Developing analytical competency through informatics teaching

Walter Oberschelp

*Lehrstuhl für Angewandte Mathematik (RWTH),
Aachen, Germany*

ABSTRACT

A short summary is given of informatics teaching in Germany. As goals for informatics education, some competences are listed which can only be achieved through informatics. This leads to a justification for obligatory informatics courses in secondary instruction in Germany. Such teaching must be supported by tools, which have certain fundamental requirements: zooming in time and space, algorithmic transparency and a simplicity of interfaces. Graphical facilities using the discrete pixel plane become necessary components of software, which is available now or shortly. One major concern is to prevent the degeneration of algorithmic capabilities as a possible consequence of software that is too easy to use.

Keywords: tools, algorithms, graphics, inter-active, instruction

INTRODUCTION

As an introduction, it is important to provide a short resumé of the experiences of teaching informatics in German high schools. This is followed by comments and personal opinions on recommendations for the curriculum [1]. The author has been one of the members in the committee preparing these recommendations, based on the so called Dagstuhl-Recommendations [2].

It is my opinion that the experiences in Germany with basic courses on computer literacy for all pupils, called ITG (Informationstechnische Grundbildung) have not been convincing. There was neither a general consensus on the curriculum nor did we get any realization line for a broadly based teaching system for ITG. The attempt to make mathematics teachers responsible for this kind of instruction met with resistance from all groups, and there was no general agreement on fitting an ITG-block into the curriculum (for conflicting positions see [4] and [5]). But, the impossibility of achieving ITG is fortunately connected to the fact that ITG seems not to be really necessary: the mere skills of handling a computer must not be the object of school instruction at all, any more than the skills in car driving, typewriting or handling another complicated technical tool. The trend that makes computer interfaces more easily understandable, as a consequence of the marketing needs of industry, combines with the experience that our children do not fear the computer. Other aspects of ITG, in particular the reflection on social implications, can be covered by conventional educational components, since the substance is too thin for special courses with only a small and elementary background.

On the other hand, my proposition is that there should be compulsory courses in informatics at least in each curriculum leading to college maturity. This position is not generally accepted (see [6] and [7]). The development of high school informatics in Germany over recent years shows a somewhat chaotic situation as a result of different cultural policies in our federal system. In general, a big boom has occurred, with many teachers - mostly mathematicians - having undergone additional and voluntary training in informatics. Thus, in many schools a high quality instruction in computer programming and programming principles is available, but normally on a free choice basis. Virtually nowhere is informatics compulsory. A good description of the situation and some references are given in [1]. More details can be taken from [8] and [9], while recommendations for the teachers' education are given in [10].

There appears to be growing competition with other fields, mainly with mathematics, not only about curriculum proportions, but also about the content of instruction. Mathematics seems to become more and more aware of the fact that lost fields have to be regained - in particular the whole world of algorithms. There are also new challenges. The advent of formula manipulation systems into mathematics instruction will cause a revolution, as the invasion of high-level software tools into informatics seems to make many "classical" applications obsolete.

GENERAL PRINCIPLES OF INFORMATICS INSTRUCTION

It would seem that informatics instruction has to change towards the future with more maturity. At the present state of technology it is possible to make recommendations which are rather independent from the actual availability of those tools. It is important to keep in mind and to insist on some educational principles which might be unpopular and uncomfortable from the aspect of those advanced tools.

The justification of this opinion is based on the fact that there are special competences for our information based society which cannot be developed in other fields of instruction. In general, in order to master the new opportunities of information processing, it is necessary to know how to use these opportunities and to understand the fundamental processes which are going on in complicated systems, on both the elementary and more complex levels. Equipped with those competences, pupils will be able to use, analyze and evaluate information and to create their own solutions for special problems later, thus preparing for autonomous positions in our complex world. The competences, which need to be developed, are listed as follows.

- recognize and analyze problems, which are solvable appropriately by a computer, (observation competence), in particular
 - a) recognize language (natural or artificial text) - as the fundamental vehicle of communication and of interactive structured problem solving
 - b) use visualizations and pictures and their inherent patterns as an aid for perceiving structures and for recognizing correlations between facts;
- deliver precise and appropriate (re)actions on a syntactically and semantically sound basis in interaction either with linguistic or graphical representations of a software tool or with partners of a working group (serving competence);
- apply those abilities to various fields and design sequences of actions either using a software tool in order to solve a problem (programming competence) or working autonomously or as part of a group (combinational competence);
- recognize the role of basic low-level-activities in simple and complicated settings and practice those activities with or without the support of a software environment (algorithmic competence);
- find quantitative estimates for complex processes, distinguish essential parts and judge the role of communication and interfaces between those components (evaluation competence);

- combine, with the help of appropriate tools, components, which have been, or can be, fully evaluated in a systematic and incremental way to a completed structure (assembling competence);
- verify the functionality of a construction by efficient checking procedures (test competence);
- improve solutions by appropriately adapting the requirements to each other (optimizing competence);
- distribute tasks and find parallel solutions bearing in mind communication needs between processes (parallel design competence);
- estimate human conditions and social contexts in the classroom and in a creative working group and develop responsibility (personal competence).

These competencies should form the basis of informatics instruction.

SOFTWARE TOOLS - REQUIREMENTS AND PROPERTIES

These general goals of informatics instruction affect both the tools and the subjects of our instruction. The software and hardware tools of instruction must be selected and developed according to their value as educational devices. The usual evaluation of software favours technological perfection for direct and fast solutions. This has also been the case in German institutions, which evaluate software for instruction purposes, as I know from personal communication with K. Keidel (Augsburg) of the Bavarian test institute. Of course, we want to follow technological progress; but usually advanced commercial software is not suitable for instruction, since transparency and motivation are lacking. We should encourage the use of those tools, which show intermediate steps, support imagination and analytical endeavours and which are equipped with flexible interfaces. In contrast to the fixed special purpose interface for the private and commercial user, students should be able to communicate with the system or with their teams through a flexible or at most semi-automatic interface. We know from experience that working with such interfaces challenges improvisation and needs robustness and intensive dedication. Therefore, in order to prevent an isolation of students, such that the computer is their only communication partner, the teacher often schedules an instruction unit that co-ordinates the work in advance by informal agreements, as support through the system itself cannot and should not be relied upon.

The big achievement of the computer is its ability to transform any action of a user directly into a reaction. Thus users have direct evidence that their actions cause an effect. The most direct immediate contact with

the medium is the interactive screen in handling menus, texts and graphics. Working with the editor in an interactive programming environment, for example, a text system editor, requires lexical and syntactic correctness. Thus the editor is a teacher by itself. The supervision function of the system insists on absolute exactness, it gives hints to localize errors and forces a search for the error with more or less support. The position of the cursor supports precision in sequential thinking, it makes conscious the fundamental state-space principle, that is the system is at any time in an actual state (within a partially hidden context) and the cursor reflects the activity of the user, who works with comfortable input devices.

It is obvious that students also have to know the semantic meaning of an action. They have to master a minimal repertoire of meanings for procedures on data structures, such as sort, search, retrieve, select, cancel, compare, count, map, merge, collect, compile, concatenate, and transitive closure. This and the knowledge of the fundamental logical and arithmetic functions are an indispensable subject of instruction and must be covered by any curriculum.

ALGORITHMIC SKILL AND TIME AND MOTION CONSUMPTION

In this context it is important to evaluate critically formula manipulation systems. Experiences in Germany and Austria [11] have shown, that a considerable acceleration in mastering the mathematics curriculum can be achieved and that in particular the teacher, who is designing classroom tests, can take advantage of such systems. But on the other hand the need to formulate exactly the desired goal sets a clear limit for applications without precise ideas, and we have formulated (cf. [12]) the danger of degeneration for the permanent user of a formula manipulation system. Thus, in the same way that the lack of use of muscles can lead to physical degeneration, the use of perfect automatic tools may lead to mental atrophy and to an algorithmic disabling. I will return to this point later in the paper. There is no clear position among teaching authorities in Germany and as yet a lack of consciousness in the public discussion. But this is not unexpected; after all the discussion about value and limitations of the pocket calculator in instruction has not yet reached a conclusion. While mathematical rigour and logical precision is the hard core of mathematic instruction and competes with numerical calculations, the ability to perform algorithmic manipulations and thus to experience the

resulting cause-effect-relation is an indispensable informatics quality, which cannot be substituted by even perfect software tools.

It is well known that the attraction of using such an advanced tool without any exertion, like a recreational toy, is one of the main motivations for our youth to be occupied with the computer. Students are not to be blamed, therefore, but they have to make some progress and move from playing with the tool to challenging it, in order to look behind the apparent view of the action. Thus, for example, the question "what is going on algorithmically if a little man is moving over the screen, if it is suddenly disappearing or created *ex nihilo*?" should be discussed as well as the question how to manage the creation of straight lines, circles and nice curves on the screen.

In my opinion is a crucial point; it is important to develop strict ideas about the conditions for "implementing" motion on the screen. It should become clear to the student that within 50 msec - this is the critical time to get the impression of a continuous motion - a 20 MIPS-computer can perform one million instructions. It is easy to build a simple figure on the screen, it is possible to change the direction of a drawn line within that time, if we know what to do in advance or if the system has to react to a mouse click. But all this is impossible if the action is triggered by the result of a combinatorial calculation like the inspection of all possible orders of ten objects or on a logical condition, where twenty binary variables are involved. Most students in Germany have no experience in discrete optimization and thus are too innocent to take into account the influence of the combinatorial explosion on computer performance. But I feel that for a software tool the transparency of the performance time is a most important quality. All our tools should be equipped with a device to achieve time-zooming in order to expose discrete motions, to use a step by step mode and to control a speed-down and, if possible, a speed-up by continuous parameter changes. In many cases this is a rather harmless extra feature of the software, but it is indispensable for learning purposes.

There are important examples to illustrate the time problem in informatics. One classical, well elaborated project [13] is the one-way (trapdoor) calculation problem, that is the totally different time requirements to multiply two big numbers and to factorize a given number. We use this facility for achieving data security. It is similarly possible to study the tricks of a logarithmic speed up by divide-and-conquer techniques in order to improve sorting, searching and number crunching.

CONSTRUCTIVE CREATIVITY FROM ALGORITHMIC CAPABILITY AND PERCEPTUAL LEARNING

In general, the amount of material which is important for informatics instruction is tremendous. But the most important issue is the way in which the materials enable learning. The learning process to create, design and implement a complex solution by the structuring, incrementally supplementing, prototyping, implementation, testing and documentation can happen in different environments. The students can work in a cooperative situation with a real world team, or in a network of processors or with an interactive software tool. In the latter case the reliability of the components is within the scope of only one's own responsibility; in other cases the student has to estimate the reliability of the components or of the companions. Both situations are highly educative and should be practiced in various situations.

What is the basic ability a student needs in order to achieve all this? Students need to develop the ability to use informatics as an appropriate access to create artificial worlds and to simulate real-world-actions on the computer. But teaching informatics cannot only mean to teach the use of those complex tools and to work in self-created maybe highly sophisticated artificial worlds. It is also important to guard against the danger of loosing reality and becoming a somewhat autistic hacker without contact with the real life.

On the other hand, it is also important to fight against the degeneration and shrinking of vital and necessary abilities by insisting on training the natural gifts not only of our body but also of our mind. In the same way, as we preserve physical fitness by training and by forcing the student to produce sweat, we should have a concern to produce "mental sweat" by occasionally and on purpose stopping the computer and running things on our own brain. This might be the most unpopular part of our education business, which is as hard to motivate as making a man run with his own feet when he usually moves by car or plane. In the same way as we have to preserve literacy by not only relying on pictures or to maintain numeracy by sometimes avoiding the calculator, we have to preserve logicacy and algorithmacy by continuously training simple mental activities without artificial tools. The development of realistic training programs and a good coordination to the other parts of informatics instruction is one of the most important curriculum concerns at all. I have the impression that these needs are not yet recognized in our school system.

To return to the discussion of materials and methods, one important chance for informatics instruction is the use and development of simulations; simulating a system in particular with stochastics-support is the most direct application of the immediate cause-effect-visibility paradigm, and simulation is a fundamental technique for system development. Thus, for instance, modelling the ecological balance of a lake by means of a predator-prey simulation can make transparent the behaviour of linear systems, of non linear feed-back-controlled systems, under the logistic equation, and even of chaotic systems [13]. Here, the crucial topics are linear systems in their temporal development.

I emphasize again that the importance of real time can hardly be over-estimated in informatics instruction. The puristic concepts of top-down or bottom-up techniques turn out to be no longer necessary under modern software tools, since incremental working very often helps to avoid several passes through the system, which are usually necessary in systematic system design. But in this context my former remarks concerning degeneration through luxurious tools also apply. The minimal requirement for using a database-tool, for example, is to make the structure of such a system visible, to show access-paths, reveal with painstaking exactness the reasons for concurrency conflicts. It is important to visualize the actions, which happen during the execution of a select-from-where- statement such as merging, selection, tests, decision trees. The same is true for learning to understand the actions which are going on in computer hardware at the binary (logical) level [14]. There exist already excellent visualization tools which illustrate these processes and which can be applied, when the basic low-level facts have been mastered. I know of several such solutions, which have been developed by teachers with good results in classroom instruction. There is much discussion on this visualization point. The system Cabri-géomètre [15] and [13], is a useful contribution here, which is used in several places in Germany in order to make geometric notions more dynamic and thus more evident. The already historic turtle-geometry technique also reminds us that motion in a graphical setting is one of the basic fundamentals of perceptual learning.

THE PARADIGMS OF DISCRETENESS AND OF PARALLELISM

Why do I concentrate the discussion to these temporal aspects? It should be clear that for most people the main source for cognition is visual perception on the screen. I feel that students sometimes are deceived by

its misleading high-resolution. It is important to emphasize that the screen is a discrete and pixel-oriented device. In many cases this should be demonstrated by making the raster visible through zooming. Informatics with its discrete philosophy has to fill the gap which arises from the usual neglect of discrete facts in mathematics instruction. Unfortunately, discrete pixel geometry is more complicated and not as nice as Euclidean Geometry compared with reality. There are also many algorithmic needs in computational geometry under Euclidean assumptions - thus computer algorithms revitalize a lot of classical long-forgotten mathematics. Being also a mathematics teacher, I want to formulate here my belief, which I know annoys many computer scientists, that informatics has the closest and most natural connections to mathematics and that we saw off our own position-branch if we cut off the connections to mathematics.

Some important spatial notions are better understood in the discrete manner. Most important seem to be the concepts of locality versus globality: the use of templates, that is, the processing of a pixel P by a function which depends on the neighbourhood of P , is the most effective algorithmic technique of graphics and of image processing, which can eventually also yield global informations about a picture P , like size, connectedness and genus [16]. The most attractive feature of those local operators in graphics is their use in the parallel mode - the inertia of the eye and the speed of the computer makes it simulate actions as parallel as if they are done by one processor sequentially. Thus thinking and designing in parallel is supported by visual perception on the two-dimensional screen, and this ability also works as the basis for pattern recognition with important applications in medical diagnostics. I am sure that in a very short time the equipment for informatics courses will contain the facilities of a graphic station.

Thus the discussion finally moves to parallel competence. School instruction has the chance to create and to further develop, for example, the natural ability to comprehend the working of a pipeline or more general of a systolic array. It is not easy to design and to verify such a fundamental device of technique. From a theoretical point of view it seems attractive also to deal with fundamental protocols of parallelisms which handle, for example, synchronization, distributed access, mutual exclusion or deadlocks. All these phenomena appear in our society from planning of production to the simulation of social processes. It seems to me, however, that we must restrict ourselves usually to study so called SIMD-processes (single instruction on multiple data) like mesh-connected or hypercube-calculations or multi-connective nets like perfect-shuffle, Butterfly-, Benes- or Clos- structures [14]. In my opinion there is no other

field of education than informatics, where all these fundamental materials can be learned.

HOW CAN WE ACHIEVE ALL THIS?

Finally it is necessary for us to consider the position of informatics in the general process of education (in German we use the non-translatable word *Bildung*). I have attempted to identify genuine and irreplaceable qualities of this instruction, as well as examples where informatics touches other fields. On the border with the linguistic sciences we have the concept of syntactical structure and the need to formulate ideas with appropriate semantical notions. We touch the historic sciences in analyzing and creating temporal processes considering contextual state conditions and taking into account necessary, expected and possible actions. The frontier with mathematics is the competition on algorithms and in modelling, with physics the analysis and handling of strict quantitative laws, with biology the technique to represent knowledge and to work with evolutionary concepts. The future of informatics teaching at school will require a recognition of these specific features of our field and to transfer them into the education process by designing a reasonable curriculum.

The way to achieve this competence in a one year course can hardly be sketched here. Starting from a medium position, that is neither top nor bottom, which is close to the actual situation of the pupil and using commercial software cautiously, some examples or paradigmatic small theories can be pursued incrementally in different directions. This can lead to a partly tree-like, partly meshed experience with concrete and important material. The examples would have to be selected with great care so that a representative block of know-how, which is valuable for our educational purposes, is accomplished. Some detailed descriptions are in our paper [3], but more of the advanced tools which we have mentioned must also be included.

The choice of material should be decided through national or even international discussions. The day that interested teachers create their own material for their purposes should be over, while still allowing some room for the creativity of the teacher. I think that such a common basis of knowledge, selected by admittedly somewhat arbitrary arrangements, might nevertheless be an appropriate basis for the development of informatics abilities and for the acceptance of informatics in the general curriculum.

REFERENCES

1. Fäkulttentag Informatik (1993) *Empfehlungen zum Schulfach Informatik (Sek. II) und zur Ausbildung von Informatik-Lehrkräften*. (Informal paper).
2. Claus, V. et al (1992) *Dagstuhler Empfehlungen zur Aufnahme des Faches Informatik in den Pflichtbereich der Sekundarstufe II*, Dagstuhl. (Informal paper).
3. Oberschelp, W. (1987) *Algorithmen und Computer im Unterricht (Projekte)*. Fernkurs an der Fernuniversität Hagen (Fachbereich Mathematik und Informatik).
4. Baumann, R. (1990) *Didaktik der Informatik*. Klett, Stuttgart
5. Penon, J., Sack, L. & Witten, H. (1992) *Informationstechnik und Allgemeinbildung*. LOG IN 12, 22-28
6. MNU (1993) *Positionen der MNU zum Unterricht in Mathematik, in den Naturwissenschaften und in Informatik*, MNU 46 , Heft 8
7. DMV (1983) *Stellungnahme zum Informatikunterricht an Gymnasien*. Dt. Math. Vereinigung, DMV, MNU 36, 304 ff
8. Baumann, R. & Koerber, B. (1991) *Informatik in der Schule der 90er Jahre*. LOG IN 11, 31-35
9. Peschke, R. (1990) *Grundideen der Informatikunterrichtes*. LOG IN 10, 25-33
10. (1987) *Gesellschaft für Informatik GI, Empfehlungen zur Lehrerbildung im Bereich der Informatik*. Printed in LOG IN 7, Heft 5/6.
11. Weigand, H.G., & Weth, T. (1991) *Das Lösen von Abituraufgaben mit Hilfe von Derive*. MNU 44, 177 ff
12. Oberschelp, W. (1988) *Rechenverfahren und Formelalgorithmen als Unterrichtsgegenstände*, in: *Arbeiten aus dem IDM der Universität Bielefeld*, Occasional Paper 116, Koll, W. & Steiner, H.G., ed. Winkelmann, B.
13. DIFF (1988) *Computer im Mathematikunterricht*, Dt. Institut für Fernstudien DIFF Tübingen, Beltz, Weinheim Steiner-Colloquium, Winkelmann, B (ed).
14. Oberschelp, W. & Vossen, G. (1994) *Rechneraufbau und Rechnerstrukturen*, Edition 6, München (Oldenbourg)

15. Schumann, H. (1990) *Neue Möglichkeiten des Geometrielernens durch interaktives Konstruieren in der Planimetrie*, MNU **43**, 230 ff
16. Oberschelp, W. (with Dohmen, M.) (1991) *Mathematische Methoden für Bildverarbeitung und Computergraphik, Schriften zur Informatik und angewandten Mathematik*, RWTH Aachen, Nr. 149

Walter Oberschelp was born in Herford, Germany, studying at Göttingen, Tübingen and Münster, gaining his PhD in Mathematical Logic in 1958. After holding professorships at Münster, Hannover and Illinois, he was appointed to his present post as Professor of Applied Mathematics and Informatics in Aachen where he has developed the Informatics Department and engaged in scientific work. He has delivered courses on informatics, discrete structures and algorithms.

Walter is also an advisor for many student teachers in mathematics and informatics, giving courses in high school instruction for both students and teachers.