Technical Report of the Virtual Unix Lab Project

Education of System Administration

Hubert Feyrer <hubert@feyrer.de>
Computer Science Department of the University of Applied Sciences Regensburg and
Information Science Department of the University of Regensburg

December 4, 2007

Abstract

This article discusses education of system administration. Theoretical foundations cover psychology, learning theory, instruction theory, and instructional design, leading to an ideal progression for education, and related tools. The main part describes an existing class on system administration as given at the University of Applied Sciences (Fachhochschule, FH) Regensburg, Germany. It is analyzed for its history and target audience, the current curriculum, course layout, and for the didactic instruments used. The current situation is analyzed under the theoretical foundations given, and future directions for improvements are given.

Keywords: Applications in subject areas, Virtual Unix Lab, system administration, learning theory, instructional theory, instructional design, curriculum, Regensburg

Contents

1 Introduction **3**

2 Fundamentals of education **3**

2.1 Psychology and Learning Theory 3

2.2 Didactic Realisation, Instruction Theory & Instructional Design 7

2.3 Dimension of Implementation and Adoption 11

2.4 Alternative Learning-Theoretical Approaches 12

2.5 Education - Ideal Progression and Tools 14

3 Didactic Model of the System Administration Course **15**

3.1 History and Target Audience 16

3.2 Current Curriculum 16

3.3 Course Layout 25

3.4 Didactic Instruments 29

4 Analysis of the Current Situation **32**

5 Future directions **33**

References **35**

1 Introduction

This article discusses education of system administration. It starts by illustrating theoretical foundations of psychology, learning theory, instruction theory, and instructional design. This leads to an ideal progression for education, and related tools.

The main part describes an existing class on system administration as given at the University of Applied Sciences (Fachhochschule, FH) Regensburg, Germany. It is analyzed for its history and target audience, the current curriculum, course layout, and for the didactic instruments used. The current situation is analyzed under the theoretical foundations given, and future directions for improvements are given.

This paper is an update to what was previously published in german language as [Feyrer, 2005a] and [Feyrer, 2005b]. The topic is also discussed with less focus on technical details of the "System Administration" class in [Feyrer, 2008].

2 Fundamentals of education

There are several aspects to instructional research, which will be investigated in this paper. Firstly, psychology and learning theory are the basics which explain (or try to) how acquisition of new information works in the human mind. Secondly, didactic realisation is explained by instruction theory, which describes how to model information so that it is best fit for one of several learning theories, and thirdly instructional design determines how to prepare teaching material to fit for instructional and learning theories[1]. After looking at the various theories, dimensions of implementation and adaption will be discussed, which will lead back to another look at several alternative learning-theoretical approaches to consider, leading to an ideal progression of education and an optimal set of tools that will be applied to a course on system administration in the next chapter.

2.1 Psychology and Learning Theory

The process of human learning can be approached from two sides, philosophy and psychology. From the philosophical side, epistemology gives a view on knowledge and learning with it's impact on teaching[2]. On the other side, natural science, and psychology in particular, have recognized the relation of human

[1] [Kuyper, 1998] p. 49
[2] [Hammer and Elby, 2000] p. 2

learning with their subject early, and have developed theories of learning, which in turn were used to develop theories of instruction and instructional designs[1].

We will look at a short overview of the various aspects of teaching here, starting with the psychological learning theories and moving on the possible realisations, with the next section containing more details on this.

Behaviourism: Early learning theories go back to Iwan Pawlow, who observed "conditional reflexes" at his famous drooling-dog experiment[2], and Edward Thorndike who found laws about learning by trial and error by experimenting with cats[3]. John Watson and Burrhus Skinner picked up their works and while Watson coined the term "behaviourist" in his article "Psychology as the Behaviourist Views it"[4], Skinner made experiments with operant conditioning of pigeons[5]. Skinner was also influenced by Sidney Pressey's "testing and learning" machines[6,7], and in collaboration with James Holland he worked on a a "teaching machine"[8], which led him to define the term and concepts of "programmed teaching"[9].

Based on these fundamentals as well as Norbert Wiener's theories of cybernetics[10] and Helmar Frank's "cybernetic pedagogy"[11], it was hoped that teaching could be automated with the aid of machines (computers), so that human learning can be guided in a better way[12].

The focus of behaviouristic approaches to teaching is to teach information, give the learner time to understand, then ask a question on the subject taught, and give feedback based on the quality of the reaction. The whole process can be seen in image 1[13,14].

By repeating this sequence, simple tasks can be trained quite well, as Pawlow has shown, and the same method works for training humans as well. A few examples on how to design instructions to fit the behaviouristic learning will be introduced in section 2.2.

The behaviouristic learning theory is most appropriate for small learning steps, and bigger learning goals have to be split into several smaller goals,

[1] [Kuyper, 1998] p. 49
[2] [Pawlow, 1972] pp. 203
[3] [Thorndike, 1911] Chapter 2. "Animal Intelligence"
[4] [Watson, 1913] pp. 158
[5] [Skinner, 1947] p. 168ff
[6] [Pressey, 1926]
[7] [Pressey, 1927]
[8] [Holland and Skinner, 1961] p. V
[9] [Skinner, 1968]
[10] [Wiener, 1948] pp. 11
[11] [Frank, 1969]
[12] [Seidel and Lipsmeier, 1989] p. 32
[13] [Nösekabel, NA] p. 6, Figure 2
[14] [Kerres, 1998] p. 46

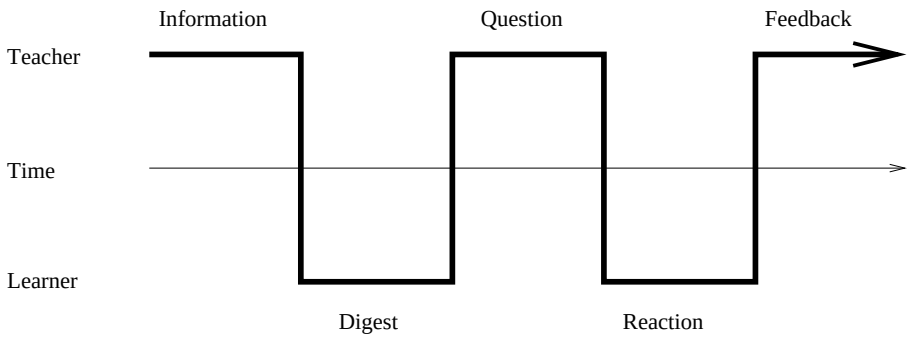

Figure 1: Behaviouristic approach of teaching

which are usually presented in a sequential manner[1].

Critics of behaviourism point out that the approach doesn't consider the individual nature of human beings enough, as Watson described there is "no dividing line between man and brute"[2]. This led to development of metacognition and cognitivism as learning theories[3].

Cognitivism: The concept of cognitivism goes back to the early days of the 20th century, and notable names are Jean Piaget, Edward Tolman, Jerome Brunners and Wolfgang Köhler[4].

The idea in cognitivism is to view the learner as individual, which can process external stimulus on it's own, and not only react to it. As such, the learner behaves as an interactive receiver of messages that contain news and knowledge in the sense of Shannon and Weaver's communication theory, and messages can be carried in various media[5]. Learning is considered a creative process of problem solving[6], and Piaget proposed that the learner adapts to the problem domain and solves it using the concepts of assimilation and accommodation[7]. In this context, accommodation and assimilation mean to adjust the learned cognitive concepts to new environments, and to match new external objects and conditions to the individual's internal structure by modifying the existing cognitive structures[8]. Cognitive development happens through both external influence by learning material and internal influence by the learner's existing cognitive structures. The learner's "knowledge" is considered the sum of all the patterns of recognition, understanding and processing available to the individual, including

[1] [Tulodziecki, 2000] pp. 57
[2] [Watson, 1913] p. 158
[3] [Seidel and Lipsmeier, 1989] pp. 36
[4] [Seidel and Lipsmeier, 1989] p. 26
[5] [Shannon and Weaver, 1949] pp. 31
[6] [Seidel and Lipsmeier, 1989] p. 26
[7] [Piaget, 1967] pp. 7
[8] [Schulmeister, 1997] p. 71

it's environment[1].

A number of ways to model instructions after constructivist approaches will be illustrated in section 2.2.

The cognitivistic learning theory is best used for complex subjects that go beyond pure factual knowledge i.e. depending on the learning goals, constructivistic approaches can be used to[2]

- determine which kind of knowledge structures to build up.
  Cognitive theories can not only be used for learning simple facts (declarative knowledge), but also rules (procedural knowledge) and concepts (contextual knowledge).
- determine how knowledge is stored in the brain.
  Various theories can be applied here as well, for example by giving the subject taught a context with other subjects that it can be associated with (theory of meaning structures), explaining concepts both in words and non-verbal (dual encoding theory) or trying analyze all structural and functional components (theory of mental models).
- determine if specific topics or general strategies should be learned.
  Specific topics include various areas of science which provide an intellectual challenge as for example system administration. In contrast, it is also possible to teach general development aspects, e.g. social or moral considerations.

Leaving the various possibilities that the constructivist learning theory offers aside, the concept is still based on the interaction between external medial presentation and internal processing.

Constructivism: The constructivist learning theory is based on works by a number of philosophers, among them Jean Piaget. The central thesis is that cognition is construction and interpretation, and that objective, subject-independent learning and understanding is not possible[3]. As such, it goes one step further than cognitive theory: constructivism emphasizes the "individual" component like experience and way of thinking first found in cognitivism even more, to a point where it does not include any external instructions to the learner. Instead, the idea is to act freely in an environment and construct new knowledge from existing knowledge and interpretations of feedback given to various actions in an act of recognition, which is individual to each subject.

Due to this subjective nature, there is no "best" way of teaching. Instead, learning happens by actively dealing with tasks, which provides a context to the learning process, making acquired knowledge context-bound or "situated". This approach also prevents "inert knowledge", i.e. knowledge that was once learned, but cannot be applied in a given situation as there

[1] [Tulodziecki, 2000] p. 58
[2] [Tulodziecki, 2000] pp. 58
[3] [Bruns and Gajewski, 2002] p. 14

is no mental connection between the context given by the situation and the knowledge needed to be applied[1].

During the learning process, knowledge is created dynamically and not stored in a fixed way. Due to this, knowledge cannot be passed on without repeating the same learning process in the receiving learner, which has to re-construct that knowledge[2].

The "creation" of knowledge can also be improved by encouraging communication between students and a teacher or in a learning group among themselves, which allows changing role and perspective. That way, the the classical roles between teacher and student aren't sharply defined any more, and it becomes clear that social interaction between learners is an important part of constructivism[3].

There are several approaches[4] to model "instruction" (put into parentheses here, as there isn't really a concept of instruction in constructivism), among them the concept of cognitive apprenticeship, knowledge communities and cognitive tools. Some of these will will be discussed in section 2.2.

As a summary, it can be said that constructivistic approaches are good for approaching complex subjects and learning goals, as it goes far beyond the cognitive teach-review cycle. There are downsides though, which become obvious when looking at the didactic realisation.

This section introduces three fundamental learning theories with some of their basic ideas. Not each theory is ideal for teaching each subject. This needs to be considered when approaching the didactic realisation of a teaching system, which is what the next section will cover.

2.2 Didactic Realisation, Instruction Theory & Instructional Design

The psychological and pedagogical foundations given in the learning theories need to be applied to realize learning systems, which is covered in instruction theory and in instructional design. Starting from the three learning theories introduced in the previous section, some methods for realizing them should be introduced here.

This section only gives an overview to gain judgement over the methods needed to teach system administration in the next sections. Related introductory texts on the subject can be found in [Eikenbusch and Leuders, 2004, pp. 153] and [Wiggins,

[1] [Bruns and Gajewski, 2002] S.15
[2] [Schulmeister, 1997] pp. 73
[3] [Bruns and Gajewski, 2002] pp. 15
[4] [Schulmeister, 1997] pp. 81

1989], an in-depth coverage of the topic can be found in [Gagné, 1967], [Gagné and Briggs, 1974], [Richey, 1986], [Schulmeister, 1997] and [Reigeluth, 1983].

Behaviourism: The "instruction paradigm" provides a realization of the behaviouristic learning theory. It assigns the learner a passive (but still important) role of receiving and processing information and instruction given by a tutor, teacher or a teaching program. After each learning unit, feedback is provided before moving on to the next unit[1]. This kind of instruction is also known as "drill & practice" due to it's main components, and it provides the basic concept behind programmed teaching[2].

Big lectures are split into small learning atoms my experts on the subject. It is possible to teach the learning atoms in several ways, either by classroom teaching in a class, by providing it in book form or via a computer program. Except in the former case, the learner can decide on his own about the speed that he progresses[3].

With the aid of computer programs, it's even possible to deny the progress to later learning units until prior ones were mastered successfully, where some system is needed to assert the "success" in that case. This is reflected in the TOTE-model developed by Miller, Galanter and Bram in 1960, which consists of four phases illustrated in figure 2, Test, Operation, Test and Exit. First a condition is tested, and unless it's satisfied, a learning operation has to happen. This is repeated until success of the test is indicated by convergence towards a pre-defined goal, which then leads to an exit of the procedure[4].

Cognitivism: As an alternative to the instruction paradigm, the "problem solving paradigm" corresponds to realisation of a cognitivistic learning theory. The central idea is to provide an environment where learners can search their own challenges within an open learning environment, or solve given problems with no clear description of how to solve them. That way, learners are encouraged to find their own solutions, and construct new knowledge from the one they already have and the tools and information available in the learning environment[5].

Environments that encourage this kind of learning are the "explorative learning" as described in [Bruner, 1961] and as a special case the microworlds described by Seymour Papert. A microworld in this context means a small ("micro") environment ("world") with a fixed set of rules, within which a given task should be solved. The probably best-known microworld includes the "logo" programming language. Logo allows teaching procedures, interaction and list processing on one side, but as it also provides a facility to

[1] [Bruns and Gajewski, 2002] p. 32
[2] [Seidel and Lipsmeier, 1989] p. 40
[3] [Kerres, 1998] p. 49
[4] [Seidel and Lipsmeier, 1989] p. 28
[5] [Bruns and Gajewski, 2002] p. 32

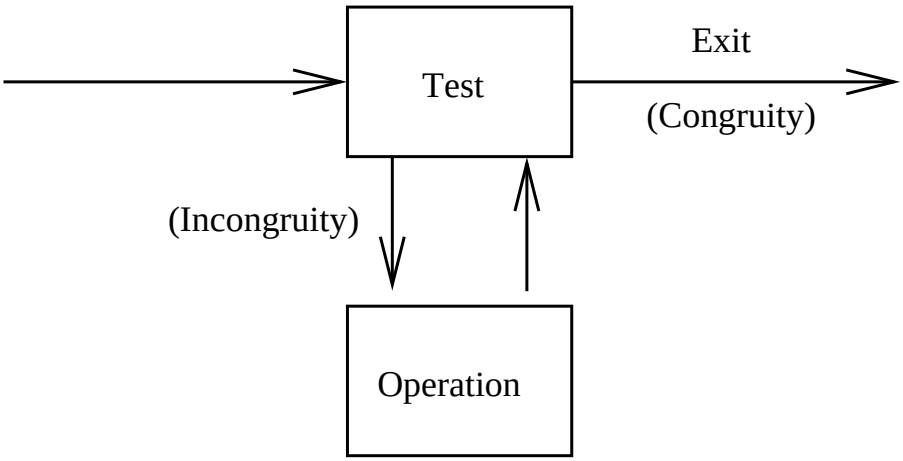

Figure 2: The TOTE model

move a turtle across the screen in a way described by the user, it can also be used to teach basic concepts of computer science and programming[1].

In general, several kinds of tasks can be requested from the learner, depending on the kind of knowledge that he should build up. On the one hand, correlations can best be learned by predicting the behaviour of the system when changing various parameters. On the other hand, the task can be to explain which parameters need changing to achieve a certain condition of the system, and problems can be solved by choosing the proper conditions and changes for a given effect[2].

Another approaches to realize cognitive learning theories is by not dealing with a particular subject either directly or via some (possibly simulated) interface, but by talking about it. For this approach, a teacher or tutor is needed to ask questions that the learner answers, and the most well-known form of this is known as the "Socratic dialogue". By considering all aspects of a certain subject, unknown areas will be discovered, and by more interaction with a guiding instance, relationship to existing can be used to build up new mental connections and thus knowledge[3].

Constructivism: Verbalizing as described in the prior paragraphs helps the learner order vague concepts that are mentally present and internally connected to a subject into a clear form needed for communication. Learning in dialogue with a teacher or tutor or with other learners in a learning group is not only good for working on given tasks. Due to the changes in role and position required for participants of a learning group, it goes beyond the given tasks and encourages constructing new knowledge. Tools found useful for supporting communication can be divided into synchronous and asynchronous

[1] [Papert, 1982] p. 152
[2] [Kuyper, 1998] p. 53
[3] [Bruns and Gajewski, 2002] p. 31

groups[1]:

- Asynchronous communication tools include email, discussion forums and electronic bulletin board systems, user galleries, and facilities for giving feedback on existing material.
- Synchronous communication tools include video- and audio-conferencing, application sharing, interactive whiteboards, chat, and instant messaging,

Similar to cognitivism, the approach taken in constructivism is to follow the "problem solving paradigm"[2], but given the basic idea behind constructivism that no instructions are given at all, the learning environment needs to be a much more flexible, providing a framework to create reflections of learned knowledge in various media, communicate and cooperate with others and construct new media from existing one[3].

To allow learners free navigation in a wealth of information, a display of information is necessary that connects all information with everything related to it, effectively building a hypertext or even hypermedia structure as can be found (to some extent) on "the Internet" today[4], i.e. having a tightly interconnected system like the World Wide Web which also allows references to other resources of information as can be found in electronic libraries, discussion groups as found on the Usenet etc., and which accommodates a wide variety of media formats, including (hyperlinked) text, images, audio and video[5].

Methodical elements to provide these media and various way for accessing them include hyperlinked content, guided tours, a document pool, encyclopedia, interactive exercises, business games and map exercises, simulations, online tests, movies and help-functions for all these facilities[6].

Besides offering an environment for communication, collaboration, navigation and construction, learning systems like microworlds, simulations and intelligent tutoring systems (ITS) offer interaction within an environment for creating new knowledge from various views on new and existing knowledge gained through them. Rolf Schulmeister describes a number of projects realizing these ideas[7,8], illustrating them would be beyond the focus of this section.

This section covered various methods which can be employed to realize various learning theories. There is a variety of options to do so, and the effort to do so is

[1] [Bruns and Gajewski, 2002] pp. 48
[2] [Bruns and Gajewski, 2002] p. 32
[3] [Bruns and Gajewski, 2002] p. 16
[4] [Schulmeister, 1997] p. 78
[5] [Schulmeister, 1997] p. 19
[6] [Bruns and Gajewski, 2002] pp. 44
[7] [Schulmeister, 1997] pp. 177, 225, 375
[8] [Schulmeister, 2002b] pp. 16, 178, 241

wildly varying. There are various levels at which these methods can be integrated into teaching environments, which is covered in the next section.

2.3 Dimension of Implementation and Adoption

In 1929, Edward Thorndike and Arthur Gates pondered "*If, by a miracle of mechanical ingenuity, a book could be so arranged that only to him who had done what was directed on page one would page two become visible, and so on, much that now requires personal instruction could be managed by print*"[1]. Looking at this idea from today's perspective, it's obvious that one would use a computer to construct a book with these constraints to realize Thorndike's idea.

Offering a guided tour through a book is only one of several methodical forms for teaching. The spectrum ranges from pure classroom teaching as performed in the current form of the "System Administration" lecture, described in the section 3 below, over a mixture between presence teaching with virtual components to pure virtual teaching as e.g. offered by the "Virtuelle Hochschule Bayern" (VHB)[2] and other[3]. When employing methods for online learning, various degrees like self-paced online learning, collaborative online learning (tele-tutoring), live online learning (tele-teaching) exist[4].

The central entity to look at here is the *learning environment*, in which teaching and learning happens[5]. Depending on the kind of education and the learning theory applied in it, various methodical communicative elements can be used, as described in the previous section, 2.2.

One component of the learning environment wasn't covered so far, which is the *feedback* given to learners as a reaction to some task given to them to solve[6]. While existing learning platforms often use multiple choice texts and gaps in a text to fill in, all these test forms train basic behaviouristic learning and no real understanding of concepts[7]. More advanced concepts like interactive maps and images, or feedback on a given scenario that the learner was asked to create are rarely, if at all, found, even if these advanced ways for evaluation and feedback were more appropriate for advanced contents taught via realisations of cognitivistic and constructivistic learning approaches[8]. On the other hand, systems implementing these methods like simulations or microworlds often don't include

[1] [Holland and Skinner, 1961] p. V
[2] [Virtuelle Hochschule Bayern, 2001]
[3] [Schulmeister, 2002b] pp. 228
[4] [Bruns and Gajewski, 2002] pp. 39
[5] [Schulmeister, 2002b] pp. 6
[6] [Schulmeister, 1997] pp. 109
[7] [Seidel and Lipsmeier, 1989] pp. 53
[8] [Schulmeister, 2002b] p. 154

any components for evaluation at all[1].

Comparing learning theories in general[2] and their realisation, esp. virtual, computer based ones, in particular[3], one comes to the conclusion that approaches following constructivistic learning theories are best, but that their realisation is just as hard, which is why most of the realisations found so far are incomplete, non-working or otherwise insufficient. Schulmeister goes as far as judging that if constructivism continues to hold up its excessively high pretenses, it best doesn't show up at school any more[4].

Two conclusions can be drawn from this: First that not all subjects can be taught by using virtual teaching[5], and second that realisation of constructivistic approaches, esp. ones which defeat the instructional components of the learning process, may not be the best, and that there is room for alternatives[6,7].

2.4 Alternative Learning-Theoretical Approaches

There are ups and downs of the various learning theories and the instruction designs resulting from them as has been discussed in the previous section. Recognizing this, a number of approaches were suggested that take a pragmatic position between cognitivistic and constructivistic approaches[8]. The instructional design of the 2nd generation combines elements of constructivism like explorative learning and communication with elements of cognitivism, which intents to add new components into the learner's existing knowledge structures, intending an integration of new contents[9]. The learner should not be made a reactive entity, but rather act proactive, with proactive learning concepts striving for a change of the pedagogical situation towards choices for the learner, room for own arrangements and self-organisation of the learning process[10].

Merrill laid the fundamentals in instruction design with his introduction of "Second Generation Instructional Design (ID_2)". Merrill's ID_2 is based on Robert Gagné's work which assumes that different internal external conditions are necessary to promote different kind of learner outcome, but ID_2 also tries to overcome the limitations of what Merrill calls the "First Generation Instructional Design

[1] [Schulmeister, 2002b] p. 223
[2] [Schuman, 2007] "What are the problems and strengths of these theories?"
[3] [Schulmeister, 2002b] p. 223
[4] [Schulmeister, 1997] pp. 220, 223
[5] [Schulmeister, 2002b] p. 160
[6] [Schulmeister, 2002a] p. 79
[7] [Merrill et al., 1991] pp. 3
[8] [Tulodziecki, 2000] pp. 59
[9] [Bruns and Gajewski, 2002] S. 17
[10] [Weidenmann, 1993] pp. 11

(ID_1)" by integrating sets of knowledge and skills, producing pedagogical guidelines, selecting and sequencing instructional transaction sets and esp. integrating phases of instructional design[1].

The following components are part of ID_2[2]:

1. a theoretical base that organizes knowledge about instructional design and defines methodology for performing instructional design
2. a knowledge base for domain knowledge, for making instructional decisions
3. a series of intelligent computer-based design tools for knowledge analysis and acquisition, strategy analysis and transaction generation and configuration
4. a collection of mini-experts with small knowledge bases for one or more instructional design decisions each
5. a library of instructional transactions, with interfaces to add new transactions
6. an online intelligent advisor program that dynamically customizes the instruction during delivery, based on a mixed-initiative dialog with the student

The interfaces mentioned in item 5 are important in learning systems that incorporate many sources of teaching materials, teachers and subjects taught. For the present discussion the topic of interfaces and formats of meta-data describing them to facilitate easy exchange of data are beyond the scope, more information can be found on pages 202 and 207 of [Schulmeister, 2002b]. The advisor program and dynamic customisations of instruction mentioned in item 6 will be addressed later when discussing tutoring systems, personalisation and user-adaptive systems.

Another approach to address the named problems can be found in the concepts of situated cognition and situated learning.

Situated cognition assumes that thinking and learning are bound to a certain context, in which knowledge is learned. This context is defined by the learning environment, which also includes and defines the goals that should be learned, as learning is most efficient when the learning goals are known to the learner[3].

Situated learning goes into more details. It emphasizes the fact that an individual's learning performance is not only affected by the content presented and his

[1] [Merrill et al., 1991] p. 9
[2] [Merrill et al., 1991] p. 10
[3] [Schulmeister, 1997] p. 78

internal learning processes, but also by the context in which the learning material is presented. Real world examples are considered important, so the acquired knowledge and problems solving methods can be applied[1]. At the same time, a variety of examples should be used, to achieve decontextualisation. Following the concept of situated cognition, situated learning is employed in a learning environment which gives contextual information as well as instructions on goals to achieve, providing methods to reach the goals and also emphasizes social interaction to facilitate elaboration and reflection[2].

Depending on the extent to which virtualisation should be employed in situated learning, i.e. whether a "teacher" is present either in real of in the form of a computer program, or if there is no guide, and how strong didactic embedding is, there are a number of approaches which realize situated learning, and which all use multimedia technology to various degrees[3]. The approaches range from a teacher/student relationship in "Cognitive Apprenticeship" over learning in groups in "Knowledge Communities" to using cognition-promoting tools in the "Cognitive Tools" theory[4].

The difference in situated learning to a pure constructivist approach is that individuals are both guided in what they should learn as well as being provided with an environment that promotes solution of the given problems. The difference to cognitivistic learning approach is that more emphasis is put on the learning context and elaboration, not only acknowledging the (internal) individual character of the learner, but also providing more (external) context for learning. As such, situated learning is placed between cognitivistic and constructivistic teaching approaches.

This section covered various alternative approaches to learning theory and instruction design. Using a synthesis of the "classical" approaches and their derived forms together, a set of powerful teaching tools is available, and it is possible to achieve teaching methods that are considered ideal in traditional education, as outlined in the next section.

2.5 Education - Ideal Progression and Tools

A number of structures for instructional design of lectures have been proposed[5], an ideal course of teaching is considered of the following steps[6]:

[1] [Lave and Wenger, 1991] pp. 32
[2] [Mandl et al., 1994] p. 170
[3] [Mandl et al., 1994] pp. 171
[4] [Schulmeister, 1997] p. 78
[5] [Clark, 2000]
[6] [Tulodziecki, 2000] pp. 62

1. a collection of assignments, collecting and discussing spontaneous ideas for solving
2. defining learning goals and discussing their meaning
3. communication about proceeding towards these goals
4. acquiring fundamentals needed to solve the assignment
5. putting the assignment into effect
6. comparing various solutions, and summarizing what has been learned
7. introduction of and working on domain specific assignments
8. discussing the knowledge learned and the ways it was learned

Discussion so far has named a number of instruments to realize various learning theories and instructional designs. A number of these instruments can be used to shift the focus from the result of the learning process to the learning process itself[1]:

- empowering learning environments, to promote creativity
- games to increase motivation
- cognitive tools to promote understanding and representation of cognitive processes
- tools to support writing and reasoning
- programs to support reflection of the mental processes of the learner

The above lists are guidelines for realizing learning environments. This section has discussed the various learning theories and instruction designs resulting from them, in which circumstances to use one over another, or maybe even a mixture of several approaches to gain a maximum benefit from all approaches.

3 Didactic Model of the System Administration Course

Discussion has been kept on a theoretical level in the previous section. This section will look at an existing lecture for “System Administration” that is part of the mandatory curriculum for studying computer science at the University of Applied Sciences (“Fachhochschule”) Regensburg for several years now.

This section will outline the history and target audience of the lecture, outline the contents of the current curriculum as well as the layout that the components are arranged in, and discuss the didactic instruments used so far.

[1] [Schulmeister, 1997] pp. 79

3.1 History and Target Audience

The "System Administration" course is offered to students of computer science at the University of Applied Sciences ("Fachhochschule") Regensburg. It was started as optional course for students of in the advanced study period, i.e. usually in the 7th or 8th semester, by Prof. Jürgen Sauer in 1994, and held until 1998. Since 1999, the course was given by Dipl.-Inf. Hubert Feyrer also as an optional course to students of the advanced study period in 7th and 8th semester. Starting since 2003, the course was added as mandatory. This discussion only covers the lecture in it's mandatory form as it is given since 2003.

The target audience of the "System Administration" course are students of general computer science ("Allgemeine Informatik") in the advanced study period, usually in their 5th semester. Volunteer students from technical computer science ("Technische Informatik") or economical computer science ("Wirtschaftsinformatik") are allowed to participate and take the course as optional lecture.

3.2 Current Curriculum

The course consists of two lectures and one lab exercise a week, with one lecture and lab exercises being 90 minute each. For lab exercises, the students are split up in two groups due to lack of sufficient working places. On average, a course consists of 40 students, resulting in two groups of 20 students each.

Given other focuses in the curriculum and in basic education, Students usually have little Unix knowledge. Some experience is available for understanding of the basic operating system and networking principles from corresponding courses, but routine in using the Unix operating system, it's commands as well as concepts for automating tasks are not available. As such, part of the lecture repeats basic commands and concepts of the Unix operating system focusing on the latter use for system administrative tasks.

The following topics are covered in the lecture and accompanying lab exercises[1]:

0. **Introduction[2]:** The introduction of the lecture gives a small historical overview of the past lecturers and tutors, intended use of the online script and exercises as well as a description of the overall goal of the lecture.
1. **Historical Overview[3]:** This section illustrates the history of Unix, starting with AT&T and going to BSD and the various commercial and free systems

[1] [Feyrer, 2007b]
[2] [Feyrer, 2007b] "Vorwort"
[3] [Feyrer, 2007b] "Historischer Überblick"

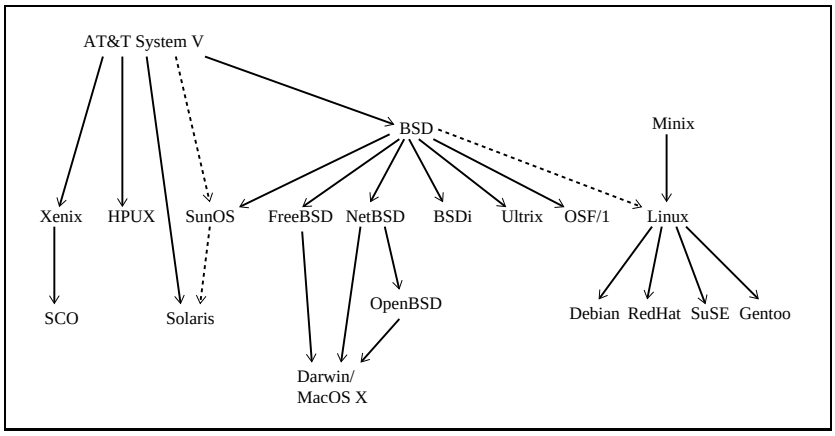

Figure 3: History: A brief overview of the Unix history

derived from it, as well as where Linux stands in that scheme. Figure 3 gives a brief illustration of the historical development introduced in more detail in the lecture.

Exercises cover comparison of various Unix systems using the "Rosetta Stone for Unix"[1], and looking at descriptions of commands in standards like POSIX and the Single Unix Specification[2].

2. **Login process, process correlation[3]:** As an introduction to the topic of system administration, the classical login process is discussed, including the processes involved, their correlation and the general Unix process model shown in figure 4 as well as the concepts of signals and job control. As a side note, handling of Unix manual pages, manpage sections and notation for writing them are covered.

3. **User commands (standalone and for shell programming)**[4] Assuming that only few students have a sound Unix background, basic Unix commands are discussed to cover a number of areas which are useful both when used alone as well as for using them in shell programming later. Areas covered include general syntax of commands, managing files and directories, permissions and access control in a multiuser environment, principles behind commands like "ls(1)" and "rm(1)" as well as text processing using regular expressions.

   Exercises help students getting familiar with the basic commands, general use and options available. Commands are practiced for getting information on any of the discussed commands, applying them in some simple scenarios to run given commands, understand their output, and find appropriate descriptions in documentation for unknown topics.

[1] [Hamilton, 2007]
[2] [The Open Group, 2004]
[3] [Feyrer, 2007b] "Login Prozeß, Prozeßzusammenhänge"
[4] [Feyrer, 2007b] "Hilfsprogramme (Standalone und für Shell-Programmierung"

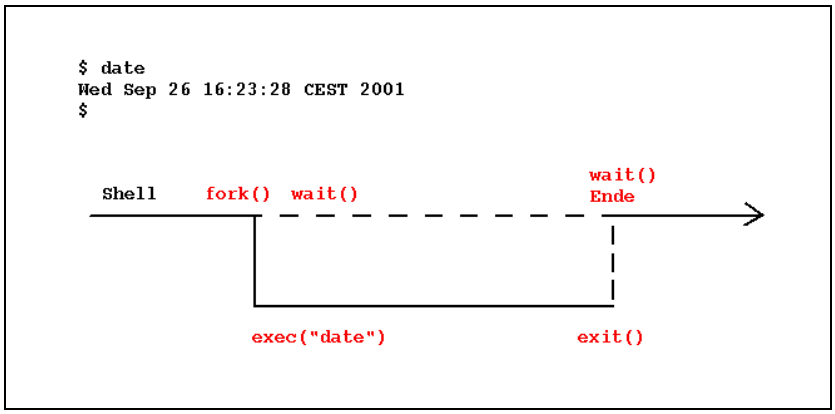

Figure 4: Login process: The Unix process model

4. **Information about the system**[1]**:** To properly administrate and tune a system, it is essential to know as much data about the system's state as possible. This section covers a number of areas that may be of interest for adjusting or troubleshooting, including commands, output usually found and how to interpret it. The areas covered are processes, signals, users, installed software, system, kernel, terminals, remote machines, swap-space, process accounting, filesystems, disk quotas, device-handling and harddisks. Special emphasis is given on displaying various Unix versions like Solaris and various Linux and BSD variants, to make sure students can identify general concepts and take note which commands are platform specific.

   Exercises comprise a number of questions about finding certain kinds of information about the named areas, and students are expected to either know which program to use to find the information, or a certain command (usually with some non-standard options) is given and the output should be interpreted.

5. **Shell programming**[2]**:** Assuming a basic understanding of both use and analysing a Unix system, this chapter introduces shell programming using the Bourne shell (/bin/sh), to automate common and recurring tasks. A general understanding of scripting and alternatives to the Bourne shell is given by discussing general requirements for execution of programs and alternatives to the Bourne shell. After a brief introduction of the various startup files read by a shell, the features of the classic Bourne shell are introduced, including redirection of input and output, expansion of wildcards, shell and environment variable, quotes, control structures and shell functions.

   Exercises for this section mostly consist of programming tasks to analyse output of programs automatically, or show how to write non-trivial scripts in several steps, leading students from a simple version to more sophisticated ones. Programs to write are for recursive copying a directory structure

[1] [Feyrer, 2007b] "Informationen über das System"
[2] [Feyrer, 2007b] "Shellprogrammierung"

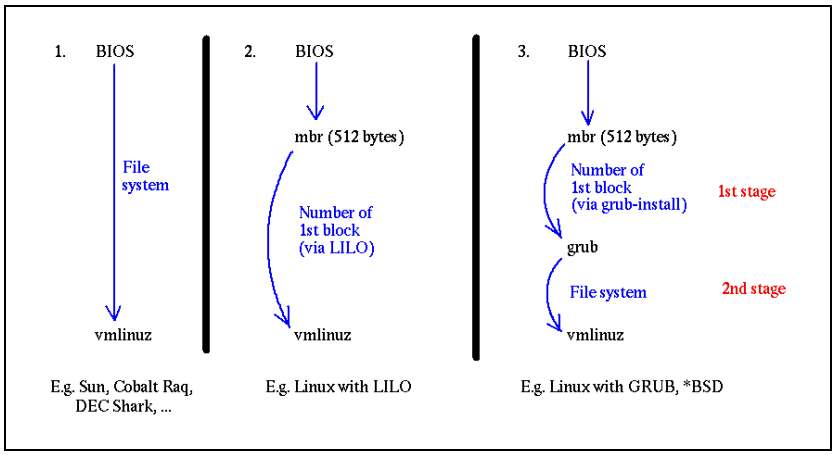

Figure 5: Booting: Various boot models

(without using existing tools to do so), and a script to add a user to a copy of the system's user database.

6. **Application of shell scripts: booting and shutdown**[1]**:** Students have been introduced to all the features that are available in shell programming, and have also written several scripts on their own, to make them familiar with concepts like variables, protection of arguments and control structures. This section shows an application of shell programming by observing the system's startup mechanism, which is usually realized as a set of shell scripts. The approach of letting students read existing code written by experts instead of writing their own is intended to show solutions for common problems and also practice reading and understanding of flow of code and data. The section introduces general booting of systems with varying intelligence in firmware aided by one or more bootloaders as shown in figure 5, and outlines the System V "init"-system briefly displayed in figure 6 then. Attributes of the init-system discussed include runlevels, concept and functionality provided by start- and stop-scripts as well as their layout in the filesystem. After the System V "init" system, alternative approaches for disabling/enabling of services and determination of order in which to start services are discussed, introducing the mechanisms used on BSD and Irix systems as well as the hybrid approach found on SuSE Linux.

   Exercises for this section are mostly of analytical nature, as changing the system's boot system to gain experience would require system privileges, which cannot be handed out - in order to assure proper system operation, the machines would have to be re-installed after boot, which is not an option, unfortunately. As such, exercises include analyzing the existing startup systems found on Solaris, SuSE Linux and NetBSD including the documentation, start-/stop-scripts and the framework used to determine order of

[1] [Feyrer, 2007b] "Anwendung von Shellscripten: Hoch- und Runterfahren des Systems"

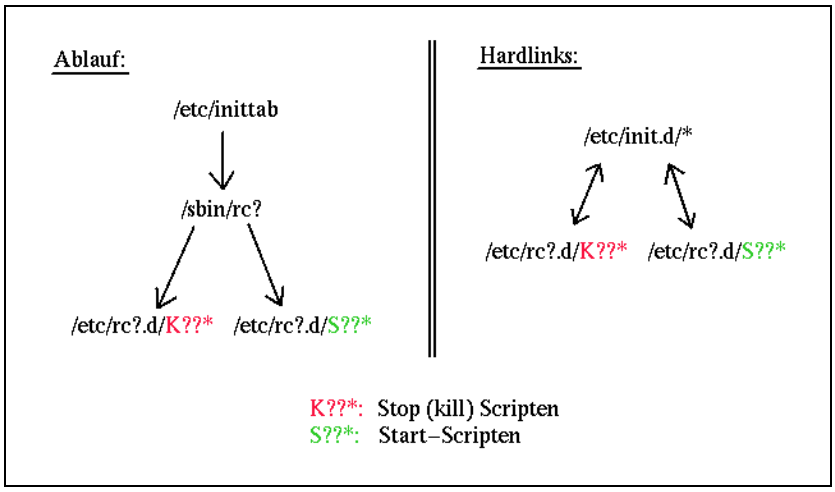

Figure 6: Booting: The System V Init-Concept

the scripts and if they should be ran at all.

7. **Networking**[1]**:** After shell scripting and system booting, the next big complex covered is networking. As students in the 5th semester have the "Data Communications" lecture in parallel with the system administration lecture, a basic understanding of networking and TCP/IP basics can be assumed. For the sake of completeness, the administrative parts covered later are first repeated, to bring all students to an equal base. The basics recalled are addressing, routing and name service concepts. Building upon these basics, the network model of Unix is explained, again covering various implementations with an emphasis on Solaris, but also Linux and NetBSD. With the network model understood, next steps covered are how to configure the system and name resolving for basic TCP/IP networking, followed by an overview over existing TCP/IP based services with some discussion on their use. Following this introduction, three topics are picked up that are considered important when managing clusters of workstations: in the Secure Shell (ssh), setup of public key authentication is explained, and to manage homogeneous clusters of workstations, the Network File System (NFS) and Network Information System (NIS) are covered, including setup details for both clients and servers.

   For practical exercises, the same situation as described above in "Booting and shutdown" applies: for maximum learning effect, system privileges would be needed, but they cannot be handed out to students. As such, exercises mostly consist of analysis of existing Solaris, Linux and NetBSD systems and the differences between them, covering information kept in systems for the various levels of the OSI stack, routing and name service configuration.

[1] [Feyrer, 2007b] "Networking"

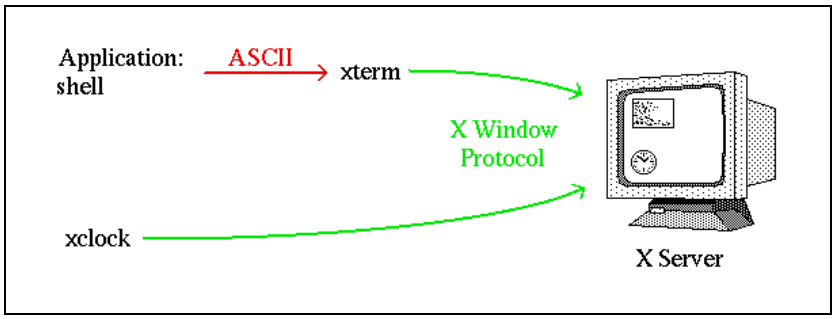

Figure 7: The X Window System: Application (client) and presentation (X) server

8. **The X Window System**[1]**:** An application of networking that is not obvious at first is the X Window System, which is the graphical subsystem used on Unix to build modern desktop systems like KDE and GNOME on top of it. The interesting fact here is that the X Window System itself is network transparent, i.e. an application can run on one machine and display it's graphical output on another machine. The lecture starts with some basic concepts like the client/server concept as illustrated in figure 7[2], addressing displays, simple access control, and lists some of the standard clients that are part of the X Window System. Redirection of applications across the network is covered next, first using the mechanisms provided by the system, and then using the mechanisms provided by the secure shell. After the basic authentication methods for displaying windows on remove displays has been introduced, an advanced method is introduced, followed by the startup process of the X Window System including processes and files involved. Last, the functionality of a "Window Manager" is explained in the context of a demonstration the KDE desktop environment, displaying the various components of such a complex system and their interaction.

   Lab exercises for the X Window System start by telling the login system to not start any graphical user environment, and the build that environment step by step manually, starting with looking at various window managers, placing a number of client windows next, followed by tuning application look and feel. Redirecting clients from/to other systems is part of the next exercise steps, followed by playing with example configuration of the fvwm2 window manager. As playing with a bare X environment to setup complex environments is inconvenient, The Xnest X server should be explored to run an X server in a windows, which should then be used to investigate the components of the KDE and GNOME desktop environments.

9. **Security**[3]**:** Security is considered an important and ever-present need today, mostly requiring system administrators to take appropriate measures

[1] [Feyrer, 2007b] "Das X Window System"
[2] [Feyrer, 2001] "An X terminal emulation application – xterm"
[3] [Feyrer, 2007b] "Security"

to establish secure systems. The curriculum of computer science includes special lectures on security to give basic understanding and methods for handling. After giving an overview of the wide area of "security" and the various topics attached to it, the system administration lecture approaches "security" from the practical side by discussing what kinds of problems may exist, and how to approach them. The most important area covered is host security, with special attention being paid to data manipulation, unauthorized access to resources and privileges, covering both how to detect problems and how to solve them for each case. After host security is discussed in detail, network security is covered as well, showing that a number of problems have an equal origin in both host and network security. Special network security issues like sniffing, spoofing, break-ins via the network and distributed denial of service (DDoS) attacks are also covered. The lecture closes by pointing at various sources of information, from full disclosure sources over general security lists to vendor provided information to assist in securing systems.

Student exercises start by a briefing on the legal situation of computer security, and that any security holes found should be reported to the system administrator immediately. The lab systems should then be analyzed and monitored for the various classes of security problems discussed, followed by finding special system services that may get exploited. Special emphasise is put on examining and understanding stack/buffer overflows as illustrated in figure 8, which lead to many problems in both host and network based scenarios, and which students of computer science should know how to avoid when writing own programs.

10. **Practical Extraction and Report Language - perl**[1]**:** A language found often in system administration environments is perl, and the various incarnations of the user management systems used at the computer science and computing center's departments of the University of Applied Sciences ("Fachhochschule") Regensburg were all written in perl, which emphasizes this point. Assuming understanding of C/C++ or Java and Bourne shell programming, programmers new to perl can usually get to usable results in very short time. The introduction given to perl in the system administration lecture covers the difference to the named programming languages in data types, discusses input/output and control structures. Features only present in perl are discussed next, including processing of regular expressions, arrays, lists and stacks as well as hash tables. The perl programming language's built in functions as well as creating own functions and using existing modules including an overview of all existing modules round up the introduction to perl.

    Exercises for perl include programming tasks that handle lists and associative arrays, analyze web server protocol files and scan and sort mailboxes.

[1] [Feyrer, 2007b] "Practical Extraction and Report Language - perl"

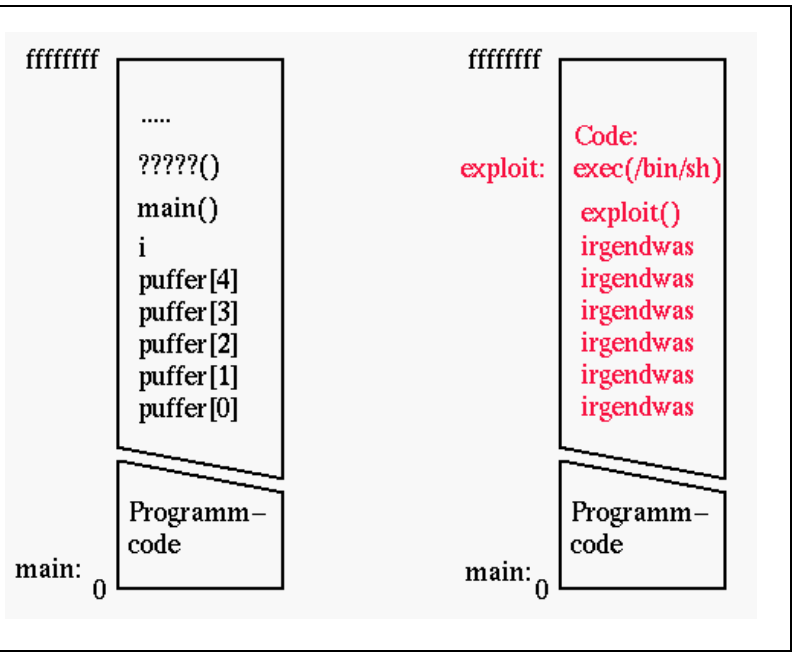

Figure 8: Security: Explaining stack/buffer overflows

11. **User management**[1]**:** After giving an general introduction to the perl programming language, user management is used as an example for demonstrating a moderately-sized perl script that's used at the department of computer science of the University of Applied Sciences ("Fachhochschule") Regensburg. The section approaches user management by repeating the related concepts in Unix, including user databases, password encryption, home-directories, dot-files, crontabs, quotas as well as site-specific setup steps like entries to access databases, create webpages, and so on. A number of standard tools available for both shell/script based access as well as GUI based tools are introduced next, including a short discussion on the advantages and disadvantages of these tools. The tools used for user management at the Fachhochschule Regensburg's computer science are then introduced to show an approach of large scale user handling (the computer science department has an average of 1.000 students which have access to various Unix machines), and at the same time as an example of how to realize user management in perl, giving students a chance to learn from existing perl code. The perl scripts are first discussed from a user/administrator's point of view, including how to call them and what command line options are available and then outline implementation of the scripts including a walk-through of the various scripts and modules.

    Exercises are unfortunately restricted in this exercise again, as the students can't work with system administrator (root) privileges. As such, the exer-

[1] [Feyrer, 2007b] "Benutzerverwaltung"

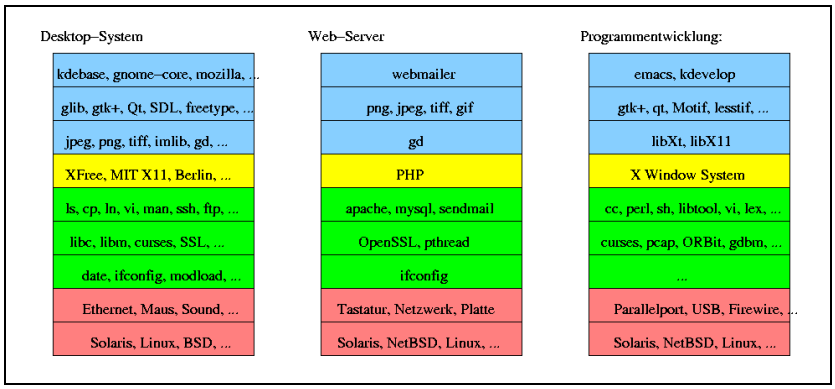

Figure 9: Software management: various layers in theory

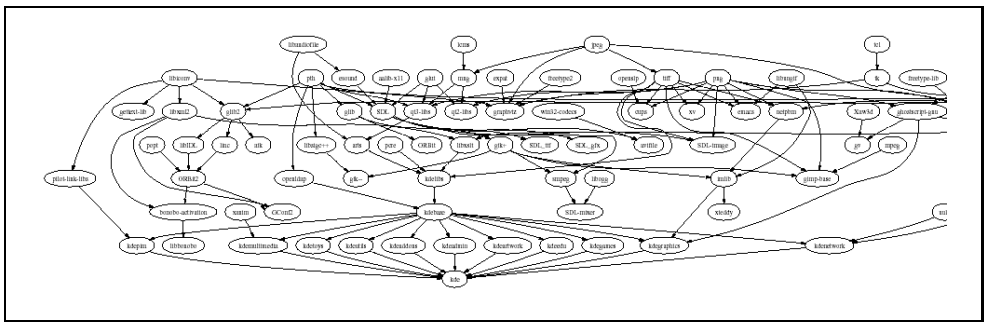

Figure 10: Software management: components found in practice

cises consist of examining various operating systems' tools via their documentation and to the extent possible with normal user privileges.

12. **Software management**[1]**:** After describing system operations and user management, handling application software for users is the third big topic covered in the "System Administration" lecture. This section introduces the software architecture found on operating systems including separation into "operating system" and "applications" at various fine-grained levels (see figure 9), and describes the components found in theory and their interconnections found in practice, see figure 10. The historical development that led to the various models and components is explained. Theoretical steps of software installation are covered next, followed by in-depth discussion of handling of precompiled binary software using software management tools found on Solaris, Linux Systems using the RedHat Package Management (RPM) system and NetBSD. Source-based methods of software installation are left out on purpose, they are covered in a different lecture[2].

    Exercises for software management involve getting familiar with the various ways of handling binary packages on various systems with no root

[1] [Feyrer, 2007b] "Software-Management"
[2] [Feyrer, 2007a] "Source-Pakete"

privileges, which is only partly given due to the mentioned restrictions of not being able to hand administrative privileges to students. Doing software installation as "normal" user is considered good practice, and enforced in this case. After getting familiar with various package systems, the meta data used by these systems is investigated, and dependencies between software packages gets analyzed in textual and graphical way.

13. **Backups**[1]**:** The last section of the "System Administration" lecture covers backing up data. Topics discussed include media presently available on which to backups, various concepts of backups from file based to whole filesystems, tools commonly found on Unix systems to do various forms of backups as well as compressing data for efficient use of backup media. An overview of integrated solutions including commercial backup systems ends the lecture.

    Lacking not only access to systems on a filesystem/harddisk base, but also backup hardware and enterprise solutions for performing backups, practical exercises are kept on the base of backing up single files and directories.

This section has described the various topics covered in the "System Administration" lecture with a focus on the contents taught in class and the lab exercises students are expected to do. The reasons for ordering sections in the given already indicated, and the next section will give more information on the overall layout of the course and the reasons behind that.

3.3 Course Layout

After the contents of the single lectures of the "System Administration" lecture were described in detail in the last section, this section illustrates the overall building blocks of the lecture, how they are arranged to reach the goal of the lecture, and also how the building blocks reflect the change demanded in learning strategies, from behaviouristic learning for the basics to cognitivistic learning for the more advanced topics.

There are two goals of the lecture:

1. General understanding of Unix, and that there is not "the" Unix system, but several implementations.
2. System management of large scale clusters, with system, user and software management as well as procedures to setup the necessary network infrastructure

[1] [Feyrer, 2007b] "Datensicherung"

The first goal aims at giving students a general understanding of Unix, assuming they are not familiar with the concepts to use and/or administrate such a system, due to the fact that education in other parts of the curriculum are heavily focused on Windows systems. With Unix, some concepts are very difficult, which are intended to be made clear. Throughout the lecture, exercises are given to show differences in platforms, using the (shrinking number of) hardware and operating systems' platforms available to students at the computer science and computing center's department of the University of Applied Sciences ("Fachhochschule") Regensburg. The main system that the lecture is based on Sun Microsystems' "Solaris" operating system, and other systems discussed throughout the class and exercises are "SuSE Linux" and "NetBSD".

The second goal is a focus for the contents of the lecture. "System Administration" itself is a wide field, and the goal of administrating a cluster of workstations is considered worthwhile as the various steps needed for this goal are difficult to learn e.g. in a self-teaching home-environment, while other topics like setup of mail, DHCP, DNS, Web and Samba servers may be easy to learn and practice.

For the further discussion, here is a list of the topics covered during the lecture again, as introduced in more detail in section 3.2. Short names are given for the further illustration:

Section	Title	Short name
0.	Introduction	-
1.	Historical Overview	-
2.	Login process, process correlation	-
3.	User commands (standalone and for shell programming)	UserCmds
4.	Information about the system	SysInfo
5.	Shell programming	ShellProg
6.	Application of shell scripts: booting and shutdown	Booting
7.	Networking	Network
8.	The X Window System	X
9.	Security	Security
10.	Practical Extraction and Report Language - perl	perl
11.	User management	UserMgmt
12.	Software management	SWMgmt
13.	Backups	Backup

When analyzing the structure of these topics, it becomes obvious that some topics need understanding of other topics as fundamentals. Figure 11 displays the goal of "Cluster Management" as main goal of the lecture, and illustrates the relations between the various topics discusse. These topics can be divided into three groups, shown in figure 12:

- User Management
- System Operations

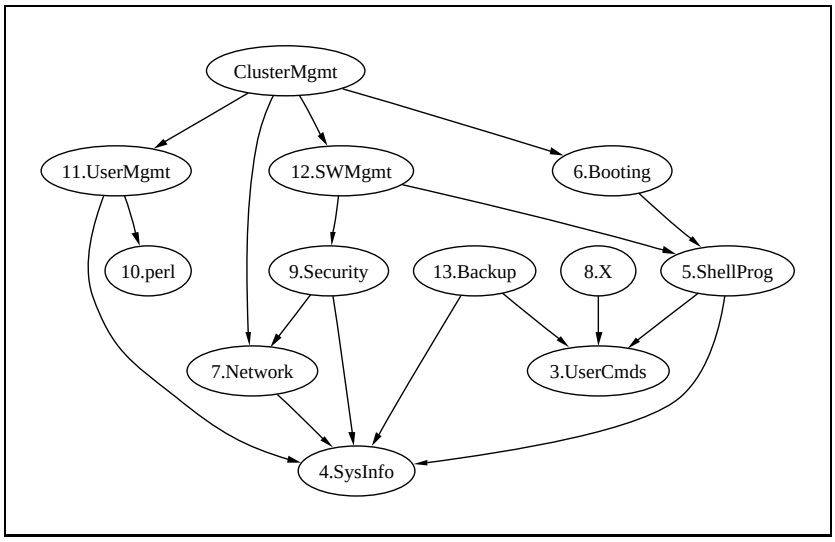

Figure 11: Structure of the “System Administration” lecture

- System Startup

The “User Management” group on the left of figure 12 assumes an understanding of the perl programming language, and it also needs an understanding of commands from “System Information, for the areas of user databases, and how to handle them. The “Systems Startup” group on the right needs shell programming to understand, which in turn uses a variety of user and system specific commands to determine information like which services to start. Finally, the middle group of “System Operations” Is more or less a collection of a number of topic that are loosely coupled, covering software management, security and networking, which again build up on the information derived from the system as well as various user commands.

Examining the discussion of the groups, it becomes obvious that each group can be divided into various levels, according to the difficulty or advancedness of the topic discussed, i.e.

- Basics
- Advanced
- High-level

Figure 13 illustrates this separation. The basics upon which all other topics rely are user commands and commands to determine information about the system, as well as understanding of networking concepts and configuration. The perl programming language listed as “advanced” here should be in the “basic” category

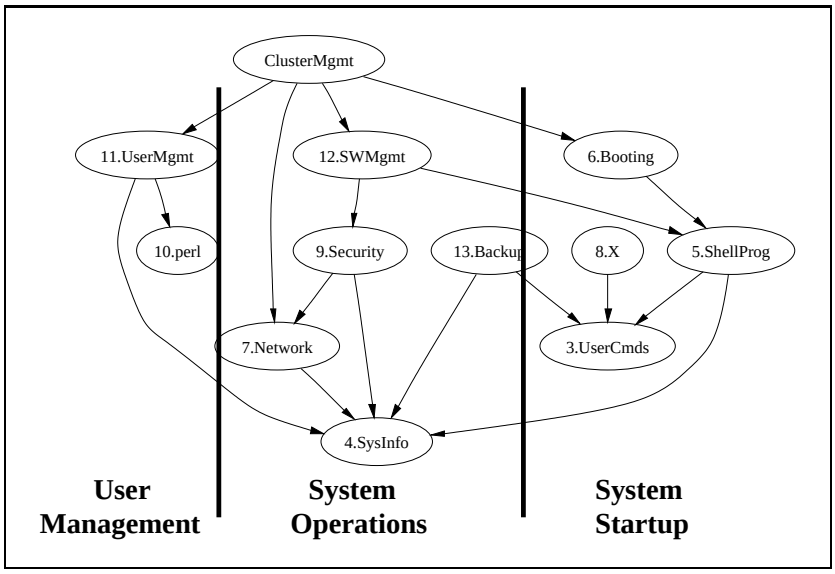

Figure 12: Thematic groups in the “System Administration” lecture

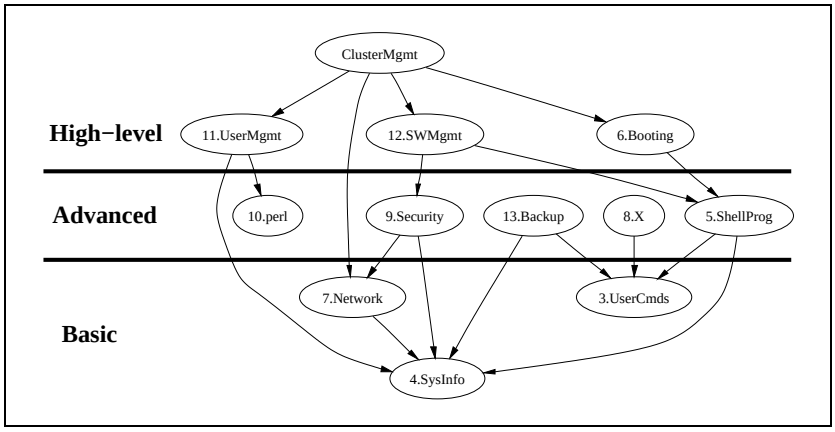

Figure 13: Levels of difficulty in the “System Administration” lecture

here too, the inaccuracy is due to the mechanical creation of the graph. Advanced topic building upon the basics are backups, the X window system, shell programming and security. Using all these basic and advanced topics, the high-level goals of user and software management as well as booting of the system (which is important for system configuration esp. in large scale environments) can be realized, to give the foundation needed for management of large clusters of Unix workstations and servers, the ultimate goal of the "System Administration" lecture.

As not all of the fundamentals of such a complex topic can be explained at once, the lectures have been split both horizontal into various levels of fundamentals building one upon another (see figure 13), and vertically to separate topical groups that can be separated to build logical units (see figure 12), with the result being a collection of single topics as presented in section 3.2.

3.4 Didactic Instruments

Besides the learning goals presented in the previous section, there are a number of didactic instruments that are used in various parts of the lecture and lab exercises that shall get presented in this section, grouped by what part of the course they are used in.

Lecture: The "System Administration" lecture consists of two class sessions with 90 minutes each, plus another 90 minutes for lab exercises. During class, laptop and video projector are used to present slides. Occasionally examples are developed on the blackboard to illustrate examples that are not immediately clear from the slides; the slides usually get updated to include these blackboard examples after lectures.

The teacher's notebook is used to display examples for commands, procedures and to show example outputs of various systems, either the local laptop or a remote system accessed via the ethernet connection available to the teacher in each classroom.

A number of books covering various topics covered in the lecture or related are passed around in the first lecture[1], but students are not required to read all or parts of them - all the lecture material is present in the lecture slides and online lecture notes.

The history of Unix is illustrated by a printout of a graph displaying all historical Unix releases so far, printed on many sheets of paper and glued together[2]

Online lecture notes: The online lecture notes are identical with the slide presented during the lecture, each student has online access to the lecture notes

[1] [Feyrer, 2007b] "Literatur"
[2] [Lévénez, 2007] .

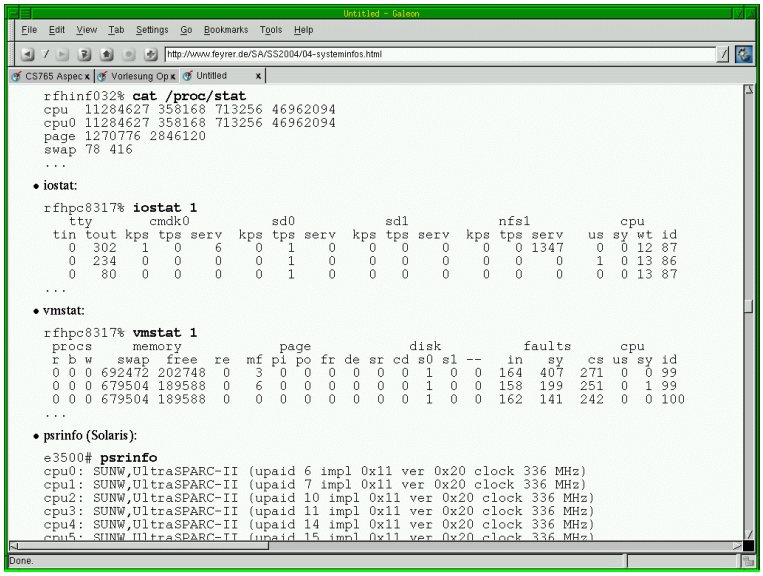

Figure 14: Examples help learning without a computer

so he can print them in advance, bring them to the lecture and make personal notes if needed. If a situation is found to be described suboptimal during class, the notes are updated after the lecture to clarify the situation

Lecture notes are available at http://www.feyrer.de/SA/ as a set of HTML files[1].

Examples: The lecture notes include examples for many situations that may arise in the topics described. Examples include a description of the situation, the exact command name to input and example output, so students don't have to sit in front of a computer to learn what a command does, but can do so from the online lecture notes' examples only. Examples are design to be as complete as live demonstrations for the purpose of learning with no machine at hands, which is esp. important as the lecture tries to give examples of many different machines and operating systems, many of which aren't widely available. Figure 14 shows an example found in the online lecture script.

Live Demonstrations: System administrative procedures that consist of a number of steps are demonstrated live. Demonstrations give detailed description of context in which the demonstration happens, including machine hardware and operating system that the procedure is happening in, a description of intention of the demonstration, with an outline of the conceptual steps first, followed by a description of commands and tools used, with reasoning why each one is used. Next step is an interpretation of output and

[1] [Feyrer, 2007b]

other effects resulting from demonstration steps as well as an analysis and description of the system after the demonstration, with retrospect on how each step affected the system, and in which way.

Analysis of existing systems: Strong emphasis is given in the whole "System Administration" course on the multi-platform property of Unix, which is supported by many examples and exercises that are intended to be ran on multiple different systems, to learn the properties of single systems as well as differences, so the students can infer concepts commonly found on many Unix systems as well as others that are only found on single systems, e.g. as discussed during the "Networking" or "System startup" sections.

For that purpose, a number of machines are available: Solaris/x86, Linux/i386 and NetBSD/i386. Possible systems for future demonstrations may include SGI machines running Irix and Sun machines running Solaris/sparc. Students have logins on all these systems, thanks to the NIS/NFS infrastructure of the computer science department.

Lab exercises for hands-on learning: Each section of the online lecture notes contains suggested exercises that students are expected to work on during lab exercises. The lab for these exercises consists of 15 PCs running a dual boot of Solaris/x86 and Windows 2000 (little to not used for the exercises), and four PCs running NetBSD. During lab exercises, machines are reserved for the students to do the exercises. No solutions to the exercises are published, to motivate students to come up with their own solutions. Personal experiences show that when handing out solutions for the mixture of exercises, students prefer to just read the solution (or learn it by heart), and don't go through all the steps to learn the aspects of the topics discussed.

Attended tutorials: During lab exercises, a tutor is present for asking questions about the working environment, machines, operating systems, their configuration, the exercise in question, and any related questions arising. Questions can be answered during lab exercise time; As students are free to do the exercises outside the lab exercise time, they can contact the teacher via email (and in some rare occasions via IRC chat) to ask questions. Using the lab exercise time for asking questions is encouraged to the students.

These instruments described here are currently in use in the "System Administration" course given at the University of Applied Sciences ("Fachhochschule") Regensburg. With some variety on machines and operating systems, the instruments are the same for the past few years that the lecture is given. A number of alternative instruments could be used in theory, some of which are discussed in the next section.

4 Analysis of the Current Situation

This section analyses the "System Administration" lecture discussed in section 3 under the light of the learning theories and didactic fundamentals given in section 2.

Taking the suggestions for ideal progression and tools for a lecture given in section 2.5 into account, a list of measures can be identified to improve student orientation and learning performance of students over the current form of the lecture:

- Define goals at the start of the semester: besides setting general, system/distribution-independent understanding of Unix as a goal, cluster management should be mentioned explicitly.
- Give a better overview of the way how the given goals are reached. This needs change in two places: First, give an overview over all chapters at the start of the semester, similar to the overview given in section 3.3. Second, at the start of each chapter, outline the contents that will be presented.

These changes can be realized easily in future incarnations of the "System Administration" lecture.

Another problem is more difficult to solve, though: In advanced topics, practical exercises are indispensable, which can be seen from the description of the current lecture in section 3.2 and the "wishlist" of alternative instruments given in section 5. Merrill supports this by stating that "much new scientific knowledge is dynamic in character and cannot be understood without a more active representation and student involvement"[1], and the approaches of situated learning and related cognitive concepts introduced in section 2.4 prove this. Hubwieser also asks for modelling and simulation to be part of the educational principle, and not part of the lecture contents[2].

While the course starts out with basic topics that can be easily learned without practical exercises from merely looking at examples and descriptions, more advanced sections need practical dealing for understanding, and on highest levels, system administrator privileges are required to do these practices. The gradual move from behaviouristic learning theories for basic topics to cognitivistic learning theories for advanced topics also shows that these two forms aren't the only ones needed to fulfill all the needed requirements, and that mixed forms like illustrated in section 2.4 are needed. Unfortunately, practical exercises with administrative privileges to access the system configuration level and related didactic

[1] [Merrill et al., 1991] p. 7
[2] [Hubwieser, 2000] p. 69

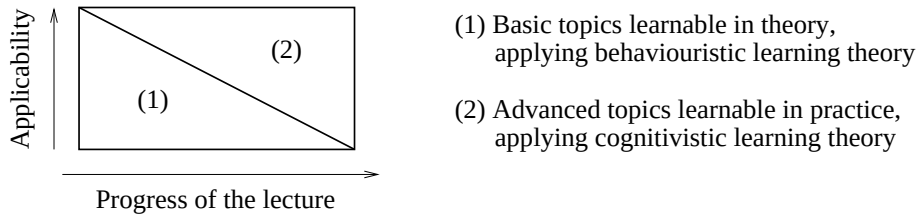

Figure 15: Change in learning paradigm with advancing level

instruments to let students master these exercises are not possible at the present time as described in sections 3.2, 3.4 and 5.

Figure 15 illustrates the correlation between basic and advanced topics: Basics introduced at the start (top) of the lecture can be learned without practical exercises, applying behaviouristic learning theory only. But the more advanced the lecture gets, both in time and in level of topics discussed, the more are practical exercises needed for understanding and learning, applying cognitivistic learning theories.

Basic topics can be learned through simple behaviouristic learning methods like drill-and-practice in theoretical manners, but with increasing level of difficult practical exercises following cognitivistic or constructivistic models are required, even if no system privileges are necessary at first. But for the "big" topics which build the goals of the "System Administration" lecture, it is not sufficient to cover them on a theoretical level. Practical exercises esp. with system privileges are mandatory for throughout understanding.

As a summary, there is a need for practical exercises applying cognitivistic learning methods in the "System Administration" lecture, esp. with system privileges on one side, and a lack of manpower and resources to setup and re-install machines on the other side, preventing this. A possible solution would be to build a virtual environment that allows practicing the real goals of the course where only theoretical coverage of these topics is possible so far.

The realization of this "Virtual Unix Lab" is more demanding, and will be covered in the remaining chapters of this document.

5 Future directions

This section intents to describe a number of didactic instruments that are not currently used in the existing course, but that may be useful for future improvements, to solve the one or other problem noted before.

- Solutions to (selected) exercises could be handed out, including a description of the solution. Providing well-written examples esp. for the programming parts like Bourne and perl programming may give students a better idea of how to use certain constructs.
- Access to more (different) machines with more operating systems would be helpful for better understanding of system attributes. Useful machines would be SGI machines running Irix, Sun SPARC- and AMD-based machines running Solaris as well as a set of PC machines running different Linux distributions like RedHat, Gentoo, Debian and Slackware (besides the SuSE already available) would be nice for analyzing setups even without root privileges. For example, the NIS and NFS client setup could be derived from such machines.

  Problem here is the costs of hardware and operating systems on one side, and maintenance of machines on the other side, which the computer science department cannot provide currently.
- At some points, contents could concentrate on using available GUI tools like SuSE's "yast", Solaris' "admintool" or the Solaris Management Console etc. instead of the basic building blocks provided by configuration files and command line tools. These tools could be used if it was ensured that students understood the underlying concepts properly, e.g. they would be more appropriate for using them in an "Advanced System Administration" class instead of a class teaching basics, like the current "System Administration" lecture intends to.
- It has been pointed out in section 3.2 that many topics cannot be practiced properly, due to lack of machines which can be accessed with system administrator privileges. While a number of workstations are available for all students – 15 running SuSE Linux, 15 running Solaris/x86, 4 running NetBSD – none of them is available for practicing system administrative tasks, as all of then are public machines that other students need to use too. Handing out system administrator privileges on these machines would require re-installation of the machine after the exercise, as the system's state would not be known, and this couldn't be trusted for public services.

  Examples of exercises where exclusive access to hardware would be useful include:

  – setup of various operating systems, and related configuration to get the systems going in a predefined environment
  – setup of various client/server scenarios, e.g. mail, POP, IMAP, spam filtering, DNS, DHCP, FTP, SSH, Samba, NFS, NIS and many others[1].
  – troubleshooting scenarios where systems are setup to misbehave in one way or other, students are expected to identify and solve problems.

  Due to the lack of manpower, this reinstallation cannot be performed after each lecture, and due to no dedicated machines for system administration

[1] [Ernst, 2004]

training, no practical exercises are currently offered for many areas that need these privileges, e.g. user and software management, basic and advanced network configuration, esp. for setting Network File System (NFS) and Network Information Service (NIS) client and server machines.

- Besides machines and operating systems, access to network hardware like cabling, hubs and switches as well as hardware for backup, such as tape drives of various technologies like AIT or DLT, and maybe some external disks and RAID arrays would be useful for practicing some of the basic setup and operations principles.

  Again, this is not possible or available at this time due to the desolate financial situation of the computer science department.

- For describing certain setup or troubleshooting situations, it would be useful to have machines available in exactly such a situation as described, which students then could pick up for further practicing, realizing an "Anchored Instruction" approach[1].

  Besides the lack of hardware resources, such setups would needs lots of preparation to define the systems to be in a specific state, and even more effort to backup and restore exactly that situation for later replay by all students. While this would be very useful for troubleshooting setups, it is again not possible due to lack of manpower, machines and money.

- Right now, the whole "System Administration" lecture is centered around classroom teaching where students are expected to being present. While students are free to take the lab exercises outside of the lab hours, it is recommended to take them when the teacher is present, to get optimal feedback on questions and problems.

  Moving the lecture into a "virtual" environment, where students decide on the process themselves would be possible due to the availability of the online lecture notes. Moving the lab exercises into a completely virtual environment with no teacher present any more would be more demanding: Interactive, tutoring and adaptive components would be needed to help students in situations where teachers can look over their shoulder today, and react to the situations they see[2].

This list of further instruments that could be used in the "System Administration" lecture is by no means complete, but it illustrates a number of approaches that could be used to improve education of system administartion.

[1] [Mandl et al., 1994] p. 171 and 173
[2] [Bruns and Gajewski, 2002] pp. 22

References

[Bruner, 1961] Bruner, J. S. (1961). The act of discovery. *Harvard Educational Review*, 31(1):21–32.

[Bruns and Gajewski, 2002] Bruns, B. and Gajewski, P. (2002). *Multimediales Lernen im Netz – Leitfaden für Entscheider und Planer*. Springer Verlag, Heidelberg, Germany.

[Clark, 2000] Clark, D. Developing Instruction or Instructional Design [online]. (2000) [cited 2007-10-05]. Available from: `http://www.nwlink.com/~donclark/hrd/learning/development.html`.

[Eikenbusch and Leuders, 2004] Eikenbusch, G. and Leuders, T., editors (2004). *Lehrer-Kursbuch Statistik*. Cornelsen Verlag Scriptor, Berlin, Germany.

[Ernst, 2004] Ernst, T. (2004). Mögliche Szenarien für das Virtuelle Unix Labor. Technical report, Fachhochschule Regensburg, Computer Science Department.

[Feyrer, 2001] Feyrer, H. TTYs and X Window: Unix Now and Then [online]. (2001) [cited 2007-10-05]. Available from: `http://www.onlamp.com/pub/a/onlamp/2001/03/22/tty.html`.

[Feyrer, 2005a] Feyrer, H. (2005a). Didaktik der Systemadministration. In *GUUG Frühjahrsfachgespräch 2005 Proceedings*, Munich, Germany. Available from: `http://vulab.fh-regensburg.de/~feyrer/vulab/hubertf/guug-sa-did.pdf` [cited 2007-10-05].

[Feyrer, 2005b] Feyrer, H. (2005b). Didaktik der Systemadministration. *UpTimes - Mitgliederzeitschrift der German Unix User Group (GUUG) e.V.*, February 2005.

[Feyrer, 2007a] Feyrer, H. Open Source [online]. (2007) [cited 2007-10-05]. Available from: `http://www.feyrer.de/OS/`.

[Feyrer, 2007b] Feyrer, H. Systemadministration unter Unix [online]. (2007) [cited 2007-10-05]. Available from: `http://www.feyrer.de/SA/`.

[Feyrer, 2008] Feyrer, H. (2008). System Administration Training in the Virtual Unix Lab. preprint.

[Frank, 1969] Frank, H. (1969). *Kybernetische Grundlagen des Lernens und Lehrens*. AGIS Verlag, Baden-Baden, Germany.

[Gagné, 1967] Gagné, R. (1967). *The conditions of learning*. Holt, Rinehart and Winston, New York, NY, USA.

[Gagné and Briggs, 1974] Gagné, R. M. and Briggs, L. J., editors (1974). *Principles of instructional design*. Holt, Rinehart and Winston, New York, NY, USA.

[Hamilton, 2007] Hamilton, B. Rosetta Stone for Unix [online]. (2007) [cited 2007-10-05]. Available from: `http://bhami.com/rosetta.html`.

[Hammer and Elby, 2000] Hammer, D. and Elby, A. (2000). Epistemological resources. In Fishman, B. J. and O'Connor-Divelbiss, S. F., editors, *International Conference of the Learning Sciences – Facing the Challenges of Complex Real-World Settings*, pages 4–5, University of Michigan, Ann Arbor, USA.

[Holland and Skinner, 1961] Holland, J. G. and Skinner, B. F. (1961). *The analysis of behaviour*. McGraw Hill, New York, NY, USA.

[Hubwieser, 2000] Hubwieser, P. (2000). *Didaktik der Informatik*. Springer Verlag, Heidelberg, Germany.

[Kerres, 1998] Kerres, M., editor (1998). *Multimediale und telemediale Lernumgebungen: Konzeption und Entwicklung*. Oldenbourg Verlag, München, Germany.

[Kuyper, 1998] Kuyper, M. (1998). *Knowledge Engineering for Usability*. PhD thesis, University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, Netherlands.

[Lave and Wenger, 1991] Lave, J. and Wenger, E. (1991). *Situated learning*. Cambridge University Press, Cambridge, MA, USA.

[Lévénez, 2007] Lévénez, E. Unix History [online]. (2007) [cited 2007-10-05]. Available from: `http://www.levenez.com/unix/`.

[Mandl et al., 1994] Mandl, H., Gruber, H., and Renkl, A. (1994). Situiertes Lernen in multimedialen Lernumgebungen. In Issing, L. J. and Klimsa, P., editors, *Information und Lernen mit Multimedia*, pages 167–178. Psychologie Verlags Union, Weinheim, Germany.

[Merrill et al., 1991] Merrill, D., Li, Z., and Jones, M. (1991). Second Generation Instructional Design (ID_2). *Educational Technology*, 30(1):7–11. Available from: `http://id2.usu.edu/Papers/ID1&ID2.PDF` [cited 2007-10-05].

[Nösekabel, NA] Nösekabel, H. (N/A). Mobile Education. Doctoral thesis draft as of 2004-01-09, unpublished.

[Papert, 1982] Papert, S. (1982). *Mindstorms: Kinder, Computer und neues Lernen*. Birkhäuser Verlag, Basel, Switzerland.

[Pawlow, 1972] Pawlow, I. P., editor (1972). *Die bedingten Reflexe*. Kindler Verlag, München, Germany.

[Piaget, 1967] Piaget, J. (1967). *Six psychological studies*. Random House, New York, NY, USA.

[Pressey, 1926] Pressey, S. L. (1926). A simple apparatus which gives tests and scores - and teaches. *School and Society*, 23(586):373–376.

[Pressey, 1927] Pressey, S. L. (1927). A machine for automatic teaching of drill material. *School and Society*, 25(645):549–552.

[Reigeluth, 1983] Reigeluth, C., editor (1983). *Instructional-Design Theories and Models*. Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, USA.

[Richey, 1986] Richey, R., editor (1986). *The theoretical and conceptual bases of instructinal design*. Kogan Page, London, UK.

[Schulmeister, 1997] Schulmeister, R. (1997). *Grundlagen hypermedialer Lernsysteme*. Oldenbourg Verlag, München, Germany, 2nd edition.

[Schulmeister, 2002a] Schulmeister, R. (2002a). *Grundlagen hypermedialer Lernsysteme*. Oldenbourg Verlag, München, Germany, 3rd edition.

[Schulmeister, 2002b] Schulmeister, R. (2002b). *Lernplattformen für das virtuelle Lernen*. Oldenbourg Verlag, München, Germany.

[Schuman, 2007] Schuman, L. Perspectives on Instruction: What are the problems and strengths of these theories? [online]. (2007) [cited 2007-10-05]. Available from: `http://edweb.sdsu.edu/courses/edtec540/Perspectives/Perspectives.html`.

[Seidel and Lipsmeier, 1989] Seidel, C. and Lipsmeier, A. (1989). *Computerunterstütztes Lernen – Entwicklungen, Möglichkeiten, Perspektiven*. Verlag für Angewandte Psychologie, Stuttgart, Germany.

[Shannon and Weaver, 1949] Shannon, C. E. and Weaver, W. (1949). *The mathematical theory of communication*. University of Illinois Press, Urbana, IL, USA.

[Skinner, 1947] Skinner, B. F. (1947). 'Superstition' in the pigeon. *Journal of Experimental Psychology*, 38:168–172. Available from: `http://psychclassics.yorku.ca/Skinner/Pigeon/` [cited 2007-10-05].

[Skinner, 1968] Skinner, B. F. (1968). *The technology of teaching*. B. F. Skinner Foundation, Cambridge, MA, USA.

[The Open Group, 2004] The Open Group (2004). *Single Unix Specification*. American National Standards Institute, 1430 Broadway, New York, NY 10018, USA. IEEE Std 1003.1, 2004 Edition. Available from: `http://www.opengroup.org/onlinepubs/007904975/toc.htm` [cited 2007-10-05].

[Thorndike, 1911] Thorndike, E. L. (1911). *Animal Intelligence*. MacMillan Publishing, New York, NY, USA. Available from: `http://psychclassics.yorku.ca/Thorndike/Animal/` [cited 2007-10-05].

[Tulodziecki, 2000] Tulodziecki, G. (2000). Computerunterstütztes Lernen aus mediendidaktischer Sicht. In Kammerl, R., editor, *Computerunterstütztes Lernen*, pages 53–72. Oldenbourg Verlag, München, Germany.

[Virtuelle Hochschule Bayern, 2001] Virtuelle Hochschule Bayern (2001). Verordnung über die Virtuelle Hochschule Bayern. *Hochschulrecht in Bayern*, 1180.

[Watson, 1913] Watson, J. B. (1913). Psychology as the Behaviourist Views it. *Psychological Review*, 20:158–177. Available from: `http://psychclassics.yorku.ca/Watson/views.htm` [cited 2007-10-05].

[Weidenmann, 1993] Weidenmann, B. (1993). *Pädagogische Psychologie*. Psychologie Verlags Union, Weinheim, Germany.

[Wiener, 1948] Wiener, N. (1948). *Cybernetics, or control and communication in the animal and machine*. MIT Press, Cambridge, MA, USA.

[Wiggins, 1989] Wiggins, G. (1989). A True Test: Toward More Authentic and Equitable Assessment . *Delta Phi Kappan*, 70(9):7–11.