presented.

Charact. of Software: Interaction and Action	Group	PY	TM	TE	SE	**Total**
CIn_MedCIn_ConcAc_P_I_S	0	0	26	15	23	**64**
CIn_ConcAc_P_I_SCIn_Med	0	0	23	15	18	**56**
Ac_P_I_SCIn_MedCIn_Conc	2	0	22	14	17	**55**
CIn_MedCIn_ConcAc_P_I_SCIn_Med	0	0	22	14	17	**53**
Ac_P_I_SCIn_MedCIn_ConcAc_P_I_S	0	0	21	13	16	**50**
Ac_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_Med	0	0	21	13	16	**50**
CIn_ConcAc_P_I_SCIn_MedCIn_Conc	0	0	21	13	16	**50**
CIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_S	0	0	21	13	16	**50**
CIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_Med	0	0	21	13	16	**50**
CIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_Conc	0	0	21	13	16	**50**
CIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_S	0	0	21	13	16	**50**
Ac_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_Conc	0	0	20	12	15	**47**
CIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_Med	0	0	20	12	15	**47**
Ac_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_S	0	0	19	11	14	**44**
Ac_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_Med	0	0	19	11	14	**44**
CIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_Conc	0	0	19	11	14	**44**
CIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_S	0	0	19	11	14	**44**
CIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_Med	0	0	19	11	14	**44**
CIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_Conc	0	0	19	11	14	**44**
CIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_S	0	0	19	11	14	**44**
Ac_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_ConcAc_P_I_SCIn_MedCIn_Conc	0	0	18	10	13	**41**
Total	**2**	**0**	**431**	**260**	**328**	**1021**

Table A.3.1. Groups of categories 'Interaction' and 'Actions'

Here we can see that almost all of the actions in this sequence are related to the use of software by the pupil individually. This fact is interesting and shows that they were designing what the user should do in the classroom without involving the teacher as

user of the software. The Software Engineer, Teacher M and Teacher ME combine the deign of actions and the design of interactions with the software at a medium and concrete level of abstraction.

haract. of Software: Browsing and Teaching Strategy	Group	PY.	TM	TE	SE	**Total**
Br_ConcTStr_Fee	2	25	44	45	55	**171**
Br_ConcTStr_FeeCBr_Conc	0	10	28	20	53	**111**
Str_FeeCBr_Conc	0	24	22	19	41	**106**
Br_ConcTStr_FeeCBr_ConcTStr_Fee	0	7	17	14	49	**87**
Str_FeeCBr_ConcTStr_Fee	0	5	19	17	45	**86**
Str_FeeCBr_ConcTStr_FeeCBr_Conc	0	4	22	20	24	**70**
Br_ConcTStr_FeeCBr_ConcTStr_FeeCBr_Conc	0	3	21	14	27	**65**
Str_FeeCBr_ConcTStr_FeeCBr_ConcTStr_Fee	0	5	12	14	16	**47**
Str_FeeCBr_ConcTStr_FeeCBr_ConcTStr_FeeCBr_Conc	0	0	17	12	15	**44**
Br_ConcTStr_FeeCBr_ConcTStr_FeeCBr_ConcTStr_FeeCBr_Con	0	0	16	10	15	**41**
Br_ConcTStr_FeeCBr_ConcTStr_FeeCBr_ConcTStr_Fee	0	0	12	6	15	**33**
Str_FeeCBr_ConcTStr_FeeCBr_ConcTStr_FeeCBr_ConcTStr_Fee	0	0	8	4	11	**23**
Str_FeeCBr_ConcTStr_FeeCBr_ConcTStr_FeeCBr_ConcTStr_Fee Br_Conc	0	0	8	6	9	**23**
Br_MedTStr_Fee	0	2	8	7	2	**19**
Br_ConcTStr_FeeCBr_ConcTStr_FeeCBr_ConcTStr_FeeCBr_Con TStr_Fee	0	0	5	4	7	**16**
Br_ConcTStr_FeeCBr_ConcTStr_FeeCBr_ConcTStr_FeeCBr_Con TStr_FeeCBr_Conc	0	0	5	4	3	**12**
Br_MedTStr_FeeCBr_Med	0	2	3	2	5	**12**
Str_FeeCBr_ConcTStr_FeeCBr_ConcTStr_FeeCBr_ConcTStr_Fee Br_ConcTStr_Fee	0	0	4	2	5	**11**
otal	**2**	**87**	**271**	**220**	**397**	**977**

Table A.3.2. Groups of categories 'Browsing' and 'Teaching Strategy'

The most frequent sequences relate browsing at a concrete level of abstraction and the teaching strategy, therefore all member of the development team had the same combinations of units as their most frequent ones. The cycles of combinations of units here were rather long, involving seven iterations between browsing at a concrete level of abstraction and teaching strategy.

Charact. of Software: Subject and Organisation	Group	PY.	TM	TE	SE	Total
CSu_MedCOr_MedCSu_Med	0	7	8	10	18	**43**
COr_MedCSu_MedCOr_Med	0	7	8	10	17	**42**
COr_AbsCSu_AbsCOr_Abs	2	6	12	6	15	**41**
CSu_MedCOr_MedCSu_MedCOr_Med	0	7	7	10	17	**41**
CSu_AbsCOr_AbsCSu_Abs	0	6	12	6	15	**39**
CSu_AbsCOr_AbsCSu_AbsCOr_Abs	0	6	12	6	15	**39**
CSu_MedCOr_Abs	0	18	18	0	2	**38**
COr_MedCSu_MedCOr_MedCSu_Med	0	6	7	7	14	**34**
CSu_AbsCOr_Abs	0	13	0	11	10	**34**
COr_AbsCSu_AbsCOr_AbsCSu_Abs	0	5	9	5	14	**33**
COr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_Abs	0	4	8	4	14	**30**
CSu_MedCOr_MedCSu_MedCOr_MedCSu_Med	0	5	6	6	13	**30**
COr_MedCSu_MedCOr_MedCSu_MedCOr_Med	0	5	6	6	12	**29**
CSu_MedCOr_MedCSu_MedCOr_MedCSu_MedCOr_Med	0	5	6	6	12	**29**
CSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_Abs	0	4	8	4	11	**27**
CSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_Abs	0	4	8	4	11	**27**
COr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_Abs	0	3	7	3	10	**23**
COr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_Abs	0	2	6	2	9	**19**
CSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_Abs	0	2	6	2	9	**19**
CSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_Abs	0	2	6	2	9	**19**
COr_MedCSu_MedCOr_MedCSu_MedCOr_MedCSu_Med	0	4	2	3	9	**18**
COr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_Abs	0	0	5	0	8	**13**
CSu_MedCOr_MedCSu_MedCOr_MedCSu_MedCOr_MedCSu_Med	0	3	0	2	8	**13**
COr_MedCSu_MedCOr_MedCSu_MedCOr_MedCSu_MedCOr_Med	0	3	0	2	7	**12**
CSu_MedCOr_MedCSu_MedCOr_MedCSu_MedCOr_MedCSu_MedCOr_Med	0	3	0	2	7	**12**
COr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_Abs	0	0	4	0	7	**11**
CSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_AbsCOr_AbsCSu_Abs	0	0	4	0	7	**11**
Total	**2**	**130**	**175**	**119**	**300**	**726**

Table A.3.3. Groups of categories 'Subject Areas' and 'Content Organisation'

They combine units at an abstract level or at a medium level of abstraction, without mixing these levels too much. The combination of these units is repeated in long cycles again. That is, they talked about the subject areas, then about the way to organise the content and then again about the subject areas, and so on.

Charact. of Software: Browsing and Organisation	Group	PY.	TM	TE	SE	Total
CBr_MedCOr_Med	2	82	101	25	78	**288**
COr_MedCBr_Med	0	68	68	22	34	**192**
CBr_ConcCOr_Conc	0	14	10	13	24	**61**
CBr_MedCOr_MedCBr_Med	4	3	0	4	2	**13**
COr_MedCBr_MedCOr_Med	0	4	4	4	0	**12**
COr_MedCOr_ConcCBr_Med	0	6	0	3	3	**12**
CBr_AbsCOr_Abs	0	2	4	0	4	**10**
COr_ConcCBr_MedCBr_Conc	0	5	0	3	2	**10**
COr_MedCOr_ConcCBr_MedCBr_Conc	0	5	0	3	2	**10**
Total	**6**	**189**	**187**	**77**	**149**	**608**

Table A.3.4. Groups of categories 'Browsing' and 'Content Organisation'

They combined these units at a medium level of abstraction. The cycles were short, that is, they talked about browsing and then about the organisation of the contents and then they stopped.

Charact. of Software: Organisation and Interaction	Group	PY.	TM	TE	SE	Total
COr_ConcCIn_Abs	0	0	14	0	14	**28**
COr_ConcCIn_MedCIn_Conc	0	0	8	4	6	**18**
COr_MedCOr_ConcCIn_Med	0	0	8	4	6	**18**
COr_MedCOr_ConcCIn_MedCIn_Conc	0	0	8	4	6	**18**
CIn_MedCOr_Med	0	0	0	8	9	**17**
COr_ConcCIn_ConcCOr_Conc	0	4	6	0	5	**15**
CIn_ConcCOr_Conc	3	5	2	0	3	**13**
CIn_ConcCOr_MedCOr_Conc	0	0	7	2	4	**13**
CIn_ConcCOr_MedCOr_ConcCIn_Med	0	0	7	2	4	**13**
CIn_MedCIn_ConcCOr_Med	0	0	7	2	4	**13**
CIn_MedCIn_ConcCOr_MedCOr_Conc	0	0	7	2	4	**13**
COr_ConcCIn_MedCIn_ConcCOr_Med	0	0	7	2	4	**13**
COr_ConcCIn_Conc	0	4	2	0	4	**10**
Total	**3**	**13**	**83**	**30**	**73**	**202**

Table A.3.5. Groups of categories 'Content Organisation' and 'Interaction'

Teacher M and the Software Engineer combined the design of the organisation of the contents at a concrete level of abstraction with the design of the interaction with the software at an abstract level of abstraction. Teacher ME combined these units mostly at a medium level of abstraction.

Aim-User	Group	PY.	TM	TE	SE	Total
User_PAims_P_S	2	16	16	0	0	**34**
Aims_T_SUser_PAims_T_S	0	6	9	7	4	**26**
User_PAims_T_SUser_P	0	7	9	7	3	**26**
Aims_T_SUser_PAims_T_SUser_P	0	6	8	7	3	**24**
User_PAims_T_SUser_PAims_T_S	0	5	7	6	2	**20**
Aims_T_SUser_PAims_T_SUser_PAims_T_S	0	4	5	5	2	**16**
Aims_T_SUser_PAims_T_SUser_PAims_T_S User_P	0	4	5	5	0	**14**
User_PAims_T_SUser_PAims_T_SUser_P	0	4	5	5	0	**14**
User_PAims_T_SUser_PAims_T_SUser_PAi ms_T_S	0	3	4	4	0	**11**
Total	**2**	**55**	**68**	**46**	**14**	**185**

Table A.3.6. Groups of categories 'Aim' and 'User'

In table A.3.6, it can be seen that they combined the design of the aims of the software for the pupil with the description of the pupil as user (particularly Teacher M and the Psychologist). It is interesting to see the long cycles alternating the design of the aims of the software for the teacher (Aims_T_S) with descriptions of the pupil as user.

Actions	Grou p	PY.	TM	TE	SE	Total
Ac_P_I_PAc_P_C_PAc_P_I_P	0	0	5	11	4	**20**
Ac_P_C_PAc_P_I_PAc_P_C_P	0	0	5	10	3	**18**
Ac_P_C_PAc_P_I_PAc_P_C_PAc_P_I_P	0	0	5	10	3	**18**
Ac_P_I_PAc_P_C_PAc_P_I_PAc_P_C_P	0	0	3	7	3	**13**
Ac_P_I_PAc_P_C_PAc_P_I_PAc_P_C_PAc_P_I_P	0	0	3	7	3	**13**
Ac_P_C_PAc_P_I_PAc_P_C_PAc_P_I_PAc_P_C_P	0	0	3	6	2	**11**
Ac_P_C_PAc_P_I_PAc_P_C_PAc_P_I_PAc_P_C_PAc_ P_I_P	0	0	3	6	2	**11**
Total	**0**	**0**	**27**	**57**	**20**	**104**

Table A.3.7. Group of categories 'Actions'

It is interesting to see that they combined the design of actions that involved what pupils would do with other pupils in the classroom (Ac_P_C_P) and what they would do individually (Ac_P_I_P).

A.3.2 Three unit sequences

This section presents the meaningful sequences composed by three units.

Charact. of Software: Interaction, Subject and Action	Group	PY.	TM	TE	SE	Total
Ac_P_I_SCSu_MedCSu_ConcCIn_Med	0	0	3	0	4	7
Ac_P_I_SCSu_MedCSu_ConcCIn_MedCIn_Conc	0	0	3	0	4	7
CSu_ConcCIn_MedCIn_ConcAc_P_I_S	0	0	3	0	4	7
CSu_MedCSu_ConcCIn_MedCIn_ConcAc_P_I_S	0	0	3	0	4	7
Ac_P_I_SCSu_MedCSu_ConcCIn_MedCIn_Conc Ac_P_I_S	0	0	2	0	3	5
CIn_ConcAc_P_I_SCSu_Med	0	0	2	0	3	5
CIn_ConcAc_P_I_SCSu_MedCSu_Conc	0	0	2	0	3	5
CIn_ConcAc_P_I_SCSu_MedCSu_ConcCIn_Med	0	0	2	0	3	5
CIn_MedCIn_ConcAc_P_I_SCSu_Med	0	0	2	0	3	5
CIn_MedCIn_ConcAc_P_I_SCSu_MedCSu_Conc	0	0	2	0	3	5
CSu_ConcCIn_MedCIn_ConcAc_P_I_SCSu_Med	0	0	2	0	3	5
Total	**0**	**0**	**26**	**0**	**37**	**63**

Table A.3.8. Groups of categories 'Interaction', 'Subject Areas' and 'Actions'

The actions that they designed in this sequences involved only the pupils using the software individually (AC_P_I_S), similar to the sequence 'Action-Interaction' described earlier.

Charact. of Software: Interaction, Aims and Teaching Strgy	Group	PY.	TM	TE	SE	Total
TStr_FeeCIn_MedAims_P_C	0	2	0	0	2	4
CIn_MedAims_P_CTStr_Fee	0	2	0	0	2	4
Total	**0**	**4**	**0**	**0**	**4**	**8**

Table A.3.9. Groups of categories 'Interaction', 'Aim' and 'Teaching Strategy'

Here they combined the design of the teaching strategy with the design of the interaction with the software and the aims of the computer for the pupil (Aims_P_C).

Charact. of Software: Organisation, Aims and Action	Group	PY.	TM	TE	SE	Total
COr_MedAims_P_SAc_T_I_P	0	0	2	2	0	4
Aims_P_SAc_T_I_PCOr_Med	0	0	2	2	0	4
Total	**0**	**0**	**4**	**4**	**0**	**8**

Table A.3.10. Groups of categories 'Content Organisation', 'Aim' and 'Actions'

They combined the design of the organisation of the contents, with the design of aims of the software for the pupil and with the design of actions of the teacher individually with the pupil.