Introduction

1. Overview of the thesis

Learning of network protocols has been studied in a project-based, internationally distributed, university course in computer systems taught jointly by two universities. The work presented in this thesis concerns students' understanding of network protocols[1]. Insights into the students' understanding of basic concepts within computer networks have been gained through phenomenographic research approach.

In this introduction I will explore the background and the goals of my research project. I will describe the framework in which my research is performed by giving an overview of the internationally distributed course – the Runestone initiative – its history and its aims, and by investigating the relationship between this project and related research projects. I will also describe some aspects of computer and data communication as well as of phenomenography, the research approach used in this project, that are relevant for the work presented here.

The technical report *Understanding computer network protocols* (Berglund, 2002) is the core part of this thesis and will be referred to as "Paper A" throughout this introduction. The report presents empirically based results on how students who participate in the Runestone course understand network protocols. Students' understanding of some protocols that are used within the project as well as their experience of the general concept of network protocols are investigated, and different ways of experiencing the protocols are discerned. Some features that indicate some aspects of a good learning outcome are identified: to be capable of understanding a protocol in different ways and of making relevant choices between the ways it could be experienced according to the context in which it appears.

The research is performed in a phenomenographic tradition. The aim of phenomenography is to analyse and describe qualitatively different ways in which a phenomenon can be experienced. It is thus an empirical research approach that puts the learner in focus, and the researcher studies the learners' ways of experiencing their study object, that is, what they are learning (Marton & Booth, 1997). Maintaining focus on the object of study is a feature of phenomenography that is important for my choice of a research approach. Phenomenography offers a possibility for me as a researcher to learn about computer science, *as it is experienced by the students*. Learning does not take place in a vacuum, but rather in a complex situation, where different aspects of the environment, together with the subject matter and the students

[1] A (computer) network protocol is a set of rules that enable communication between computers. The terms *network protocol*, *communication protocol* and *protocol* are used as synonyms in this report.

themselves, interact. The role of the context[2] of research, as well as issues concerning by whom the context is experienced, are further developed in Paper B, *On context in phenomenographic research on understanding heat and temperature* by Adawi, Berglund, Booth and Ingerman (2002), which is the next part of this thesis. This paper is a contribution to the methodological underpinnings of the phenomenographic research approach and supports the research that is described in Paper A.

2. The Runestone initative

Runestone is a joint computer science education research and development project, based on a project course in computer systems and software engineering initiated and run by Uppsala University (UU), Uppsala, Sweden and Grand Valley State University (GVSU), Allendale, MI, USA. From the start in 1998 the initiative was aimed to create an advanced course in computer science taught in a non-traditional way, that could serve as a framework, a source and a resource for research into computer science education (Daniels, 1999). The course is supported by a framework of computer science educators and computer science educational researchers at GVSU and UU, and also at University of Texas at Austin, Austin, TX, USA, St. Edwards University, Austin, TX, USA, Open University, Milton Keynes, UK, University of Kent, Canterbury, UK, and Chalmers University of Technology, Göteborg, Sweden. As will be further developed in section 3.3, other aspects of the students' learning and collaboration have been investigated by other researchers within this project.

In this section I will briefly describe the project course the students are taking, present the aims of the initiative and give an overview of its history.

2.1 The student project

The students who take this project-based course are advanced students in computer science at the two universities and are in their third or fourth years of studies. The work in the projects is performed in virtual teams, normally consisting of six students, three at each university. Each group creates a software system that controls a physical device. A Brio labyrinth (see Figure 1) was used as the physical device during the first four years (1998-2001), but in the year 2002 Lego Mindstorms are used. This report is based on the setting using the Brio Labyrinth. The task that the students were to solve is discussed in more detail in Paper A. The course is taught jointly by the universities with only a minimum of face-to-face meetings between the teachers and the students. Instead, each group is assigned to one of the teachers, either at GVSU or UU.

[2] The word *context* has many uses in educational research. Throughout this introduction *context* can be read as "meaningful background"

Figure 1. A Brio labyrinth. The aim of the game is to move a steel ball on the board from a starting point to a final point. The knobs are used to tilt the board and in this way move the ball.

2.2 The aims of the Runestone initiative

The computer science learning objectives for the students who take the course are to learn about computer systems by developing a software system and by building virtual working teams, as indicated in Paper A. However, the complete aims of the course that the students are taking should be seen in a wider context, where the students' experience of an international collaboration is another essential goal (Daniels et al., 1998). They point out that by working in internationally distributed teams, where some of the group members live in a different country and have a different educational background, an international experience is offered that reaches students who do not participate in student exchanges. The students should also gain experience of ICT-based[3] collaboration. It is also stated as one of the course goals, that the students should learn computer science concepts from their remote group members. Daniels et al. stresses that it can be assumed that the full group of students, due to the diversified skills and educational backgrounds, might be expected to produce a better software system than a local group could do.

Some objectives are formulated at the institutional level, concerning possibilities for the staff and departments involved to learn and develop. Daniels et al. state that a "secondary aim is to identify effective support structures for remote international collaboration, encompassing strategies for communication, management, and technology use". They also point pout that the peer learning among the staff concerning ways of teaching, and the possibility for the initiative to serve as a platform for research on learning in computer science, are important aspects of the project.

2.3 The history of the Runestone initiative

As already said, the Runestone initiative is supported by a web of collaborating universities that contribute with teaching and research interests in the project. In this environment several papers and reports have been written on different aspects of the project. The historical development that is described in this report is mainly based on works by Daniels (1999), Last, Almstrum, Daniels, Erickson, Klein (2000) and by Last, Hause, Daniels and Woodroffe (2002).

The two universities have taught the distributed collaborative course since 1998, when one pilot group, consisting of eight volunteer students, carried out the project. For the students in

[3] *ICT* is an abbreviation for Information and Communication Technology

this pilot group, technical issues such as discrepancies in programming language skills and the fact that only one hardware system, located in Sweden, was available, were more frustrating than factors that could be related to the virtual aspects of the collaboration according to Last et al. (2002).

From the following year, 1999, the course was scaled up to full classes at both universities. Table 1 show the development of the number of students. During these years the course has been jointly taught by one lecturer at each university. At both universities there have been changes of lecturers; that is, none of the lecturers teaching the course 2001 has participated in the project from its beginning. To accommodate the larger number of students, several copies of the hardware systems have been available at both sides.

Table 1. Total number of students taking the Runestone course

Year	Number of students
1998	8
1999	42
2000	93
2001	86

3. Related research

In this section I will give an overview of computer science education research. After a brief introduction to the field of research, I will discuss research projects other than the Runestone initiative. This presentation is mainly organised according to which research approach that is primarily used. In the next section I will further explore other research projects within the Runestone initiative.

3.1 Computer Science Education Research

Computer Science Education (CSEd) Research aims at improving learning and teaching within the field of Computer Science. As a research discipline it has a short history and has its roots in a range of disciplines including computer science: pedagogy, psychology, cognitive science, sociology to mention a few.

Computer science is a subject that some students find inexplicably difficult, particularly at an elementary level, as is shown in a literature survey performed by Ben-Ari (in press). It is a subject without the sort of experimental basis that physics has, or the educational tradition such as mathematics has. An essential issue for the research into computer science education is to tackle and resolve this problem.

3.2 An overview of research within Computer Science Education

A survey of the proceedings of recent years of the ACM SIGCSE (Special Interest Group on Computer Science Education) conference, the ACM ITiCSE (Innovation and Technology in Computer Science Education) and the PPIG (Psychology of Programming Interest Group), as well as the journal Computer Science Education (vol. 9 – 11) indicates that most of the projects presented were performed as case studies on specific courses, often performed,

evaluated and reported by the teacher of the course. These studies are normally driven by the needs of the computer science educators of the universities in question and address problems that have arisen in real teaching situations. Frequently the questions addressed concern the introduction of new methods, or new tools of different kinds, to teaching. Although valuable as a mean of sharing experiences between computer science educators, the results are, as also pointed out by Holmboe, McIver and George (2001) and Carbone and Kaasbøll (1998), often hard to generalise, since they are not based on pedagogically sound theories of learning or carried out with sound methodological principles.

The methodologically sound research in the field of Computer Science Education on problems related to education at university level has been dominated by cognitive psychologist approaches, with a rather abstract interest in the nature of knowledge structures, the acquisition of knowledge, and ways in which it can be made efficient. As examples can be mentioned the work performed by Baffes (1994), who has created a system that automatically identifies and recognises mistakes made by programming students. Based on the information that is collected across multiple students in a database, the program models the error and offers the student relevant feedback. Almstrum (1996) suggests that pre-university teaching in mathematical logic as well as the content of university level courses in discrete mathematics needs to be scrutinised. Her study, looking for the reasons for learning difficulties in the field and based on a large statistical material, shows that novice computer science students experience more difficulty with concepts involving mathematical logic than with other concepts in computer science. Holmboe (2000) argues that a teacher in computer science needs the means to evaluate students' mental models as well as guidelines for designing learning environments. He describes typical aspects of students' mental representations, built on an empirical study on students taking a course in system development.

Research in this tradition is often a based on experiments or quasi-experiments[4], testing hypotheses on the effects of educational devices by classical methods of psychology, comparing statistically the performance of a trial group against that of a control group before and after the former has been subjected to the device. Priebe (1997) has carried out a quasi-experiment, comparing a group of students that met in a cooperative learning environment with a group that in a classic way attended lectures. No significant differences were found between the groups. Wu (1997) has showed, in a controlled experiment on the teaching of recursion, a significant difference in the results for students that attended lectures that were based on concrete models, compared to those that were taught using abstract conceptual models. Al-Nuaim (1999) has developed and statistically validated a tool for formative evaluation of web-sites intended for education. McIver (2000) has in an experiment compared error rates for students who learned Logo or Grail1 as a first programming language. She compared syntax errors as well as logical errors and concluded that the design of the programming language has a substantial impact on error rates for novice programmers.

Other research projects draw their origin and motivation from pedagogical research, and have a strong focus on understanding students' learning in realistic settings. Phenomenography has been used in such research, since it allows the researchers to retain focus on the character and the structure of the subject matter, here computer science concepts and principles. It has been used to described and analyse students' learning within the field, for example where Booth (1992) has studied what it means and what it takes to learn to program. Learning to program

[4] A *quasi-experiment* aims at examining causality by quantitative research methods, in situations where complete control of the situation is not possible.

is characterised as a process of a growing awareness of what it means to program, in terms of both technical constituents, unthematised framework constituents and writing programs to solve problems. Teaching should, according to Booth, offer not only expert views of the subject, but also offer a rich variation in different ways of coming to an understanding of what it means to program.

In a phenomenographic study, Cope (2000) argues that the experience of learning in information systems has multiple educationally critical aspects. This is a reflection of the complexity of the desired way of experiencing an information system. It is likely that the students' learning experience becomes more effective, if active collaborative experiential learning tasks that focus on these critical aspects are designed.

In a socio-cultural research tradition (Säljö, 2000), where learning is seen as related to and dependent on the environment of the students and is built in collaboration with others – students as well as teachers – Holland and Reeves (1996) have studied students' work in a software development project. They have built an activity system (Engeström, 1987) that describes and helps understanding the context of the group work for the software developing student teams. They introduce the term *perspective* as a "view from somewhere" that is collective, historical and develops over the course. The teams took different perspectives, and as a consequence, they differed in their construal of the object of their work, the importance they gave to different sub-tasks and the way in which they carried out the work.

Few pedagogically sound research projects have been carried out that are based on the needs of computer science education, stemming from problems that have been proposed by computer scientists. The work performed by some computer scientists within a constructivist framework can be mentioned as interesting exceptions, as can the work of Cope (2000). For a constructivist, understanding and knowledge is not transferred from one individual to another. Instead, students construct their own understanding in an active process (Phillips, 1995). Ben-Ari (in press) argues that constructivism could inform teachers in computer science about models and how models should be taught, while Hajderrouit (1998) analyses how constructivist theory can be used to enhance students' learning of Java. There is a growing interest from the computer science education community for established research approaches such as constructivism.

Few of the studies performed in computer science education focus on network protocols. However, the journal Computer Science Education, Vol. 10, number 3, is devoted to network protocols. While most articles focus on methods of teaching or tools for teaching or for practical exercises for the students, some also analyse students' learning. Jard and Jéron (2000) present a case study on students' learning about validation of the alternating bit protocol[5]. They argue, based on their own experiences of teaching the course, that students understand the need for automated tools for protocol verification by trying to use such tools. Mester and Kruhm (2000) argue, based on their own experiences as lecturers, that animations of formal methods to a certain degree can improve students' results on exams, and that the students' ability to find imaginative solutions to particular kinds of problems increased.

The overview presented indicates that CS Education research is a small but diversified field, where researchers with different backgrounds and aims perform studies with different objectives. However, most research projects are case studies or different kind of preliminary

[5] The *alternating bit protocol* is a simple protocol that aims at transferring data over medium that can lose data.

studies, often focusing on how students use a single resource, like other students in a collaboration, a web-tool or a purpose-made device for instruction and labs. Few, if any projects, other than the Runestone project described above, have been performed where researchers, taking their starting point from the needs of CS education have studied computer science students in a naturalistic setting, to learn how students experience different resources available for their studies, and how they use these resources in their learning.

3.3 Other research projects within the Runestone initiative

As was stated in section 2.2, the Runestone initiative was from the beginning constructed to serve as an internationally distributed course in computer systems, as well as an environment for research on students' learning in computer science. The initiative is closely related to a network of computer science educators[6]and has generated several research projects that investigate different aspects of the students' work. Although these research projects use the same student project as their object of research and to collect empirical data, they have distinctively different research questions and theoretical backgrounds.

Hause and Woodroffe (2001) study the communication and interaction within the student teams, in order to find characteristics in the interaction patterns within teams that perform well and poorly in software engineering. Data is collected from the e-mail conversations as well as IRC[7] sessions between the team members. The data is coded according to a set of categories developed within the research project using discourse analysis, in the sense that Coolican (1999) describes as "qualitative analysis of interactive speech, which assumes people use language to construct the world as they see it according to their interests". Preliminary findings show that communicating among the group members is important, and that the timing of the interaction is crucial.

The group development in the student teams and the role of conflicts are in focus in the work performed by Last (2002), who uses grounded theory[8] as a research approach. In her current work, she is investigating if "group development models developed and validated with face-to-face groups require modification when applied to virtual teams"[9] and if "certain types of conflict in a team result in a more productive team and a better product".

Pears, Daniels, Berglund and Erickson (2001) have addressed the issues of the impact of the different grading scales[10] on the students' motivation to contribute to the work of the group. There it is argued, based on a statistical analyses of the students' assessment of their team-mates' work as expressed in the peer evaluation, that the different grading scales did not affect the students' level of input in the research.

---

[6] The network is further described at `http://www.docs.uu.se/csergi/`

[7] *Internet Relay Chat, IRC,* is a system for human communication over Internet. A computer running an IRC program can be used in a way similar to a text telephone and offers to the user a possibility to communicate with any other IRC user in the world.

[8] *Grounded theory* is a qualitative approach that generates theory from observation.

[9] Quotes from `http://www.cs.stedwards.edu/~lastm/SIGCSE_2002_DC_Mary_Last.htm`

[10] The Swedish students had a grading scale with only two grades, *passed* or *failed*, while the American students could get grades varying between *A* and *E*, including grades with + and –, as D+ or B-

Daniels, Faulkner and Newman (2002) discuss open-ended group projects (OEGP) in computer science education in a comparative study and create a framework for describing such projects as The Runestone initiative as one example of how an OEGP can be organised. They argue, based on student evaluations, discussions with employers and their own experiences that OEGP projects, where the end-product is not well-defined, are valuable for preparing the students for their professional lives.

This overview, together with the current study, clearly indicates that there is great interest for carrying out research within the Runestone initiative, and that different research questions can be addressed from the empirical data collected in this project. The studies presented that concern the group development, characteristics of low and high performing teams, and grading of distributed projects, as well as my study, concerning the learning of computer science and the role of the environment, and together form a whole. They create a research-based overview of Runestone and students' learning and collaboration in virtual teams.

4. Data communication and network protocols

One important objective of the Runestone course is that the students learn about computer communication. I will in this section give a brief introduction to computer and data communication and to networks protocols, following Tanenbaum (1996) and Stalling (1997).

The communication between computers can be designed, analysed, and described in different ways. Focus can be on different aspects of the communication, whether it is the physical transmission of raw data or a semantically rich communication between two advanced application programs such as mail-programs, or at any level between these two. A strategy is needed to handle this complexity, in order to make the design of a computer network and its components a feasible task. Layered design is the completely dominating methodology used to tackle the complexity and to define tasks that can be further developed.

Communication between computers is governed by sets of rules. These rules that defines how the communication between the functional units of the communicating systems, and clarifies how it shall take place, are called *protocols*. In this thesis I refer to them as *network protocols* or *computer network protocols*.

The section starts by presenting the idea of layered models, then continues with descriptions of the layers in the TCP/IP model, which is the most important in the project the students are doing. The protocols from the TCP/IP model that are used in this project are mainly TCP, and to a lesser extent, UDP. RMI is also important in the light of the Runestone project and is presented in the following section. Then before a summary, some aspects of practical programming are discussed.

4.1 Layered models

As mentioned above, data and computer communication is designed in layered models. Each layer offer services to the higher levels. The services offered by a layer are implemented with the help of the services offered by lower layers. As an example can be mentioned that an application program, as a mail-program (VM-mail, Outlook Express etc) uses services or routines that from an underlying level and that for example offers the possibility to set up a

connection to another computer. This service, in its turn is implemented with the services that are offered by a lower level, such as address translations. For a user of a mail program it seems as if data from his or her mail program was transferred directly to its peer (or to a mail-server) on another computer. From a technical perspective, data is passed to lower level routines to be delivered to the corresponding application on the remote machine.

Basically, there are two layered architectures that are discussed today: The Internet architecture, which is frequently referred to as the TCP/IP[11] architecture and the Open Systems Interconnection architecture, OSI. The former completely dominates practical applications and will be in focus in this presentation. The OSI model is clearly defined and has well-standardised protocols. The model is mainly used as a reference in order to understand networks.

4.1.1 The TCP/IP layered design

In this section I will explore some aspects of the layered design of the TCP/IP architecture as shown in Figure 2. The link level and the physical level will be referred to as underlying networks in the presentation, since their characteristics are not important to the project described in this thesis. The terminology is consistent with that used by Tanenbaum (1996).

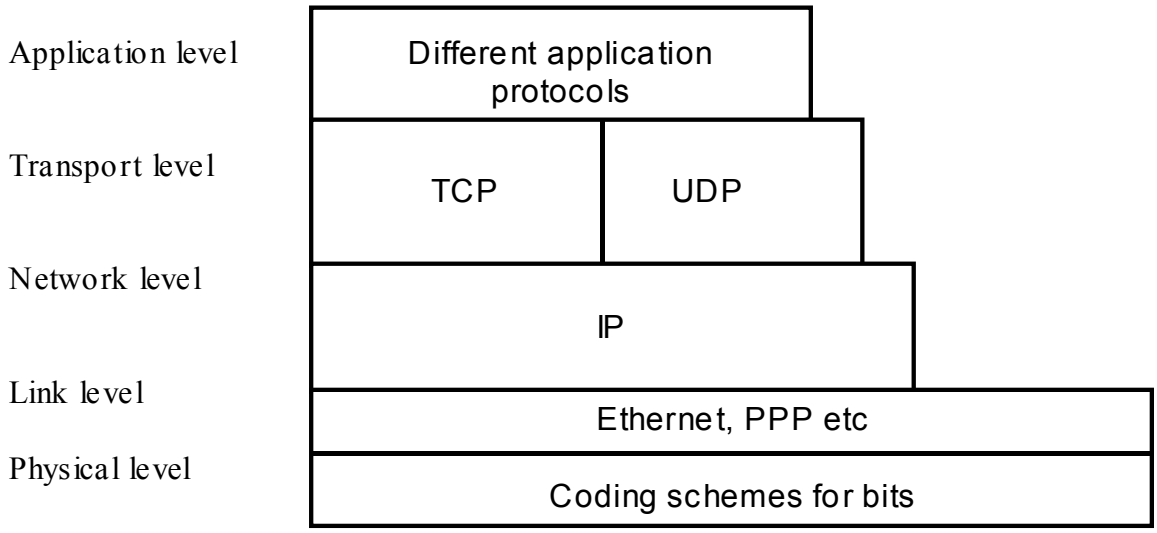

Figure 2. The TCP/IP layered model

The underlying networks

An *internet*[12] or an *internetwork* is a network that consists of a set of independent networks, that each has its own character. These networks are connected to each other through *gateways*

[11] TCP/IP stands for Transmission Control Protocols and IP for the Internet Protocol. Abbreviations are rarely spoken out within the field of computer communications. Acronyms are used as names of the protocols as well as the other entities discussed.

[12] I will follow the convention of writing *Internet,* with a capital *I*, when referring to the global Internet, and of writing *internet* when discussing independent internets. The Internet is a large and well-known internet.

or *routers*. The independent networks that form an internet can for example be Local Area Networks, LANs, or internets, with the recursive definition of internets. An internet is thus a logical network, that consists of a collection of physical or logical networks. This points to one of the important features of the Internet architecture: It does not prescribe any specific physical medium or any specific kind of networks to be connected. Instead it is designed for communication that goes over different kind of networks.

When describing the TCP/IP protocol architecture as a four-level model, this set of different network protocols, with their own characteristics (and their own possible layered architectures) together form the first level.

Network level

The second level is the *network* level that handles the issues of the heterogeneous structure of the underlying networks, large size of an internetwork, and its continuously changing character. It mainly offers two services to the level above itself: it handles addressing on an internet-wide level and offers tools for delivery of data to the destination. In other words, the protocol at the network level, the *Internet Protocol* (IP) accepts packages of data from a higher level, translates addresses so that they correspond to the actual physical settings and forwards the data, formatted as needed, to its destination using the services of the underlying levels.

Data are sent by the IP protocols as IP *datagrams*, packages of a limited length (less than approx. 65 000 bytes) that each contains a part of, or if data is short, the full message. The IP protocol tries to deliver the datagrams that each "travel" independently over the net of the others. Frequently datagrams needs to take many steps and pass several routers to reach its destination. This issue is handled on the network level. The datagrams can however arrive to their hosts in the wrong order, corrupt or not at all. The IP protocol is a *best-effort* protocol; it tries its "best effort" to send data, but does not guarantee any qualities of the service.

Transport level

If a guarantee of delivery is needed, it must be taken care of by the layer above the IP level, the *transport layer*. The transport level offers the possibility for two computers to keep an end-to-end exchange of data. To do this, the transport layer uses the services offered by the network level. There are two dominating protocols at this level, the *Transmission Control Protocol*, TCP, and the *User Datagram Protocol*, UDP, offering different types of services to the level above.

TCP, *Transmission Control Protocol*, is reliable, connection-oriented protocol. This means that the protocol allows communication where data is delivered to any other machine on the internet without errors. Data is delivered to the user of the TCP protocol; that is, to a higher level, in the order it was sent. Issues like unreliable physical networks, different speed of the underlying networks or different sizes of data packages on different sub-networks are taken care of by TCP. A user of the TCP protocol needs to set up a connection in order to communicate with another computer, a concept that is reminicent of a telephone call.

UDP, *User Datagram Protocol*, is an unreliable, connectionless protocol. This means that a sending computer does not get any confirmation if the data sent has reached its destination or not. Still, it is useful in applications where but where speed is important, such as video

conferencing. A user of video connection can more easily accept a loss in the quality of the image than delays of various lengths.

Application level

The two protocols described together form the transport layer. They offer services that are used by programs at the level above, at the *application* level. Here there is a large number of protocols available offering a rich variety of services that the user of a computer on an internet might want. As an example can the Hyper Text Transfer Protocol, HTTP be mentioned. It defines the rules for the communication between a Web-browser (as Netscape, Internet Explorer, Mosaic, Lynx etc) and a web-server, which stores and organises web pages. This means that anyone who wants to create his or her own web-browser (or web-server) needs to write a program that follows the rules of the HTTP. Normally the services of TCP are used to implement HTTP, since a reliable transfer of data is desirable.

Other examples of application level protocols are mentioned in Table 2.

Table 2. Some example of application protocols

Abbreviation	**Full name**	**Purpose**
SMTP	Simple Mail Transfer Protocol	Transfer of mail
NNTP	Network News Transfer Protocol	Transfer of news and support for news reading in USENET news
FTP	File Transfer Protocol	Transfer of files between computers
Telnet	Telnet	Login on other machines
NV	Network Video	Video Applications

4.1.2 Peer communication

A conclusion from the section above is an internet "looks" different for users at different levels. A user sending an electronic mail through a mail program experiences an internet in differently from a student who works on the Runestone project with a routine that implements communication between the camera and the game server (Paper A, section 2.4). While the former uses a program that implements the SMTP protocol, the latter is most probably writing a C++ or Java program that uses the services of TCP.

Figure 3 illustrates this with an example. A user of the FTP (File Transfer Protocol) protocol sees his or her program and the communication with another FTP program as the FTP protocol prescribes. Thus for him or her there is *peer communication* (marked as "a" in the picture) between two FTP programs. The user does not need to be aware of the underlying levels; they are hidden.

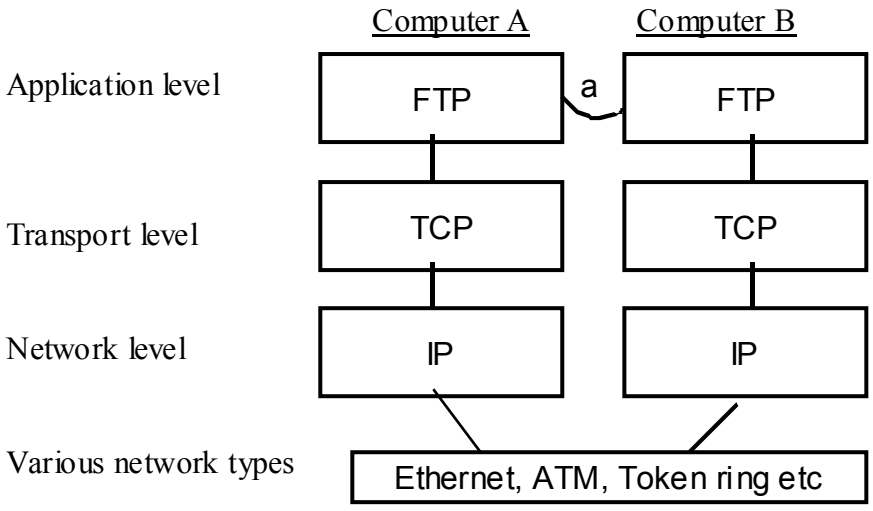

Figure 3 An example of peer communication in a layered model

4.1.3 Dependencies between protocols in the TCP/IP architecture

As has been mentioned, the TCP/IP protocol hierarchy is created to allow communication over networks with different characteristicss, and to make it possible to create application programs for large numbers of purposes. The internet architecture is diversified and constantly evolving, new protocols are created, and existing protocols are updated with new versions. A way to visualise in which level different protocols are and on which other protocols they depend, is through a protocol graph, Figure 4.

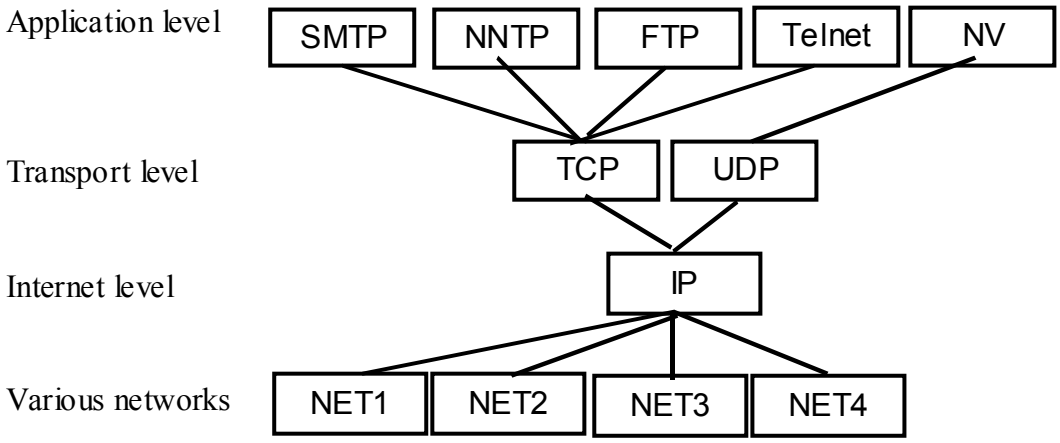

Figure 4. Internet protocol graph

The network level, with the Internet Protocol as its only protocol, is in the centre, and serves to make a communication over different networks, marked NET1, NET2 etc in the picture. On the level above the network level, there are two protocols, TCP and UDP that both use the services of IP. At the application level, there are a large number of protocols, whereof only a few, with their abbreviations explained in Table 2 are shown in the picture. Most of these protocols are implemented over the support provided by TCP, since TCP offers a reliable

communication. NV (Network Video) uses UDP, since for video applications speed is a crucial factor, while it is easy to accept losses in data.

4.2 Remote Method Invocation

RMI, *Remote Method Invocation*, provides programmers with a facility to access code or objects on a remote machine. It is implemented on top of TCP/IP. As is mentioned in Paper A, it is closely related to the object-orientation[13] in the programming language Java. An object-oriented program can be viewed as a set of interacting objects. The execution of an object-oriented program takes place in the objects and in the interaction between the objects. Java RMI offers the possibility to execute Java programs where the different objects are distributed on different machines[14] over an internet. To a programmer this means that RMI offers the possibility to use routines (called methods within the terminology used) that exist on remote machines as if they were available on the locla computer.

When transferring unformatted text between computers, all characters are, at least in simple cases (a Western 26-character alphabet, no uncommon control characters etc), treated in the same way. There is no semantically complex information in the characters that influences how the transfer should be made, with the exception of well-defined situations as indications of the end of transfer. When communicating between objects the situation is different: Parts of the data that is transferred is information about other parts of the data. A call to a method on an object contains information about addresses, permissions, data to be transferred etc. This extra complexity offers the possibility to write programs that are distributed over a set of computers in a reasonably easy way, but demands that the programmer handles security issues etc in a correct way.

4.3 Data communication in practice

As was mentioned earlier, a protocol is a set of rules that defines communication between two entities. The term is used for a protocol as an abstract entity, with formally defined rules, as well as for a programming package that implements the protocol. In this section, I will briefly describe, in a rather concrete way, some of the routines that are used by the students when they write code for the project, with the aim of explaining with what tools the students practically work during their project.

There are several possible protocols that can be used within parts of or the whole project. Three standard protocols have frequently been used during the Runestone project: TCP, UDP, and RMI. Some students have designed and implemented their own purpose-written application level protocols, based on TCP. Many possible protocols have never been used by

---

[13] *Object-orientation* is based on the idea that a program consists of a set of communicating entities. The execution of the program takes place within these entities, and in the interaction between them. Java and C++ are programming languages that support this style of design and programming. There is a vast literature on object-oriented programming and object-oriented programming languages. Budd (1999) discusses the ideas behind object-orientation and Java.

[14] By *machine* I refer to a *virtual machine* that can, in short, be described as a simulated computer that runs on another computer. In other words, a virtual machine is a program, that, when executed, behaves as a computer with well-defined properties. Virtual machines are one of the underlying techniques for platform-independent programs. Java that can be run on different kinds of computers and in different environments has a virtual machine. Java's virtual machine is (at least in theory) the only program that has to be rewritten to run Java in a new environment.

the students; for example CORBA (Common Object Request Broker Architecture) has never been used during the years of the Runestone project, as far as I know. In this presentation (mainly following Harold, 2000) I will discuss practical issues of the TCP protocol, which is the most commonly used protocol by the students in the Runestone project.

TCP sockets, or the *TCP socket programming interface* are used by a programmer who wants to access the services of TCP, most frequently in order to transfer data. This is a common task within the Runestone project. Sockets offer to the programmer practical routines, or mechanisms, to transfer data from one computer to another. A socket can be seen as an endpoint of a communication connection between two computers. This endpoint is easily accessible to a programmer through routines that are accessible from a library or integrated in the programming language. While the concept of sockets, stemming from UNIX, is now generally accepted, details of the implementations vary to a certain degree with different programming languages and operating systems. Java, however, offers an abstracted, standardised interface for sockets that is not dependent on the operating system.

The following basic operations[15] can be performed by a TCP socket:

- Connect to a remote machine
- Send data
- Listen/Wait for incoming data
- Receive data
- Close a connection
- Bind the connection to a port
- Accept connections from the remote machine on the bound port.

While the first five are rather intuitive (the reader can compare them to a telephone call), the last two aim at making it possible to have several connections to one computer. These two are only needed in the phase when the connection is established. The details of their use are outside the scope of this presentation.

The following steps indicate one of many possible ways in which a programmer can use TCP sockets. Details are omitted below:

1. A new socket is created
2. The new socket tries to connect to a remote machine
3. The connection is established and its details are agreed upon
4. Now, data is transferred
5. The connection is closed by any (or both) of the computers.

Each of these steps (except number 4) normally corresponds to one or a few lines of code, frequently using the routines of the TCP sockets.

The connection offered (number 4) is full duplex; that is, both computers can send and receive data simultaneously. Data does not have any predefined meaning. What data means depends on decisions taken by the programmer. He or she can, for example, decide to communicate through commands with an HTTP server (a Web server), transfer a file where the data does

[15] Implementations of TCP sockets offer a considerably larger set of routines. Normally they represent variations of the routines presented here.

not have any meaning for the transfer itself (but that most probably has an interpretation for the users), or by a purpose-written protocol. This means that the code handling data transfer can vary enormously in size and complexity from a few lines to long complex code units.

My intention is that this section should give insights in what the students do, when they are coding in the Runestone project. The specification that they get does not demand that a specific solution is taken or that a specific protocol is used. On the contrary, the students are encouraged to take their own design decisions and to find their own solutions to different questions.

5. Phenomenography

While the previous section has focused on the object of the students' studies, computer networks, in this section I will describe the research approach that I have chosen. As stated in the first section of this introduction, the aim of my research is to understand how university students understand, or experience, computer networks, in order to propose ways of improving learning in distributed projects in computer systems. To address this issue, I have decided to use a phenomenographic research approach, a decision that I will justify later.

The section starts with a discussion about the object of research and the aims of a phenomenographic research project, highlighting aspects that are important to the project that I describe in Paper A. In the next sub-section, I will discuss what learning means in the light of the current research project. Then I will explore, referring to my current study, some aspects of how a phenomenographic research project can be performed. As a summary, I will return to the issue of phenomenography as a research approach in my research about students' understanding of network protocols when they take an international distributed course in computer systems.

5.1 The object of research

Marton (1994) describes phenomenography as an empirical research approach in the following way:

> Phenomenography is the empirical study of the limited number of qualitatively different ways in which we experience, conceptualise, understand, perceive, apprehend etc, various phenomena in and aspects of the world around us. These differing experiences, understandings etc are characterised in terms of categories of description [...] form [...] hierarchies in relation to given criteria. Such an ordered set of categories of description is called the outcome space of the phenomenon [...]
> (p. 4424)

Since then, phenomenography has also taken a theoretical turn in relation to learning, but for my purposes Marton's description of empirical studies of ways of understanding or expriencing is appropriate.

In section 3 of Paper A, a phenomenographic research project is described in the following way:

> A phenomenographic research project [...] aims at analysing and describing the variation in ways in which central concepts of the subject matter are understood or experienced by the learners. In my study, university students' experience of computer networks is in focus, and

> phenomenography offers the possibility for me, as a researcher, to investigate the students' own experience of network protocols.
>
> One of the keystones of phenomenography is that phenomenographic researchers can arrive at a limited number of qualitatively distinct categories of description which succinctly and adequately cover the countless ways in which a phenomenon can be experienced, or understood. The results are articulated in a set of qualitatively distinct categories of description that express the variation in how a phenomenon is experienced by the learners, called the outcome space. The outcome space that I arrive at thus gives me, as a computer scientist and phenomenographer, the possibility to relate the students' understanding of computer network to the goals of the education, as it is expressed in the course descriptions and as I, as computer scientist, understand the protocols. The strong focus on the object of the students' learning is an important feature of phenomenography as a research approach in the research that I present in this thesis, maintaining as it does the subject matter that is of prime interest to me as a teacher.

Phenomenographic research is not performed in controlled experiments or as quasi-experiments, but in "close proximity (both spatial and conceptual) to the learning situation [the students] find themselves" (Booth, 2001). She continues by arguing that it is not a quantitative methodology:

> The results are descriptive and lie at a collective level, in the sense that individuals are seen as contributing fragments of data that together constitute a whole and collective experience [...]. (p. 172)

The outline above indicates the roles that are given to the phenomenon (here: network protocols) and the experiencers (students taking an internationally distributed course in computer systems) in a phenomenographic research project. The object of the research is the relationship between these two entities; that is, the students' experience of network protocols. Neither the experiencer nor the phenomenon in focus can alone serve as study objects. This perspective, a second order perspective, is illustrated in Figure 5, where arrow (a) indicate the relationship between the learner and the object of their learning. At the risk of oversimplifying, the researcher investigates this relationship, and is in the same way a learner who learns about the students' experience of a particular phenomenon, as is indicated by arrow (b).

5.2 A phenomenographic research project

In this section I will discuss some important aspects of the different phases of a phenomenographic research project. However, right from the start I want to stress that such a project is not linear in its character. Rather, the nature of a phenomenographic research project, where the aim for the researcher is to analyse and describe the experience of a learner, encourages an iterative way of working. Still the phases presented here give an overview of the work that I performed in the study that is presented in this thesis.

5.2.1 Formulating the research question

The aim of a research project is basically formulated at the outset of a study. Here, as we argue in Paper B, the researcher is the main actor and is influenced by the context in which he or she lives and works. This also influences the research questions he or she can, and finds useful, to address.

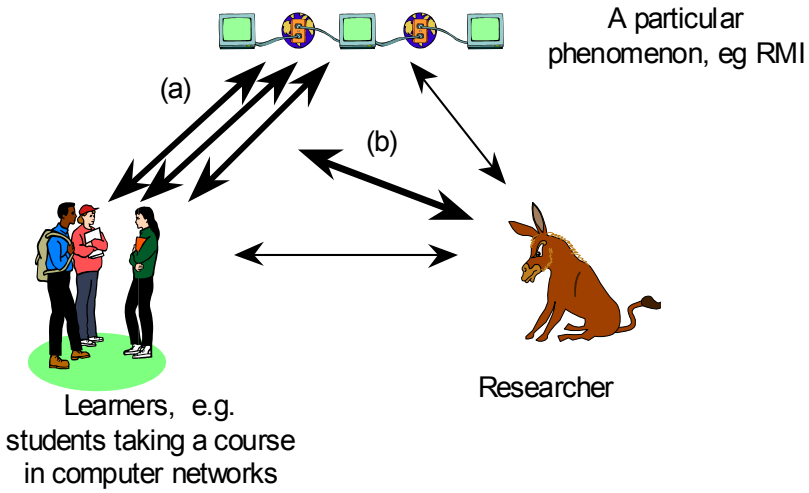

Figure 5. A description of the second order perspective, used in phenomenography.

5.2.2 Data collection

As Booth (2001) points out, "phenomenographic research into learning is empirical in that the source of data are the learners themselves" (p. 172). The aim of the data collection is to collect expressions of the varying ways in which a particular phenomenon is experienced within a group. The predominant way to collect data in phenomenographic research projects in general, as well as in the project presented here, is through interviews. In order to maximise the richness of data collected, the interviewer enters into a dialogue with the interviewee. The interview is normally based on a script of relevant questions, but has an open form, where the interviewer encourages the interviewee to talk freely and interesting statements from the interviewee are followed up.

Phenomenographic research projects do not aim at certifying causal relationships, or at finding quantitative measurements of learning. Instead, a picture of the understanding and the learning about a phenomenon across a group is created. This goal offers to the interviewer a possibility to improvise, without losing sight of the phenomenon under investigation, during the interview, and to let the interviewees talk.

A possible way to increase the richness of data is described in Paper B. We suggest that the interviewer can deliberately vary the setting of the phenomenon that is discussed, with the aim that the interviewee should experience different contexts for the phenomenon under investigation, and express his or her experience of the phenomenon in a variety of ways. In the research presented in this thesis, this has been practised. As an example can be mentioned the two excerpts from the first interview with Albert presented in section 6.1.1 of paper A, both discussing RMI, but the interviewer introduces the discussion theme in two different contexts.

5.2.3 Analysis

The aim of the analysis is to create an outcome space consisting of a limited number of categories of description that together express the ways in which a particular phenomenon is experienced by a collective. The researcher normally forms a pool of meaning that contains

all individual expressions that has been collected. Here the researcher can analyse the individual expressions outside the context from which they originally stem in order to create categories of description that reflect experience of the particular phenomenon at a collective level. The researcher recontextualises the individual statements in the categories that he or she creates and that now reflect the different ways in which a phenomenon is experienced. This is a dynamic process where the researcher "engages in with this pool of data and seeks critical differences that can act as catalysts for an understanding of the whole" (Booth, 2001, p 172). The statements are thus analysed in the context of each individual interview and in the context of the collective. It is also analysed in the context as it is experienced by the researcher himself or herself, a context that is formed by his or her understanding of the original data, the growing results, as well as his or her experience as a phenomenographer and as specialist within the subject area. (Paper B).

This point is clearly illuminated by Paper A. It would not have been possible to interpret the students' statements of computer network protocols without an understanding of the area, since the analysis is based on the researcher's interpretations of the students' experience of *network protocols*.

The creation of the categories is an iterative process, where the researcher starts with a tentative understanding, and then, through reconsiderations and refinement, reaches a description that he or she finds relevant to address the research question and honest to the data. In a sense, the process can be seen as a discussion between the researcher, the collective of the interviewees, the pool of meaning and the developing categories of description. This way of working cannot be taken as algorithmic - phenomenography is not prescriptive and consequently does not demand that certain steps are taken. Instead it is a research approach that puts forward certain principles, while "the actual methods used vary according to the specific question being addressed" (Booth, 2001, p 172).

5.2.4 Results and deployment of results

A phenomenographic research project leads to, as has been discussed above, a set of categories that together describe the experience within a group of a particular phenomenon. The results are stripped of their original context, and can be viewed as the outcome of the research project. Because of the selection of students in the study, to cover variation in important criteria such as educational achievment and motivation, a generalisation of the categories of description can be made to groups of similar students.

However, with the goal I have formulated for my research project (section 1 in Paper A), the focus is on the application in the teaching and learning situations of the universities. A way to apply the results is to bring them back to their original context. As we point out in Paper B, the results can be applied at the individual level, at collective level or at the level of the researcher.

At the *individual level*, the researcher can turn the outcome space back on the interview, to let the results illuminate the original interview. This is practised in Paper A, where a good understanding of a network protocol is described as a capacity to shift in relevant ways between different ways of experiencing the protocol.

At the *collective level*, studies can be used as a foundation for conclusions about the variation in how a certain phenomenon is experienced in a larger group of students. An understanding

of the variation in the ways a network protocol is experienced, is certainly a good tool for teachers both when planning teaching and in the classroom. The results in Paper A indicate that teaching ought to be planned and performed in a way that encourages the students to understand the protocols in different ways.

Finally, at the *level of the researcher*, the results become a part of the researcher's understanding of his field, and thus inform future research. To a certain degree this line of reasoning is visible in Paper A, where the results from section 4, that concern variation in students' understanding of individual protocols, have informed me as a researcher in my analysis of the general concept of network protocols, that is presented in section 5. Furthermore, the results presented in this thesis will be an important building-stone in my future work concerning understanding of network protocols in the context of the course that the students take.

5.3 My choice of phenomenography in relation to my research objective

The aim with this project, as is stated in Paper A, is "to explore university students' learning of advanced computer science concepts in an internationally distributed project course" with an overall objective to improve learning and teaching of computer networks at the universities.

There are several reasons why this research is performed with a phenomenographic approach. The strong focus on the object of the students' learning is important to me, since it gives me, as a computer scientist, the possibility to learn about the students' understanding of computer science. The results of a phenomenographic research project, as mine, also reflect and express the students' experience in a relatively direct way. Results are not described in abstract ways as for example cognitive models or statistical correlation. I find that this closeness to the students' world and to the subject area offers rich possibilities to give results that can be used to obtain my initial goal, which is to improve education.

6. Conclusions and future work

In this thesis I have investigated the understanding of computer network protocols among students who participate in an internationally distributed university course in computer systems. The research has been performed with an empirical phenomenographic approach. This approach offers to me, as a researcher, the possibility to explore network protocols, as they are understood by the students. Critically different ways of understanding network protocols have been identified. It is argued that students' understanding of a protocol is related to the context in which the protocol is experienced.

Recommendations for teaching of computer systems in distributed projects are made, based on the results. Universities should teach computer networks in a way that encourages students to understand network protocols in these critically different ways, and that stimulates them to shift between these ways depending on the task at hand.

In my ongoing work which will be reported in my doctoral thesis I will explore the role of the context further and in what ways the context of an international group project, as it

experienced by the participants, interact with their learning. In the continuation of this project I will take a broader perspective, and I will also study the international group work, as it is experienced by the participants. The context of the project that the students undertook, analysed as an activity system, will be related to the learning of computer networks. The upcoming work, which also has an overall goal of improving teaching and learning in computer systems at universities, aims at identifying issues in group projects that promotes learning.

7. References

Al-Nuaim, H. A. (1999). *Development and Validation of a Multimedia User Interface Usability Evaluation Tool in the Context of Educational Web Sites*. PhD thesis, George Washington University, Washington, DC, USA.

Almstrum, V. (1996). Investigating student difficulties with mathematical logic. In N. Dean and M. Hinchey, (eds.), *Teaching and Learning Formal Methods*. London, UK: Academic Press.

Ashworth, P. D. & Lucas, U. (2000). Achieving empathy and engagement: a practical approach to the design, conduct and reporting of phenomenographic research. *Studies in Higher Education*, 25, 295-308

Baffes, P. (1994). *Automated student modelling and bug library construction using theory refinement*. PhD thesis, University of Texas at Austin, Austin, TX, USA.

Ben-Ari, M. (in press). Constructivism in Computer Science Education, Accepted for publication in *Journal of Computers in Mathematics and Science Teaching.*

Booth, S. (1992). *Learning to program: A phenomenographic perspective*. Gothenburg, Sweden: Acta Universitatis Gothoburgensis.

Booth, S. (2001). Learning Computer Science and Engineering in Context. *Computer Science Education,* 11(3). 169-188.

Budd, T. (1999). *Understanding Object-Oriented Programming Using Java*. Reading, MA, USA: Addison-Wesley.

Carbone, A. & Kaasbøll, J. (1998). A survey of methods used to evaluate computer science education teaching. In *ACM SIGCSE/SIGCUE Conference of Innovations and Technology in Computer Science Education (ITiCSE'98)*.

Coolican, H. (1999). *Research Methods and Statistics in Psychology. Second Edition*. London, UK: Hodder & Stoughton.

Comer, D. (1997). *Computer Networks and Internets.* Upper Saddle River, NJ, USA: Prentice-Hall.

Cope, C. (2000). *Educationally critical aspects of the experience of learning about the concept of an information system.* PhD thesis. La Trobe University, Bundoora, Victoria, Australia.

Daniels, M., et al. (1998). RUNESTONE, an International Student Collaboration Project. In *Proc of Frontiers in Education* (FIE'98).

Daniels, M. (1999). *Runestone, an International Student Collaboration Project. NyIng report No 11*. Linköping, Sweden: Linköping University. Also available at `http://www.docs.uu.se/~matsd/NyIng.html`

Daniels, M. , Faulkner, X., Newman, I. (2002). Open Ended Group Projects, Motivating Students and Preparing them for the 'Real World'. *In Proc. Conference on Software Engineering Education and Training*, Covington, KT, USA

Derrick, J. & Fincher, S. (2000). Teaching Communication Protocols. *Computer Science Education,* 10(3). 195 - 202.

Engeström, Y. (1987). *Learning by expanding. An activity-theoretical approach to developmental research*. Helsinki, Finland: Orienta-konsultit.

Hajderrouit, S. (1998). A constructivist framework for integrating the Java paradigm into undergraduate curriculum. *SIGCSE Bulletin, 30(3),* 105 – 107.

Harold, E. R. (2000). *Java Network Programming. Second Edition.* Sebastopol, CA, USA: O'Reilley

Hause, M. & Woodroffe, M. (2001). *Team Performance Factors in Distributed Collaborative Software Development.* In *Proc. Psychology of Programmers Interest Group (PPIG), Bournemouth, UK.*.

Holland, D. & J. R. Reeves (1996). Activity theory and the view from somewhere: team perspectives on the intellectual work of programming. In . A. Nardi (Ed.) *Context and consciousness: activity theory and human-computer interaction*, 257-281. Cambridge, MA, USA: MIT Press.

Holmboe, C. (2000). A framework for knowledge: Analysing high school students' understanding of data modelling. In *Proc. Psychology of Programmers Interest Group (PPIG)*.

Holmboe, C., McIver, L., George, C. (2001). Research Agenda for computer science education. In *Proc. Psychology of Programmers Interest Group (PPIG)*.

Jard, C. & Jéron, T. (2000). An educational Case Study in Protocol Verification and Distributed Observation. *Computer Science Education*, 10(3), 203-224.

Last, M. (2002). *Virtual Teams in Computing Education*. Available on-line at
`http://www.cs.stedwards.edu/~lastm/SIGCSE_2002_DC_Mary_Mast.htm`

Last, M., Almstrum, V., Daniels, M., Erickson, C., Klein, B. (2000). An International Student/Faculty Collaboration: The Runestone Project. In *Proc. ACM SIGCSE/SIGCUE Conference of Innovations and Technology in Computer Science Education (ITiCSE'00)*.

Last, M., Hause, M., Daniels, M., Woodroffe, M. (2002). Learning from Students: Continuous Improvement in International Collaboration. Accepted for publication in *Proc. ACM SIGCSE/SIGCUE Conference of Innovations and Technology in Computer Science Education (ITiCSE'02)*

Lipnack, J., & Stamps, J. (1997). *Virtual Teams: Reaching Across Space, Time, and Organizations with Technology*. New York, NY, USA: John Wiley and Sons.

Marton, F. (1994). In The International Encyclopedia of Education. Second edition, Volume 8. Eds. T. Husén & T. N. Postlethwaite: Pergamon. Also available at:
`http://www.ped.gu.se/biorn/phgraph/civil/main/2approach.html`

Marton, F. & Booth, S. (1997). *Learning and Awareness*. Mahwah, NJ, USA: Lawrence Erlbaum Associates.

McIver, L. (2000). The Effect of Programming Language Error Rates of Novice Programmers. In *Proc. Psychology of Programmers Interest Group (PPIG)*.

Mester, A. & Krumm, H. (2000). Animation of Protocols and Distributed Algorithms. *Computer Science Education, 10(3)*, 243 –266.

Pears, A., Daniels, M., Berglund, A., Erickson, C. (2001). *Student Evaluation in an International Collaborative Project Course*. Presented at the Wise workshop of the SAINT conference, San Diego, CA, USA.

Phillips, D. (1995). The good, the bad, and the ugly: The many faces of contructivism. *Educational Researcher*, 24(7), 5 – 12.

Priebe, R. (1997). *The effects of cooperative learning on content comprehension and logical reasoning.* PhD thesis, University of Texas at Austin, Austin, TX, USA.

Stalling, W. (1997). *Data and Computer Communications*. Upper Saddle River, NJ, USA: Prentice-Hall.

Säljö, R. (2000). *Lärande i praktiken: Ett sociokulturellt perspektiv*. Stockholm, Sweden: Prisma. (in Swedish)

Tanenbaum, A. S. (1996). *Computer Networks. Third edition*. Upper Saddle River, NJ, USA: Prentice-Hall.

Tuckman, B. (1965). Developmental Sequence in Small Groups. *Psychological Bulletin*, 63, 384-399.

Wu, C.-C. (1993). *Conceptual models and individual cognitive learning styles in teaching recursion to novices.* PhD thesis, University of Texas at Austin, Austin, TX, USA.