(Preliminary version of book chapter, in *The Future of Computer Science Education*, A. Greening (ed.))

New Technologies in Computer Science Education

R. Jiménez-Peris*, C. Pareja-Flores**, M. Patiño-Martínez* and J. Á. Velázquez-Iturbide***
* Facultad de Informática, Universidad Politécnica de Madrid,
Campus de Montegancedo, 28660 Boadilla del Monte, Madrid, Spain, {rjimenez,mpatino}@fi.upm.es
** Depto. de Sistemas Informáticos y Programación, Universidad Complutense de Madrid,
Avda. Puerta de Hierro, 28015 Madrid, Spain, cpareja@eucmax.ucm.es
*** Escuela Superior de C.C. Experimentales y Tecnología, Universidad Rey Juan Carlos,
Camino de Humanes 63, 28936 Móstoles, Madrid, Spain, a.velazquez@escet.urjc.es

1 Introduction

Learning and teaching are suffering drastic changes thanks to technology advances in the education field. This is especially true when talking about computer science. These advances are favoring the emergence of new styles of learning and teaching [Rei95]. This chapter describes such advances and their impact on education, particularly computer science, and it also foretells how computer science education will evolve, describing a day in a 2020 University.

We have identified the following issues related to teaching and learning: lecture (presentation and note taking), tutoring, courseware, laboratories, grading, search for bibliographical and other pedagogical material resources. However, the chapter is structured around new technologies, as sometimes a technology has applications in more than one area. We start by describing the applications of the technology and then the technical aspects of the technology itself. These are the new technologies we describe:

- Multimedia-conferencing and computer-supported cooperative work.
- WWW and Internet.
- Automatic grading and plagiarism detection.
- Ubiquitous computing.
- Algorithm animators.
- Electronic books.
- Digital libraries.

Then a day in a 2020 University is described and finally some conclusions are presented.

2 Multimedia-conferencing

Asynchronous communication media (e-mail, news groups, ftp, etc.) has been widely used in the academic community. In the last years, with the advent of faster networks new synchronous communication media have appeared. They are usually referred to as multimedia-conferencing systems.

Multimedia-conferencing supports meetings among people in different locations. The simplest systems only allow unidirectional communication from a source to multiple people. The most advanced ones allow multi-party conferencing, so that each participant can participate both passively (by simply watching and hearing) and actively (by speaking). Technologies discussed in this section are part of a wider area: computer-mediated communication [Rap91] that deals with the interaction between people by means of computers.

Multimedia-conferencing has many applications, although the most important one is tele-teaching or distributed electronic classrooms. Lectures can be broadcasted via networks (ISDN or Internet), using a multimedia-conference system. For instance, a course can be shared by two or more centers [NEELB94], that is, students can be located at different centers, and the lecturer can be at any of the centers. Some advantages of multimedia conferencing are:

- It provides support for distant education (via distributed classrooms), so education can reach remote places [Sal95], providing a new alternative for open universities.
- Courses can be shared among centers [Ped], making possible international/inter-center joint teleclasses [Aok95].
- It can make easier multidisciplinary curricula.
- It provides new possibilities for students with disabilities as they can save displacements, and they can advance at their own pace.
- Lectures can be recorded, what allows students repeating some parts of the lecture, or attending it asynchronously (although in this case without the possibility of interaction).
- Lectures can be enriched with additional annotations (this will be explained further in section 5).
- Another important application is tele-tutoring. This technology enables the possibility of face to face interaction between a lecturer and a remote student.

There are currently two families of standards: ITU (International Telecommunication Union) and MBONE [Savetz]. ITU has developed its H.3xx and T.12x recommendations to support multimedia conferencing, both for circuit switched networks (H.320, H.324) and for packed switched networks (H.323). MBONE videoconferencing tools [Eri94] are based on IETF (Internet Engineering Task Force) standards and widely used in the academic community. These tools have been developed in part within the MICE ESPRIT project [Sas95, Mice] (Multimedia Integrated Conferencing for Europe) and the MERCI project of the Telematics Applications Program [Hug97, Merci] (Multimedia European Research Conferencing Integration). One of the goals of the MERCI project is to allow interoperability between both standards.

Multimedia-conference systems provide a set of tools to support real-time interaction. The most common tools are real-time video, real-time audio and shared white-boards. The first two allow the transmission of video. White-boards allow the presentation of electronic material to all participants, any of the participants can contribute, and his contribution is seen by all participants.

Mbone tools are freeware (they can be get at [Merci]) what make them especially attractive for educational purposes. Next, they are briefly described:

- sdr. A tool for announcing a scheduling multimedia conferences on the Mbone. Using this tool, users can see what conferences are available as well as to join them. They can also use it to announce conferences.
- Vat and Vic. They are real-time multi-party audio and video conferencing tools. It usually provides audio for a full videoconference.
- Rat. It is a robust audio tool that it is designed to cope with packet loss and disordered packets.
- Wb. It provides a shared whiteboard on which participants may write, draw and type making all contributions visible to all participants. It can also import PostScript pages.
- Nt. It is a cooperative editor, where people can edit a document simultaneously.

Lectures can be broadcasted and recorded by means of video and audio conferencing tools, whilst their multi-party versions support multi-center courses. One of the main problems with conferencing systems is their real-time requirements. Video is not very problematic as one frame per second can be enough for most occasions and small delays are not very important. On the other hand audio is more fragile, as it can loose very easily its coherence (due to late, lost or disordered packets). There are currently robust audio tools as rat, but in the future with the advent of faster networks, and ATM networks that guarantee bandwidth the situation will improve.

The integration of electronic whiteboards (like wb) in conferencing systems supports the transmission of lecture material (slides, animations, etc.), but also the lecturer annotations on it. In multi-center courses the different participants can make annotations of the presented material being these annotations broadcasted to all, thus supporting discussions between remote participants.

Nowadays, there are conferencing environments, but there are two different standards, and most applications are not interoperable. In the future, most of these tools will be interoperable, so each participant will be able to participate with the platform and conferencing environment of his/her choice. White-boards will allow to transmit the screen of any application, without forcing to save snapshots in a particular format, so the lecturer will not have to spend any effort in exporting its material to a particular format.

It is possible that in a not very far future, that conferencing systems will remove the frontier between remote and non-remote lectures, as remote participants will be virtually in the classroom with the same possibilities as a non-virtual student. Students will be able to enroll in international courses, without moving to the country or city

of the university. Although, there are already completely distant universities, traditional universities will surely offer the possibility of distant education.

3 Automatic Grading and Plagiarism Detection

Assignments grading represent a substantial share of the load of many lecturers. There have been many attempts to diminish this effort while keeping a good quality grading. The automation of project grading involves four processes: project submission, project grading, plagiarism detection and administrative work. Today, programming projects are submitted electronically at many universities. In some places the use of the network only avoids to handle a lot of paper and diskettes. But in other places it is associated to systems that automatically grade the assignments, store the grades in a database and inform both the instructor and students about the results. Some of these systems go further and they detect plagiarism. Plagiarism detection is always interesting, but with automatic grading it becomes mandatory, as the instructor will have little or no contact with projects.

Automatic grading has different applications, the most obvious one is project grading, but it can also be used for assessment in electronic books or to evaluate student progress during lectures. This latter application is not yet used but we think it will be very important in the future. The lecturer will propose exercises during the lecture, students will solve them and exercises will be graded on the fly. A report will be submitted by the grading system to each student, with errors found in their exercises. The lecturer will receive a report with statistics about the exercises, most frequent errors, etc. Thus the lecturer will be able to know which topics have been understood, general misunderstandings, and how to reorient the lecture depending on the audience. In a near future, nobody could think about grading without the assistance of these systems.

Without lost of generality we will center on programming projects in the next sections. All the ideas presented here can be applied to other subjects.

3.1 Project Submission

Project submission can take different forms: diskette, e-mail, web forms and ftp. Diskette submissions are very rudimentary. The approaches using the network will be the only used, probably in the form of e-mail attachments. Web forms can help the e-mail management as the student can be requested to fill up a web form that, in response, will generate a key that will be used to identify the e-mail project submission. Alternative approaches (especially interesting for big projects) are those based on ftp. In this case the student just sends an e-mail with its username and password (this is not necessary when the grading system runs on the same system where projects are developed and it has permissions to access the student accounts). Then the grading system accesses the student account by means of ftp to copy the project.

3.2 Project Grading

Automatic grading systems can be based on static and dynamic analysis. Static analysis is concerned with those aspects that do not require running the program, mainly style of the program and complexity measures.

The analysis of the style of the program guarantees that the student acquires good coding habits. If the style is analyzed, students will receive some feedback about it, so their style can be improved. Style checkers must find out the indentation style used (if any), its adequacy and its coherence along the program: indents must be always of the same length and statements must follow the same pattern of indentation. They must check that blank lines are used where needed, for instance to separate declarative regions, subprogram declarations, etc. Some standard comments can be enforced as well; for instance, in Pascal “end” of subprograms can be completed with a comment with the name of the subprogram, or each parameter can have an accompanying comment telling whether it is “in”, “out” or “in/out”. The GNU Ada-95 compiler, Gnat, provides an option that enforces a coherent indentation style. Other elements that must be checked, probably by the compiler, are unused variables, the use of global variables (there are pedagogical reason to forbid them in the first courses), functions with side effects, etc.

Complexity measures such as depth of loops and conditionals, number of conditionals, can pinpoint locations of the program with high complexity, that could be decomposed further. [BBFH95] count different kinds of

statements, reserved words, number of operators, depth of conditionals... all these values are compared with the developer's solution in order to estimate the complexity of the student's solution. Other tools are based on metrics like McCabe's metric [McC76].

Dynamic analysis requires running the program to observe its behavior. Some properties of the program can be measured [JU97]:

- The correctness of the program. It ensures that the program generates the specified output for different inputs. Programs are run against test data; to check their correctness, software tools can be used. In [Jac91] it is proposed to construct a compiler using Lex and Yacc to compare the output generated with the one expected.
- The efficiency of the program. This is achieved by measuring CPU time with different input sizes or examining the number of times each line of the program is executed.
- Test data coverage. Checking the adequacy of the test data used by each student. Students must learn to test their programs. This approach provides a way to see how good student test data are.

There several ways to estimate project correctness:

- The instructor writes a set of inputs and their corresponding outputs.
- The instructor writes a version of the programming project.
- When the programming project is a library or set of libraries, the instructor writes a program that imports and tests student libraries. In this program input parameter are passed to student operations and their results are compared with the expected output.

In the first two approaches it is necessary to write a program that compares student project outputs with the outputs of the "golden" version. This program can be an ad hoc one or a general one that can be configured to a particular project. The user interface must be specified to be able to correct automatically the projects, so all the projects read input data and write output in the same order and with the same format. In contrast, in the third approach what is specified is the library interface.

Instructors establish the weights and intervals of the scores for each item to be analyzed. Grades are automatically obtained and stored in a database that keeps all the information related to each student. The student receives automatically the grade for the assignment and also a report of the failures.

Students can use the system to check the results of their programs. Thus, they can try to improve them before the final submission in order to obtain better grades. An inconvenient of this approach is that students could use the system to debug their programs, so they would not learn to debug by themselves. Although discounting a percentage of the note per submission can solve it.

3.3 Administrative Work

Administrative tasks are the ones related to storing the programming projects and their marks and generating reports with student marks, statistics and the most frequent errors. Project grades must be stored in a database together with exam grades. Individual reports are sent to the authors of the project and collective reports to the instructor. The system checks for deadlines and informs both the student and the instructor if projects are submitted after the expiration of the deadline. The system can also report on the status of a given student or group assignment on request by the instructor.

Despite all this automation there is some work that instructors must do. They must prepare the project statement, prepare a set of test data or test program and choose the grading criterion. The only way to reduce this effort is by sharing resources among the Computer Science Education community. Some efforts have started from SIGCSE to create a high-quality lab repository [JKGL+97, Kno97].

3.4 Plagiarism Detection

As it was told at the beginning plagiarism detection is essential when automatic grading is used. To motivate the different techniques used, first the kinds of changes that students make when plagiarizing projects are enumerated and then each technique is described, as well as the problems it solves. [Wha90] identifies them:

- Comments, identifiers and formatting.

- Replacing expressions by equivalent ones. For instance, the order of operands in expressions.
- Data types (real for integer...).
- Adding redundant statements or variables.
- The order of independent statements.
- The structure of iteration and selection statements.
- Replacing procedure calls for procedure body.
- Combining original and copied program fragments.

- [DLS81]

The first technique applied to detect plagiarism consists in tokenizing programming projects. This eliminates many changes, in particular: comments, identifiers and formatting. However, although differences among these elements are not significant, similarities are. To solve this, it is possible to generate tables of comments and identifiers and to compare them between pairs of projects. Indentation is a little more difficult, but it is also possible to generate a table with the indentation scheme used in each project.

The second general technique is aimed to compare programming projects. If projects are compared sequentially, any change will yield that projects are different. A better approach is to extract the longest common subsequence, and repeat the process with the rest of the fragments. This approach is robust to local changes like statement exchange, addition and removal. Local changes just yield slightly shorter common subsequences but they will be long enough to detect plagiarism.

Once plagiarism has been automatically detected, the instructor must check and decide whether the projects are really plagiarized or not. Here, tools can be helpful too. Such a tool would be a browser that shows pairs of plagiarized project in two synchronized windows, highlighting the similarities between them. This browser will allow to progress backward and forward through the chain of similarities and will show the properties of each common fragment (length, common comments and identifiers, etc.).

What it is to come in the future are generic correcting systems available to all the academic community, running in most of the platforms. There are some systems opening the way like ceilidh [BBFH95], although future systems will be more powerful and flexible. Plagiarism will disappear due to plagiarism detectors, so programming projects could be given more weight in the final grade, without worrying about plagiarism.

4 WWW and Internet

With the advent of Internet, sharing digital resources is becoming a true revolution in teaching. By means of WWW, it is possible to access hypermedia documents integrating (hyper-) text, static images, animations, and audio, among others [FIP95], as well as hyperlinks to other documents. Web browsers (e. g. Netscape or Explorer) allow navigating through these resources in an integrated way. In the following subsection, we review these resources and their educational use; in subsection 4.2, the advantages and difficulties of search in Internet are focussed, and finally, in subsection 4.2 we consider the educative use of these resources.

4.1 Courseware

Nowadays, many courses have Web pages with courseware (notes, programming project statements, etc.). A web page can serve as the permanent link among students of a particular course, the instructor and the rest of the staff (laboratory assistants, etc.). Web pages can provide an interactive interface with flexible access in several dimensions:

- The temporal dimension, thanks to asynchronous communication.
- The spatial one, because of remote access through Internet.
- With respect to maintenance, changes can be made and are seen inmediatly, so the “updateability” is continuous.

A course web page can mainly hold general information and didactic resources. The former includes a conceptual view of the course (introduction, information about the course like preconditions and goals, evaluation criteria, etc.) and organizational information (lectures, laboratories and supervising schedules, and lecturers’ announcements, such as talks, grade publication, exams and other eventual activities). The latter kind of information, didactic resources, consists in several elements like the following typical ones:

- Bibliography, including both references to physical books or papers, and links to electronic material.
- Lecture notes (they can even reach the sophistication of electronic books).
- Slides.
- Animations.
- Exercises, programming projects, testing data, automatic grading tools (see section 3), etc.
- Links to the lab framework: programming environments, lab material and other resources.
- A newsgroup provides a forum to discuss conceptual difficulties in the course. It can be useful to allow asking questions anonymously, so students can ask for help more easily [Bar97]. It is an efficient medium as questions are answered just once but everyone sees the answers.
- FAQ list, compiling the most interesting questions about the course.
- Other useful tools, like search engines, glossaries (or even dictionaries), access to the campus library, etc.

4.2 Search in Internet

Sharing resources through Internet offers a main advantage: a lot of effort can be saved. However, it is necessary to devote some effort to look for them. The main difficulty is to deal with the huge amount of information available in Internet. Moreover, anybody can include new resources and update them when references become obsolete.

Fortunately, some solutions have been devised to solve this problem. Advanced information retrieval techniques are being studied and applied. There have been three approaches:

- Search engines are hosts devoted to search and index the information available in the WWW. Some of them have indexed millions of pages like Altavista, Lycos, etc. They usually provide means of filtering the query results by means of tables of keywords.
- Web libraries are compilations of available resources in Internet, hierarchically classified by topic categories, from the most general one to the most specific one. Known examples are the WWW Virtual Library, EINet Galaxy, Planet Earth, Yahoo, etc.
- Web repositories are as well compilations of resources, but they hold links to resources on a specific topic. An example of computer science education repository is the SIGCSE Computing Laboratory Repository [JKGL+97, Kno97], a collection of editor revised on-line laboratories. Another huge CSE repository is described in [McC, McC97], that collects links to resources that support the teaching of computer science education; many educators have contributed with materials and suggestions. A complete repository about algorithm visualization and animation and visualization systems can be found in [ITiCSE97]. Finally, many FAQs can be considered as well like a sort of resources repository, appropriated for a first contact with specific subjects, e.g. functional programming.

These sites are usually maintained by universities, research centers and companies.

4.3 How to use these resources

Despite the great amount of resources and tools available, lecturers have still to spend some effort. They must look for and select the most convenient resources to offer to their students, tuning the depth and difficulty level, choosing the syllabus, the sequence of topics selected, adapting the approach of the materials to his/her course, etc. Besides, he/she must organize these resources (in a web page), and maintain them, including mail-lists of students, adapting animations or exercises from other courses, etc. Some of the effort of building web pages is disappearing thanks to HTML editors and HTML filters (LaTeX2HTML, TexSpider, Netscape composer, Microsoft Word, etc.) and appropriate environments and tools [CGKS+96, Nis96].

On the other side, students can access basic and complementary resources recommended by their lecturer. Some resources can be closed (that is, the student does not leave the course site), while others can provide links to external resources as well. Moreover, they have the possibility of using the search engines in order to obtain additional material.

5 Ubiquitous computing

Ubiquitous computing enhances computer use by making many computers available throughout the physical environment, while making them effectively invisible to the user [Wei91, Wei93]. This research was initiated at Xerox PARC in 1988.

From this research three different kind of ubiquitous computers have resulted. One of the main differences among them is their size. The smallest of them, called *tab*, is the size of a palm, and provides as user interface a pressure-sensitive screen and three buttons, as well as an infrared (IR) receiver to sense its position within a building. This small appliance is capable of providing four-frames-per-second video rate and the IR bandwidth is capable of delivering it. The medium size computer is the *Pad* that it is of the same size of a notebook. It provides an X-Window-compatible writing and display surface. It can communicate with its environment both via IR and radio links (>64KB/sec). It can be used with a tether that provides recharging and operating power as well as an Ethernet connection. The big size computer is a bigger Pad, called *Board*, the size of a white board.

There have been some experiences to use ubiquitous computing in the classroom. One of the most interesting is the one carried out at Georgia Tech [AAFH+96]. In this paper it is described an on-going project, the classroom 2000, to use ubiquitous computing to capture automatically all the information generated during lectures in an integrated way and preserving the relationships among them. The approach of ubiquitous computing is less intrusive than the a-computer-on-each-desk approach, but the latter will be still necessary for some kind of lectures (e.g. in those where it is necessary to program during the lecture). In particular, programming course lectures can be more complete, when students try to solve on the fly problems proposed in the lecture.

In the classroom different forms of communication co-exist, in particular: speech, gestures, projected images (animated or not) and writing (both by the lecturer and the student when taking notes). There are few facilities to record the different materials and preserve the relationships among them. The different sources of information generated and/or used during lectures are: presentation material (slides and/or animations), public notes, private (lecturer) notes, lecturer talk, lecturer-student discussions, lecturer writing and student notes (writing notes or highlighting on lecturer provided material).

Depending on the role in the lecture, the required tools are different so we must deal with lecturer tools, whose goal is to aid the presentation and student tools, whose aim is to aid the note-taking process. The profile of these tools will be different depending on the teaching style and the note-taking style.

Let's study which tools are needed both for the lecturer and the student and what support is necessary from tools to capture the relationships between different means of communication. The main material is the presentation that can take two forms: electronic presentation and slides. The use of slides forces to capture the relationship between lecturer gestures and speech by means of video. Slides must be generated electronically to provide them to students so they can use their ubiquitous computers to take notes on them. Paper based approaches are also possible, as the one proposed by [Sti96]. For an electronic presentation, other tools can be used:

- A pen-based computer with an attached LCD projector.
- Projecting onto an upright digitizing tablet.
- A *board* such as Xerox Liveboard.

The first one is the cheapest option and the third one is the "ubiquitous" option. In any case, the annotations made by the lecturer onto the projection (or the pen-based computer in the first case) can be captured and automatically associated to the projected material. Speech can also be associated to slides by time-stamping each of the slides with the period during which they have been presented. In this way, all the speech that happened during the presentation of a slide is associated to it. If dynamic presentations are used, the synchronization between them and lecturer annotations is more difficult. A possible approach would be to time stamp the steps of the animation (if it is organized in steps) and use these timestamps to synchronize speech (from lecturer and/or students) and lecturer written annotations. All the synchronization between the presented material, speech and lecturer's written annotations must be supported by the tool used by the lecturer to show the presentation. The usual approach is that a camera (with microphone) is attached to the lecturer equipment.

Student tools share some features with the tool used by the lecturer. Students can take notes with different tools:

- A pen-based computer or a pad.
- A PDA (Personal Digital Assistant) or a tab.

Both tools yield different approaches. With the pad approach students are provided with the presentation material and/or notes. Students can follow the presentation on their pads and take notes over the material.

To be able to use tabs, a different strategy must be followed. Tabs have a screen too small to show slides, and to take extensive notes. However, they can present the outline of the talk, and the student can navigate through it and take short notes associated to the slides.

There is a post-production phase where student notes are integrated automatically into lecturer annotations, presentation material, audio and video. This integrated material is made available to students. In this way the student will have access to all the sources of information and they will be all integrated, the student will be able to advance slide by slide and see his/her own annotations, the lecturer annotations, and video and audio of the lecture.

Ubiquitous computing (if successful) will represent an advance as important as GUIs (Graphical User Interfaces) and the mouse. In [Wei98] other uses of ubiquitous computer within the campus are described. Faculty staff and students can carry tabs that will tell where they are, where his/her friends/colleges are. They will be personal agendas extended with interaction with their environment, they will show maps of the campus, they will help to find people, they will allow to read e-mail or even to communicate with other people within the campus, etc.

In the future, the price and the usability of ubiquitous computing will increase dramatically the use of technology into computer science education. Conferencing systems and lecture recording tools (that preserve the relationships among the different sources) share many features. Future systems will provide both facilities in an integrated way, so everything that has been commented in this section could be done remotely as well. The student will also be able to attend recorded lectures and still take notes within the framework exposed in this section.

6 Program Animation

The dynamic visualization of abstract concepts is a good way to facilitate their understanding by students. Animations are dynamic representations that evolve, thus showing the behavior of algorithms and concepts. They are important in computer science because many dynamic concepts are taught (in particular, algorithms and programs). In [BGGR97], three sorts of animation tools are considered:

- **Program visualization systems** show the execution of the code and data in a program, as a sequence of snapshots. Thus, program visualization systems are sometimes integrated into programming environments, such as visual debuggers, showing graphically the dynamic behavior of instructions and data. This kind of tools visualizes any program in a particular way. Examples of these systems are Visual HIPE [JPPV96, JPPV97b], VisMod [JP96] and DynaLab [BBGP+95].

- **Algorithm visualization systems** allow the annotation of programs to generate visualizations. In contrast to program visualization systems, the user must develop the animations, deciding what must showed and how. Despite of the effort request to build animations, this systems allow reusing the material shared by different algorithms, making easier this effort. Examples of these systems are Tango [Sta90], Balsa [Bro86], etc.

- **Concept animation tools** allow showing a graphical representation of abstract concepts, and their temporal evolution explains their behavior. Examples of applications can be visualizations of automata [...], the digital computer structure [...], the “locker metaphor” [JPPV96a], etc.

Besides, a fourth type of animation tool is considered in [BBGJ+96]:

- **Data visualization,** that shows operations inherent to an abstract data type.

Algorithm animation is nowadays an active research and application field, and many systems are being developed, more and more powerful. A view of visualization systems can be found in [WAK96, Trilk97, PBS94]; the latter reference includes a taxonomy of visualization systems, organized by the following aspects: scope, content, form, method, interaction and effectiveness.

The main features required to these systems are flexibility and portability. Flexibility can be achieved in three ways: by parameterization of data, different ways of interactivity, and simultaneous visualizations of algorithms and data.

- Parameterization: it must be possible to construct parametric animations for different set of input data or even that the input is introduced at run-time. Although the use of this feature requires more work from the lecturer, it makes easier the understanding of concepts by the student.
- Interactivity: the user (student) must be able to control the progress of the animation: speed (slower or faster), direction (backward and forward), the level of detail, etc.
- Simultaneous visualizations: the system must be able to show simultaneously different algorithms and/or visualizations of the same data, running at the same time, so students can compare them. In that case, it is necessary to include synchronization mechanisms in order to show their relative behaviors. Tango [Sta90] is an example of such a system.

Portability can be achieved in two ways: by animations based on widely extended compilers and/or interpreters (like Java, C++, Ada and Java bytecode), and by Web-based animations that make use of scripting languages (like JavaScript).

6.1 When and where are animations useful for?

Animations are useful to explain many programming concepts. Examples of these topics are the following:

Subprograms: activation record, scope of identifiers [NS96], stack and tree of recursion. [...Cuando est'en las figuras, se debe remitir a ellas y/o completar con alguna m'as...]

Data structures: array sorting algorithms, dynamic data structures. [WBV97, BV98]

ADTs: algorithms with ADTs can be better understood by means of conceptual representations, independent of (or complementary to) their actual implementation. A good example is the "heap", usually implemented by means of an array, but whose operations are better explained on a conceptual binary tree. Another example is the graph: algorithms considering its abstract representation like a road map can be compared with algorithms based on its usual implementations.

Other concepts: structure of a computer, digital representation of information, algorithm complexity, etc.

On the other hand, animations are useful in different occasions:

- During the lecture, the lecturer can use animations previously developed in order to show several aspects of his/her explanation.
- At the same time, these animations can be made available to students, like executable notes, they can run them later, perhaps with different sets of data, and with the possibility of tuning the progress rate. It is interesting the possibility of including small tests, useful to assess students knowledge.
- Animations can be included into Web pages, electronic books, etc. in order to make them widely available.
- Another possible approach is to let students develop their own animations [Rod96, SIGCSE-97-Stasko].

Nowadays, the use of animations is not widely spread because of some difficulties that are progressively disappearing. First of all, high-quality animations demand an important amount of effort, but the WWW provides a suitable means to share these resources [BJMN+97]. As a consequence, lecturers can focus their attention on searching and selecting the appropriate animations that better fit into his/her course. Some animation repositories are [ITiCSE97, BruVan98].

In [Bergin 97], different methods to share resources in the Web are studied. These methods are different in the following aspects: where (server or client) the visualization system is installed and run; and where the animated algorithms are located. A possibility is to be stored and remotely run in the server, while other possibility is to be stored in the server, and downloaded into the client to be executed.

The main drawback in some approaches is the system-dependency of the code in which the algorithm visualization software was written [BJMN+97]. These barriers could disappear with some widespread languages like Java, that can be interpreted by web browsers [BJMN+97] and that provide the required operating system independence making possible its remote execution.

As a conclusion, these drawbacks will be overcame in the next years, reducing the work of lecturers to develop, reuse or search for animations; besides, they will be more system independent. On the other hand, the benefits in teaching are several, making easier the understanding of difficult concepts, especially in programming.

7 Electronic Books

Since the invention of press, books printed on paper have been the fundamental vehicles for knowledge transmission. The printed book has a number of advantages, from comfort of the reading position to the facility of its transportation (in general) and its autonomy for being handled. However, it also has many disadvantages, such as the difficulty and high cost of modification, difficulty to be shared, facility for being damaged, and its static nature. With the development of new technologies, variants of the classical book have emerged which try to mitigate these drawbacks.

The concept of electronic book [Bar94] refers to a new form of book whose pages are not made of a physical material, such as paper and ink, but they are stored and interpreted in a digital device. Generally, an electronic book can be thought of as a multimedia program, where the user can move through its different parts. Of course, reading an electronic book requires a computer, and probably several specialized input/output devices for interaction, e.g. a loudspeaker or connection to Internet.

There is a great variety of formats for electronic books. Sometimes, they imitate the format of paper books in order to be more familiar to the new reader, with functions to go forward or backward one page, to move in a page, to go to the first page or to any other page, etc. In general, electronic books have a hypertext structure, where a concept allows following a link to other page. In this case, not only a richer structure is available, but it can also provide added values. Electronic books can also be multimedia, so that they include images, graphics, video, sound, etc. in addition to text. Finally, they can allow connecting to Internet, where the possibilities of additional information are almost infinite.

Let us see some non-exhaustive criteria of classification:

- Editing means of the electronic book. Basically, there are two media, CD-ROM and Internet, of course complementary. In the latter case, the reader can access other resources by means of Internet, or even the book can be distributed among several sites, with each part made by a different author.
- Capability of person-book interaction. Some books are completely static, whereas others permit different ways of interaction, such as evaluating the level of understanding by means of questions or to animate certain presentations. Animation features are especially useful for certain programming topics which are hard to understand statically, e.g. recursion. Another useful feature is their modifiability, for instance, to add new materials.
- Relation to other programs or systems. Some books simply contain descriptions proper of a traditional book, but other ones allow interacting with programming environments, Internet or other programs. We can speak about closed or open corpus. A good example of an open electronic book is BURKS-2 [B97], a CD-ROM oriented to computer science students. BURKS-2 can be browsed with Netscape as a simple electronic book, but its capabilities are further enhanced if the computer is connected to Internet, allowing downloading different computer science materials, e.g. programming environments.

Electronic books can be used, at least, for the same purposes than conventional books, that is, as a mean to transmit information in the form of lessons and proposed or solved exercises. A particular use, similar to that of simulation programs, is the presentation of situations which otherwise are very difficult, expensive or just impossible to produce, for instance some real-time systems. In addition, the new media allow widening the pedagogical uses of these books, for example, for self-study. These possibilities usually depend on their interaction capabilities, such as self-tests or customization to the reader preferences. In this way, they give possibilities of self-study similar to those in distant education and training.

Open books give more possibilities for student inquiry. The book provides the basic materials of any subject, but there are connections to other systems or resources which can be used by the student as complementary resources, to deepen in a given topic, to know alternative points of view, etc. Furthermore, the interaction to other programs (electronic dictionaries, programming environments, Internet, etc.) shows that we are in a process where traditional concepts are coming together, and we need new concepts and terms for them.

8 Digital Libraries

The term "digital library" is being used since several years ago to name depositories of material in electronic format. In fact, such a term is not precisely defined. To some it simply suggests the automation of traditional libraries. To others, it will mean a revolution in materials, the kind of stored information, its management, and its use.

Libraries always have stored information in different formats, although currently the materials diversity is much greater: libraries of programs, objects (software), images, video, etc. We can even consider as primitive forms of library the contents of the Web, databases or knowledge bases, etc. In any case, the Web would be a library without a catalog. History shows that rarely a material has completely substituted another, so we can foresee that paper will still be used. Most probably, "digital libraries" will contain electronic materials, paper and other materials. In fact, we could simply speak about future "libraries".

Libraries will contain a great diversity of contents, not only of materials. They will contain fixed and permanent materials, such as current books, but it will also be common to store variable materials (with different versions) or materials whose duration will be limited, such as drafts, "listservs" messages, etc.

An important part of documental work is the interchange of information among readers [LM95]. Libraries will foresee it, for instance, by integrating e-mail facilities in library applications.

The largest challenge today is library interconnection [LG95]. The main obstacle for interconnection is massive handling of information through the net in several aspects: transparent access to information, formats and protocols independence, etc. The treatment of variations in contents or meaning will also be crucial.

Digital libraries are an active area of research [ACM95, Com96]. Organizations [DR95] and institutions [NA88] are involved in research projects, being the most famous the Digital Library Iniciative [Com96] in the USA, but there are similar projects in other countries as well. Some key areas require further development before a true electronic librarian service comes to reality:

- Electronic publishing. Standardization facilitates format conversion between printed (PostScript) and electronic format (for example, SGML). Facilities to add value (side annotations, generation of abstracts, hypertext links) also impact the publishing and organization of materials.
- Authoring matters. More advanced means and techniques for electronic publishing (at least hypermedia and multimedia) will allow authors to enrich their materials and to collaborate more effectively. Copyright issues also are very important [Sam95].
- Educational use. New materials and ways of organization will interact with their main customers, students and lecturers [MM95].
- Data and information management. Specialized technologies are needed for compression and multimedia information. Database management methods are required to control the variety of activities in a library: catalogs, royalty administration, security control, etc. Text analysis and information retrieval techniques will allow converting, indexing, representing, etc. information. Improved human-computer interfaces are crucial to help users in effectively searching, learning, organizing, and using the vast amount of materials that will be found on a library.

9 A Day in the Life of a 2020 University

I am just finishing to prepare the next set of lectures. They are about dynamic memory. This used to be one of the most difficult topics, but now it is just another topic, thanks to available tools. I will introduce it with the "locker metaphor" [], a metaphor that consists of a locker room, locker keys and key rings. To assist the presentation, I will use LockerRoom [], a Java tool that animates the metaphor. Then I will introduce dynamic memory and its relationship with the metaphor elements. LockerRoom also helps in this as it just changes the picture of lockers by that of dynamic memory cells, keys by memory addresses, and key rings by boxes (representing pointers). Then, I can continue with programs working with dynamic memory animating them with VisMod []. VisMod is a visual debugger that presents recursion and dynamic memory in a graphical and intuitive way. All these tools are integrated in the electronic notes of the course. Later in the course, an algorithm animator is used to give a more abstract view of some algorithms.

The data structures course deals with both imperative and functional languages. For the functional implementations I use VisualHipe [JPPV96b, JPPV97], a multi-lingual interpreter of functional languages that shows graphically functional expressions. This allows following the behavior of a function working with a data structure step by step in an understandable way.

All lectures are broadcasted and recorded by means of a conferencing system. Thus, anyone can attend them synchronously (locally or remotely) or asynchronously. The conferencing system permits the lecturer to control interaction from local and remote students, and moderate discussions (deciding who will ask next, when to stop discussion, etc.). Some classrooms have a computer per one or two people, while others do not have any computer. In both cases students use the computer to annotate the material presented by the lecturer. Student notes will consist of the material provided by the lecturer, and the recorded lecture, including lecturer comments and annotations and student annotations. Students are provided with a browser application; it reproduces lectures and it allows students to include further annotations.

Computer lectures are more versatile as student can make class exercises during the lecture. When students finish the class exercise they send their solution by e-mail to the course server that corrects it automatically, sending a message with the errors to the student. The lecturer receives a report with the percentage of errors and the level of the students' answers, and statistics on previous courses. In this way, the lecturer has a very accurate idea of how effective he/she has been up to the moment, and which topics are being understood and which ones are not.

Current lab programming environments are truly educational and they are highly visual and intuitive. They detect and diagnose accurately most errors, thus smoothing the student learning curve. The effort to develop them is high, but they are used in many ways: to assist lectures, as animated student notes and as lab programming environments.

Nowadays, most people have a computer at home, as well as ISDN lines. For this reason most labs are just the materials needed for the programming projects, and not a room with computers. We can speak of open labs at home. In those cases where home computers are not powerful enough or special hardware or software is needed they use remotely department computers devoted to those labs. Students have a number of opportunities to submit their projects (they are submitted electronically), and they are graded automatically taking into account their correctness and their style. The number of submissions is taken into account to compute the grade. Current labs are very elaborated, but the average effort from lecturers is smaller than in the past. The reason is that correction is mostly automatic, and plagiarism detection systems are very powerful, so most of the effort is devoted to prepare lab materials. But the latter has also been minimized with lab repositories especially with the ACM SIGCSE one, so labs can be shared among universities, and a lab developed at a university can be extended and complemented by other ones.

10 Conclusions

We have presented some new technologies and their impact on computer science education. The benefits of using the different applications will be most evident if they are integrated. The experience in other areas is that, at the beginning, they simply support the current style of working, but they are gradually improved and give rise to new tools and ways of working. The same experience will likely happen in the use of educational technologies, giving rise to a new educational paradigm.

With the advent of faster communications and conferencing tools, lectures could be attended remotely, with the same possibilities as if they were attended locally. Furthermore, they could be attended asynchronously, thus giving more flexibility to students. In the future, the frontier between distance and traditional education will become fuzzier. The integration of ubiquitous computing with conferencing tools will allow the automatic integration of all the information sources (lecture material, video and notes taken during the lecture) within the student notes. Thus, the student will devote most of his/her time to study and will waste less time organizing the different sources.

As it is becoming usual, a course Web page will exist for any course that will integrate all its information and material. Although there are an increasing number of Web site building tools, the customization of these tools to education, as well as new ones, specific to education, will proliferate in the future. In addition, students will have access to many bibliographical resources thanks to digital libraries and electronic books.

Automatic grading and plagiarism detection systems will free the lecturer from the grading task, thus allowing to devote most of this time to prepare programming project statements and support material. They will also allow testing the degree of understanding of some topics during lectures on the fly.

We have illustrated the changes that all the agents involved in computer science education will suffer with a scenario of the daily activity of a teacher. The integration of these new technologies seems to be the highway to change drastically the face of computer science education.

11 References

[AAFH+96] G. D. Abowd, C. G. Atkeson, A. Feinstein, C. Hmelo, R. Kooper, S. Long, N. Sawhney, M. Tani. Teaching and learning as multimedia authoring: the classroom 2000 project.

[ACM95] *Communications of the ACM*, monographic issue on "Digital Libraries", April 1995.

[Aok95] K. Aoki, K. Goto. Educational Application of the Internet: International Joint Teleclass. 1995. http://aleph.pangea.org/HMP/PAPER/021/html/paper.html

[B97] BURKS-2, http://burks.bton.ac.uk/

[Bar94] P. Barker, "Electronic books and their potential for interactive learning", in *Man-Machine Communication for Educational Systems Design*, M. D. Brouwer-Janse and T. L. Harrington (eds.), Springer-Verlag, 1994, pp. 151-158.

[Bar97] D. Barnes, "Students Asking Questions: Facilitating Questioning Aids Understanding and Enhances Software Engineering Skills", SIGCSE Bulletin, vol. 29, num. 4, dec. 1997, pages 38-41

[BBFH95] S.D. Benford, E.K. Burke, E. Foxley, and C.A. Higgings " The Ceilidh System for the Automatic Grading of Students on Programming Courses" in Proceedings of the 33rd Annual ACM Southeast Conference, 1995, pp. 176-182.

[BBGP+95] M.R. Birch, C.M. Boroni, F.W. Goosey, S.D. Patton, D.K. Poole, C.M. Pratt, and R.J. Ross, "DynaLab: A Dynamic Computer Science Laboratory Infrastructure Featuring Program Animation", in Proceedings of the 26th SIGCSE Symposium, SIGCSE Bulletin, vol. 27, pp. 29-33.

[BBGJ+96] J. Bergin, K. Brodlie, M. Goldweber, R. Jiménez-Peris, S. Kuri, M. Patiño-Martínez, M. McNally, T. Naps, S. Rodger, and J. Wilson, "An Overview of Visualization: its use and design", in [ITICSE'96], pages 192-200

[BJMN+97] J. Bergin, R. Jiménez-Peris, M. McNally, T. Naps, M. Patiño-Martínez, V. K. Proulx, J. Tarhio, "Using the WWW as the delivery mechanism for interactive, visualization-based instructional modules". Report of the ITiCSE Working Group on Visualization. ACM Conference on Integrating Technology into Computer Science Education. Uppsala, Sweden, June 1997

[BGGR97] C.M. Boroni, F.W. Goosey, M. Grinder, R. J. Ross, and P. Wissenbach, "WebLab! A Universal and Interactive Teaching, Learning, and Laboratory Environment for the World Wide-Web", SIGCSE Bulletin, vol. 29, num. 1, march 1997, California, pages 199-203

[Bro86] M.H. Brown. "Exploring Algorithms using Balsa II". IEEE Computer, vol. 21, num. 5, pp. 14-36.

[BV98] P. Brummund, J. Vanderhyde, "Hope College Animations". Available at http://www.cs.hope.edu/~alganim

[CGKS+96] D. Carlson, M. Guzdial, C. Kehoe, V. Shah, J. Stasko, "WWW Interactive Learning Environments for Computer Science Education", in [SIGCSE'96], pages 290-294

[Com96] *Computer*, monographic issue on "Digital Library Initiative", May 1996.

[DanHea97] D.D. Dankell, and J. Hearn, “The Use of the WWW to Support Distance Learning through NTU”, in [ITICSE'97], pages 8-10

[Dawson95] K. Dawson-Howe. Automatic Submission and Administration of Programming Assignments. SIGCSE Bulletin 27, 4, Dec 1995. pp. 51-53.

[DLS81] J. Donaldson, A. Lancaster, P. Sposato. “A Plagiarism Detection System”. SIGCSE’81. pp. 21-24.

[DR95] P. J. Denning and B. Rous, “The ACM Electronic Publishing Plan”, in [ACM95], pp. 97-103.

[Eri94] H. Eriksson. Mbone: the multicast backbone. Communications of the ACM 37:8, 1994, pp. 54-60.

[FIP95] M. Fiorito, D. Iovane, and P. Pantano, “An Educational Environment Using WWW”, Third International World-Wide Web Conference, available at http://www.igd.fhg.de/www/www95/proceedings/papers/97/EduEnv.html

[Hug97] Hughes, J., Sasse, M.A. Technical innovations deployed by the MERCI project. Proc. Networkshop 25, Belfast, March 1997, pp. 181-189.

[Hung93] S. Hung, L. Kwok, R. Chan. Automatic Programming Assesment. Computers Education. 20,2, 1993, pp. 183-190.

[Ibrahim95] B. Ibrahim, and S. D. Franklin, “Advanced Educational uses of the World-Wide Web”, WWW'95: The Third International World Wide Web Conference, 1995

[ITICSE96] Proceedings of “Integrating Technology into Computer Science Education”, SIGCSE Bulletin, vol. 28, special issue, Barcelona, 1996.

[ITICSE97] Proceedings of “Integrating Technology into Computer Science Education”, SIGCSE Bulletin, vol. 29, num. 3, Uppsala, sept. 1997.

[Jac91] D. Jackson. Using Software Tools to Automate the Assessment of Student Programs. Computers Education. 20, 2, 1993, pp. 133-143.

[JU97] D. Jackson, and M. Usher. “Grading Student Programs Using ASSYST”. SIGCSE’97 pp. 335-339.

[JP96] R. Jiménez-Peris, and M. Patiño-Martínez, “A Modula-2 interpreter/visualizer” in [ITiCSE96], pp 232.

[JPPV97] R. Jiménez-Peris, C. Pareja-Flores, M. Patiño-Martínez, and J.Á. Velázquez-Iturbide, “Visual HIPE: A Prototype for the Graphical Visualization of Functional Programs”, in [ITiCSE97], supplemental proceedings, pp 149.

[JPPV96a] R. Jiménez-Peris, C. Pareja-Flores, M. Patiño-Martínez, and J.Á. Velázquez-Iturbide, “The Locker Metaphor to Teach Dynamic Memory”, in [SIGCSE96], pp 169-173.

[JPPV96b] R. Jiménez-Peris, C. Pareja-Flores, M. Patiño-Martínez, and J.Á. Velázquez-Iturbide, “Graphical Visualization of the Evaluation of Functional Programs”, in [ITiCSE96], pp 36-38.

[JKGL+97] D. Joyce, D. Knox, J. Gerhardt-Powals, E. Koffman, W. Kreuzer, C. Laxer, K. Loose, E. Sutinen, and R.A. Whitehurst . Report of the ITiCSE’97 Working Group on Designing Laboratory Materials for Computing Courses: Developing laboratories for the SIGCSE Computing Laboratory Repository: guidelines, recommendations, and sample labs.

[Kno97] Deborah L. Knox, “Enhancing Accessibility of Lab Materials”, SIGCSE Bulletin, vol. 29, num. 4, dec. 1997, pages 20-21. Repository available at http://www.tcnj.edu/~compsci/

[LG95] C. Lynch and H. García-Molina, “Interoperability, scaling, and the digital libraries research agenda”, Report of the IITA Digital Libraries Workshop, August 1995.

[LM95] D. M. Levy and C. C. Marshall, “Going digital: A look at assumptions underlying digital libraries”, in [ACM95], pp. 77-84.

[MarHur96] A. D. Marshall and S. Hurley, “Interactive Hypermedia Courseware for the World Wide Web”, in [ITICSE'96], pages 1-5

[McC] http://www.cacs.usl.edu/~mccauley/edlinks/

[McC76] T.A. McCabe, “2ª Complexity Measure”. IEEE Transactions Soft. Eng. SE-2(4), December 1976, pp. 308-320

[McC97] R. McCauley, “Computer Science Education Links”, SIGCSE Bulletin, vol. 29, num. 4, dec. 1997, pages 19-20

[McFSte97] Ryan McFall and Gordon Stegink, “Introductory Computer Science for General Education: Laboratories, Textbooks, and the Internet”, SIGCSE Bulletin, vol. 29, num. 1, march 1997, pages 96-100

[Merci] MERCI project. Home page. http://www-mice.cs.ucl.ac.uk/merci

[Mice] MICE project. Home Page. http://www-mice.cs.ucl.ac.uk/mice/mice_home.html

[Ming] L. Ming-Chih, C. Bih-Horng, and Y. Shyan-Ming, “Toward a New Educational Environment”, http://www.w3.org/Conferences/WWW4/Papers/238/

[MM95] G. Marchionini and H. Maurer, “The roles of digital libraries in teaching and learning”, in [ACM95], pp. 67-75.

[NA88] National Academy, “Developing a Digital National Library for Undergraduate Science, Mathematics, Engineering, and Technology Education”, National Academy Press, 1988.

[Nap96] T.L. Naps, “Algorithm Visualization Served off the Word Wide Web: Why and How”, in [ITICSE'96], pages 66-71

[Nap97] T.L. Naps, “Algorithm Visualization on the WWW. The Difference Java Makes”, in [ITICSE'97], pages 59-61.

[NS96] T.L. Naps and J. Stenglein: “Tools for Visual Exploration of Scope and Parameter Passing in a Programming Languages Course”, in [SIGCSE'96], pages 305-309.

[NEELB94] NEELB – Secondary School Videoconferencing. 1994. http://www.infm.ulst.ac.uk/~neelb/vid/nellb1.html

[Nis96] T. Nishida, “Lecture Supporting System by Using E-mail and WWW”, in [SIGCSE'96], pages 280-284

[Pax96] J. T. Paxton, “Webucation: Using the Web as a Classroom Tool”, in [SIGCSE'96], pages 285 - 289

[Ped] G. Pedersen. Distributed Electronic Classrooms. MUNIN-project. http://munin.uio.no/English/intro.html

[PBS94] B. Price, B.A., Baecker, R.M., and Small, I.S. “A Principled Taxonomy of Sofware Visualization“, Journal of Visual Languages and Computing 4(3):211-266. Available at the URL: http://www-cs.open.ac.uk/~doc/jvlc/JVLC-Body.html

[Rap91] M. Rapaport. Computer mediated communications: bulletin boards, computer conferencing, electronic mail, and information retrieval. 1991. Wiley.

[Rein95] A. Reinhardt. New ways to learn. Mar. 1995. BYTE. pp. 50-71.

[Rod96] S. H. Rodger, “Integrating Animations into Courses”, in [ITICSE'96], pages 72-74

[Sal95] M. Salonen. Aspects of Classroom Interaction in Two-Way Videoconferencing. 1995. http://222.nks.no/fjernund/worddoc/21salone.html

[Sam95] P. Samuelson, “Copyright and digital libraries”, in [ACM95], pp. 15-21 y 110.

[Sas95] M.A. Sasse, R. Bennett. Multimedia conferencing over the Internet: The MICE project and tools. Proc. of NetWorkshop23, 1995.

[Savetz] K. Savetz, N. Randall, and Y. Lepage. Mbone: Multicasting tomorrow’s Internet. ISBN: 1-56884-723-8. IDG Books Worldwide.

[SIGCSE'96] Proceedings of the 27th SIGCSE Symposium, SIGCSE Bulletin, vol. 28, num. 1, Philadelphia, march 1996.

[Sta90] J. Stasko. “Tango: A Framework and System for Algorithm Animation”. IEEE Computer, vol. 23, num. 9, pp. 27-39, September 1990.

[Sti96] L. J. Stifelman. Augmenting real-world objects: A paper-based audio notebook. In Proceedings of ACM CHI’96 Conference, pp. 199-200, April 1996.

[Tinoco97] L. Tinoco, E. Fox, N. Dwight. Online Evaluation in WWW-based Courseware. SIGCSE’97 pp. 194–198.

[Trilk97] “Software Visualization”. Available at the following URL: `http://wwwendres.informatik.tu-muenchen.de/leute/trilk/sv.html`

[WBV97]. Web Based Visualizations: http://sol.pace.edu/webvis

[Webber96] A. B. Webber, “The Pascal Trainer”, [SIGCSE'96], pages 261-265

[Wei91] M. Weiser. The computer for the 21st century.

[Wei93] M. Weiser. Some computer science issues in ubiquitous computing. Communications of the ACM. V. 36, N. 7. July 1993. pp. 75-84.

[Wei98] M. Weiser. The future of ubiquitous computing on campus. Communications of the ACM. V. 41, N. 1. Jan. 1998. pp. 41-42.

[Wha90] G. Whale. Identification of Program Similarity in Large Populations. The Computer Journal, V. 33, N. 2, pp. 140-146, 1990.

[Wie95] G. Wiederhold, “Digital libraries, value and productivity”, in [ACM95], pp. 85-98.

[WAK96] J. Wilson, R. Aiken, and I. Katz, “Review of animation systems for algorithm animation”, in [ITICSE96], pages 75-77

[Wis96] M. Wise. “YAP3: Improved Detection of Similarities in Computer Program and Other Text”. SIGCSE’97 pp. 130-134.