6

Teaching informatics to nonprofessionals: why, what and how?

Charles Duchâteau
CeFIS, Facultés Universitaires Notre-Dame de la Paix
Rue de Bruxelles, 61, B-5000 Namur, Belgium
e-mail: charles.duchateau@fundp.ac.be

Abstract

Informatics is now fifty years old. Initially, only professionals used computers and it was quite easy to determine what the profile of these professionals was and, by consequence, to train them. Ten years ago a new reality appeared: nonprofessional users and thousands of software tools. As a consequence computer use is now appearing as a new field with its own mental representations, procedures and concepts. Several terms have been suggested to name this new domain: 'informatics technology', 'information technology', etc. There are links between this new field and informatics, but there are also many differences. In short: what must a user know about informatics systems, what concepts and principles make up the new discipline of 'informatics for users'?

Keywords

Informatics, university education, noninformatics majors, curriculum (core), role of CIT

1 INTRODUCTION

Let us first try to answer the question: 'why teach informatics to students in universities, who will be only users?'. Maybe we do not have to teach anything. Maybe, using a computer is only a matter of basic skills. In this paper the reasons for teaching 'something about informatics for users' will be compared with the traditional reasons put forward for teaching mathematics or physics to students whose primary interests are for example biology or pharmacy. For this comparison

Informatics in Higher Education F. Mulder & T. van Weerts (Eds.)

we in any case have to find and highlight the basic principles and concepts of 'informatics for users'. What are the unchanging features which characterize the vast field of computers use? What is the basic knowledge required for efficient and creative uses of different software tools? What are the real mental models allowing users to adapt to new tools and environments?

The second section of the paper will try to pick up some of these fundamentals. For example:

- data processing by a computer is always a formal process (in the word 'informatics' we should read 'form' and not 'information');
- computers do not process information, but work with a representation in numeric code of this information;
- the only thing the user is faced with is the pair: computer plus software.

The third section will present some ideas for teaching these basic principles.

- Basic principles from which users can infer and explain the behaviour of 'informatics systems' which very often are expressed by short sentences or slogans like 'the computer does not exist', 'do not speak about "information", only about "form"', 'to program is to have it done by...', 'informatics is not computer science'.
- Simple questions like 'from where do in fact the programs come which make my computer work?', 'what does the computer send to the printer?', and 'under what conditions can information be processed by a computer?'.
- Main threads which allow us to present various features in a coherent and consistent way (like the definition of a computer).
- Deductive reasoning which leads to important conclusions.

In conclusion the paper will show that these issues are brand new and that, as teachers, we will gain by an exchange of our experiences, trials and questions.

2 A TEACHER HAS TO TEACH!

One may ask: 'Why all these questions?'. Because a teacher has to teach! Whatever the answers to the questions mentioned above, whatever my opinion about the necessity of teaching the basics of computer science in some university study, I still have to teach every week first year students in biology, geography or pharmacy. My task is 'to talk about computer science without including any programming practice'. As a consequence in my teaching I have had to answer all these questions concerning teaching of 'informatics for end-users' to people who would not become computer scientists. At the moment I am in charge of a thirty hour course (two hours a week, for three months) complemented with fifteen hours of laboratory work supervised by a colleague. The students are in their second year of biology, geography or geology. I also teach a fifteen hour course (without laboratory work) to first year students in pharmacy. Both groups consist of more

than one hundred students. The conditions are typical of first year classes: I am not able to control the environment and the schedule in which the students will be brought into contact with computer science or at least in contact with computerized environments. I can only teach in a very classical way, with students sitting in a lecture hall for 55 minutes of informatics, after 55 minutes of physics and before 55 minutes of botany. A teaching assistant is in charge of exercise sessions: groups of twenty students spend two hours in a computer room (two students per computer) and discover the basic skills of word processing and spreadsheets.

3 ARGUMENTS FOR INCLUDING 'INFORMATICS' IN FIRST YEAR

Only fifteen years ago informatics was still excluded from the first year university curriculum, but today the situation is quite different. Above and beyond local circumstances and priorities reasons have been and are still put forward by academic authorities to include informatics in the curriculum. These reasons have changed with time. Not all of them have the same importance and not always the same persons have put them forward.

The 'electron microscope' argument

Firstly, informatics courses are motivated by computer use. This reason is similar to requiring physics classes for learning how to use a microscope efficiently. The argument goes: users must know the laws and principles underlying the functioning of an electron microscope in order to be able to understand how it works. As a matter of fact, physics programs generally contain an 'alibi chapter': 'Principles of the functioning of the electron microscope'. This argument is important and is worth some discussion. It is embarrassing when we have to talk - just like with microscopes - about 'the principles underlying the functioning of a computer'. According to the level of description principles of this functioning come under physics, electronics, algebra, algorithmics, etc. Or under what we find in most of the books about internal architecture (processor, memory, etc.) or global architecture (central processing unit, mass memory, etc.). These material descriptions are insufficient for the users to understand how their favourite word processing software works. As a consequence they lack coherent and efficient understanding.

A description which only deals with the computer itself cannot be satisfactory because the computer does not exist on its own (see also below). What the user is faced with is a duo 'computer plus software' where software takes very different forms: The computerized system used by a musician is completely different from the one used by an architect or a secretary.

However, the electron microscope argument is a very important one: it offers us the opportunity to state general principles which explain and underlie any use of

an informatics system (computer plus software). If we are not able to discuss these principles, then 'informatics for users' could be a myth.

The 'it is essential for future classes' argument

This is also a traditional argument to justify first year programs. They have to study mathematics because of the physics classes et physics in order to understand some parts of the chemistry classes, and chemistry because they will have biochemistry the following year.

The reason justifying my informatics class to biology students is that they will have a class about a statistical package after the first year. The mathematics class might well be more elaborate than needed to understand the physics class, and the physics class more elaborate than needed to understand the chemistry class.

The 'it is necessary in professional life' argument

Tomorrow, my first year students in pharmacy will have to work with computers in their dispensaries, so we have to train them in the use of computers and learn them to deal with hardware and software providers. However, computer scientists using the computing systems of today have not received the corresponding training during their studies. And the long period of time till the training can be used in the professional life speaks for a training stressing principles and unchanging features, focusing on analysis rather than on description.

The 'today it is part of basic academic culture' argument

This is a common argument for university first year. Computer science now has its own place in the university; other departments recognize this new field and give it some room in their curriculum.

The 'it is formative' argument

This argument is similar to the one which ascribes a formative character to the learning of mathematics or physics. It is often used in a defensive way: 'Even if it does not sound useful to you, it is formative'. This argument has been used in the case of a course focusing on programming, not for its direct or future usefulness (few biologists or pharmacists will have to program), but for its formative side. I have earlier discussed this (Duchâteau, 1990), as other researchers have done before me (Arsac, 1980). It is a valid argument, even if formative and transferable effects can only be noticed after long (Romainville, 1989). This argument, however, becomes shaky and irrelevant if you leave out the algorithmic part in the informatics course for users.

The 'informatics deals with old issues' argument

As I am going to try to show, informatics, when taught in an appropriate way, allows treatment of central issues in other disciplines in a simple and operational way. 'Information' and 'knowledge', or 'form' and 'meaning' are for example at the heart of linguistics, even if the terms 'signifier' and 'signified' are more

common. Semiology is also a concern of computer science, just as 'artificial intelligence' raises philosophical questions. This seems to me to be the main reason for teaching informatics: it deals with problems as old as and suggests specific answers: what is 'meaning', 'intelligence' or 'conscience' (Hofstadter and Dennet, 1987)? What is 'to give a name' (Arsac, 1987)? what is a 'sign'?

Does this justify a place for informatics in university curricula? The answer probably depends highly on what we mean by 'informatics' in this context. I will discuss this in the second part of my paper.

4 NEW DEMANDS FROM OTHER DISCIPLINES

Since informatics classes were started in 1983, two major changes can be noticed.

- Initially, the informatics course for biology, geography and geology students was optional at the end of the fourth year (last year of the program). The course was moved forward to the second year and has become compulsory.
- Initially it was essentially devoted to algorithmics (Duchâteau, 1990). Later on, I was asked to choose more useful contents, dealing with end-users needs. Which in one word means to pass from 'informatics for computer scientists' to 'informatics for users'.

This change is revealing. The key factor in this change is the appearance of the end-user concept linked to software developments in the middle of the eighties. From then on the computer is in a position of secondary importance; software tools now have primary importance. See for a discussion of development stages Van Weert and Tinsley (1994).

Advocated as main characteristics of the use of software tools were: simplicity, efficiency, easiness of use, user friendliness. But these claims were false. The main difficulty was not to find the on-off switch, it was not 'easier than driving a car'. You have to know some essential principles to deal with the multiplicity of software and computerized environments. So we have to find and teach these basic principles in 'informatics for users'.

5 METHODOLOGY

In a course the organization of contents and the methodology used are essential. The 'informatics for users' course is structured around the following.

- Important explanatory principles from which consequences which end-users notice can be easily inferred. Frequently, these principles take the form of slogans or brief assertions:
  - > do not talk about 'informatics' but about 'in**form**atics!';
  - > the computer does not exist;
  - > for **everything** it does the computer needs a program!

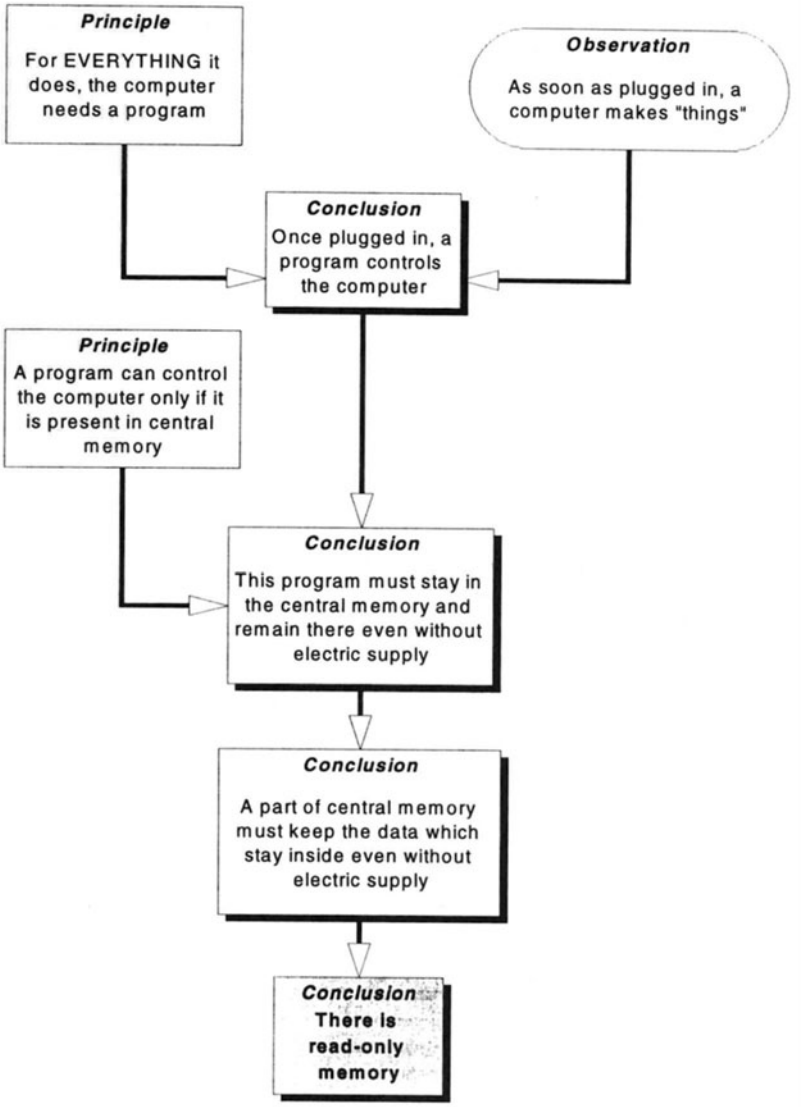

Figure 1 Example of a deductive thought process.

> to **program** is only 'to have it done by';
> if informatics is the science of computers, then meteorology is the science of thermometers.

- Easily understood questions which lead to important facts. For example:
  > where do the programs which are loaded into central memory and which run the computer come from, really?

> what information, whatever its form, needs to be processed by a computer?
> what does the computer send to the printer?

- Main themes which will allow a coherent presentation of the main aspects. For instance, the definition of a computer: 'machine, to process information, in a formalist (formal) way, as long as it is told how to do it'. Or the evolution of programming languages from three different points of view: how is data to be processed specified, what are the basic actions and how are these specified and how are these scheduled?
- Deductive thought processes which starting from stated principles will lead to interesting conclusions to posed questions (see Figure 1).

6 CONTENTS

An incomplete definition of the computer is the skeleton of the first parts of the course: 'machine, to process information, in a formalist (formal) way, as long as it is told how to do it'. The different parts of this definition are scheduled in the way described below

'to process information'
(see Figure 2)
'in a formal way'
(see Figure 3)

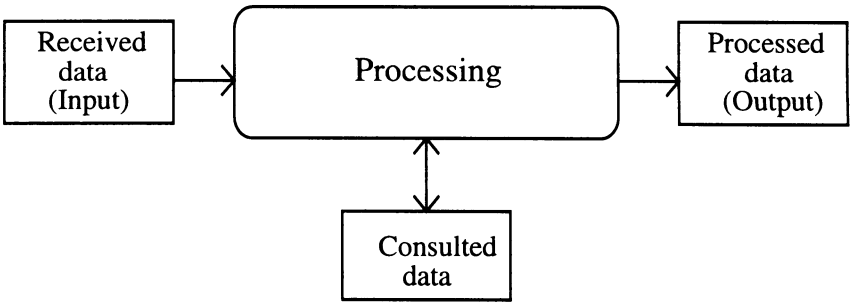

Figure 2 Information processing.

	Formal or easily formalised process	Hardly formalised process / non formalisable process (Meaning, signification, ...)
Processing dealing with numbers		
Processing dealing with language		
Other		

Figure 3 Classification of information processing.

An operational criterion is put forward in order to classify information processing on a continuum (see Figure 4).

Figure 4 Information processing continuum.

This is the 'my Spanish friend criterion': the difficulty of explaining different forms of information processing to this friend, allows us to assess the difficulty of formalization of this processing. Following a definition of informatics is suggested (as a slogan): 'the point of informatics is to transform formalizable processes into formalized processes'. We will be faced with the problem of the intelligence needed to successfully carry this out (see Figure 5).

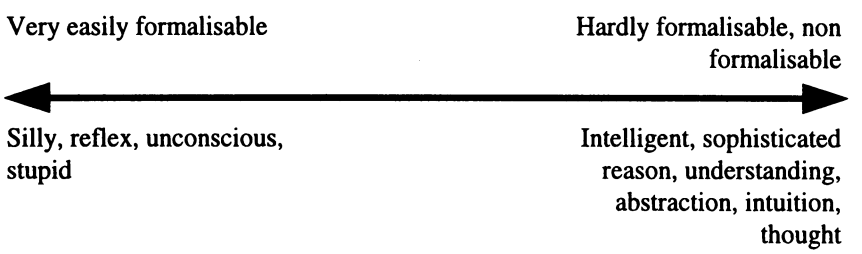

Figure 5 Information processing and intelligence.

'information'

For a piece of information to be processed we have to be able to represent it, to code it as a finite set of integer numbers. This is going to lead to text coding (ASCII or ANSI code), to the coding of real numbers (floating point representation), to the coding of pictures (bitmap or object-oriented coding), to the representation of sounds (sampling and analogue/digital conversion) and the functions of input or output peripherals (coding and decoding). Also problems around optical character recognition and speech recognition are discussed. The whole section results in a final statement: 'We are leaving an analogue era to enter a digital one. And the computer is both the reason for and the means of this transition'.

'as long as it was told how to process'

For everything it does, the computer needs a program (= processing instructions). It is essential that the end-user knows that a computer does not exist on its own. That during a single session of computer work the machine does not change, but the second element, the program, changes several times, sometimes surreptitiously. These observations lead to meaningful consequences. For example, it is impossible to indicate a key on the keyboard which has always the same effect when you press it. 'When using a computer system, nothing is always valid' (this is another important statement). Later we will draw other consequences from these principle: the need for a bootstrapping program in the read-only memory and the need for an operating system to load applications into the random access memory. It is also important to present the programmer's point of view; he is the one who has to create and to provide processing instructions to the computer. The job of a programmer is not 'to do', but 'to have it done by...'. As you could guess, the short sentence 'as long as it was told how to process' covers the whole universe of programming, where you have nothing to do, but where everything has to be done by...

'machine'

The main question here is: How does Figure 2 change when dealing with hardware? Figure 6 shows the two things the computer needs: data to process and programs explaining how to process these. I also discuss, stressing their coding or decoding function, various input devices: keyboard, mouse, scanner, microphone and analogue/digital converter. And output devices: screen, printer. The time has also come to give some assessment details about the processor and the central memory and to tackle operating principles, without too many technical details. The usual jargon has to be defined : bit, byte, binary coding, etc.

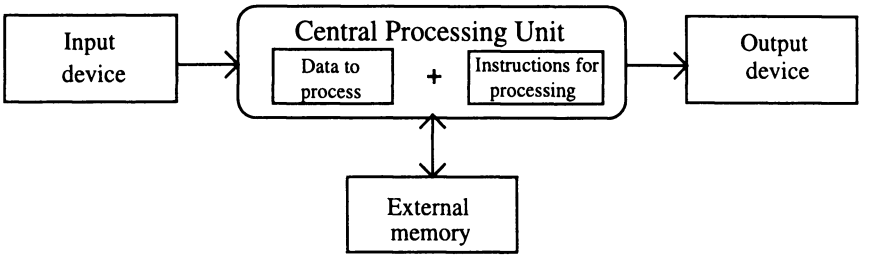

Figure 6 The machine.

Of course this is not the end of the course. There is a section dedicated to the evolution of programming languages and dealing with the specific difficulties of programming. See Duchâteau (1990). Another section is devoted to the history of computer science and focuses on micro-computing. See Duchâteau (1992). Software (operating systems, application software) is also dealt with, stressing the evolution of user interfaces.

7 CONCLUSION

I hope I have shown the spirit of the course and the way it is taught. I hope I have answered, at least partially, the question regarding the definition 'informatics for users'. Maybe the knowledge and mental representations which end-users need can only be learned through actually using computer systems (Duchâteau, 1994). In any case, many teachers in university first year courses are faced with the same content and methodological issues. I have tried to contribute to a joint reflection, which is more than ever necessary.

8 REFERENCES

Arsac, J. (1980) *Premières leçons de programmation.* Cedic / Nathan, Paris [in French].

Arsac, J. (1987) *Les machines à penser. Des ordinateurs et des hommes.* Editions du Seuil, Paris [in French].

Duchâteau, C. (1990) *Images pour programmer. Apprendre les concepts de base.* De Boeck-Wesmael, Bruxelles [in French].

Duchâteau, C. (1992) *L'ordinateur et l'école ! Un mariage difficile ?.*CeFIS, Facultés Universitaires Notre-Dame de la Paix, Namur [in French].

Duchâteau, C. (1994) Faut-il enseigner l'informatique à ses utilisateurs?, in *Actes de la quatrième rencontre francophone de didactique de l'informatique*, AQUOPS, Montréal [in French].

Hofstadter, D. and Dennet, D. (1987) *Vues de l'esprit. Fantaisies et réflexions sur l'être et l'âme*. InterEditions, Paris [in French].

Romainville, M. (1989) Une analyse critique de l'initiation à l'informatique : quels apprentissages et quels transferts ?, in *Actes du colloque francophone sur la didactique de l'informatique*, Editions de l'Epi, Paris [in French].

Tinsley, D. and van Weert, T. (eds.) (1994) *Informatics for Secondary Education. A Curriculum for Schools*. Unesco, Paris.

9 BIOGRAPHY

Charles Duchâteau was born in 1946 in Belgium. After studying to be a primary school teacher, he undertook a degree in mathematics at the Catholic University of Louvain and then a Ph.D. (about incomplete information differential games) at the University Notre-Dame de la Paix in Namur. It was within the Mathematics Department of this university that he founded the CeFIS (a centre for the training of teacher in informatics). He is now member of the Education and Technology Department. His research field is that of informatics didactics and the meaning of the elements constituting an 'Information Technology Culture'.