

Revealing the Programming Process

– Using Videos to Unfold Basic Programming Techniques

Jens Bennedsen
IT University West
Fuglesangs Allé 20
DK-8210 Aarhus V
Denmark
jbb@it-vest.dk

Michael E. Caspersen
IT University West
University of Aarhus
Aabogade 34, DK-8200 Aarhus N
Denmark
mec@it-vest.au.dk

ABSTRACT

The most important part of an introductory programming course is to teach the students a systematic approach to the development of computer programs. Revealing the programming process is an important part of this; however, textbooks do not address the issue – probably because the medium is unsuitable for this kind of descriptions. We have found that videos in the form of narrated programming sessions are a simple, cheap, and efficient way of providing the revelation.

We identify seven different elements of the programming process for which videos are a valuable communication media in order to enhance the learning process. Student feedback indicates both high learning outcome and superior learning potential compared to traditional class room teaching.

Categories and Subject Descriptors

K3.1 [Computers & Education]: Computer Uses in Education – *computer-assisted instruction, distance learning*.

K3.2 [Computers & Education]: Computer and Information Science Education – *computer science education, information systems education*.

General Terms

Design, Documentation, Experimentation, Human Factors, Languages.

Keywords

CS1, Programming Process, Video, Model-Based Programming, Objects-First, Design, Incremental Development, Testing, Refactoring, Programming Education, UML, Conceptual Modelling, Systematic Programming, Pedagogy.

1. INTRODUCTION

We hold the position that the most important part of an introductory programming course is to teach the students a systematic approach to the development of computer programs. Revealing the programming process is an important part of this, and we have found that videos in the form of (screen captured) narrated programming sessions is a simple, cheap, and efficient way of providing the revelation. We hereby expand the applied apprenticeship approach as advocated in [2, 13].

Revealing the programming process to beginning students is important, but traditional static teaching tools and materials such as textbooks, lecture notes, blackboards, slide presentations, etc. are insufficient for that purpose. They are useful for the presentation of the result of a process (a product), but not for the presentation of the process itself. Besides being insufficient for the presentation of a development process, the use of traditional tools and materials has another drawback: typically they are used for the presentation of an ideal solution which is the result of a non-linear development process. Like others [18, 19, 20], we consider this to be problematic; the presentation of the product independently of the development process will inevitably leave the students with the false impression that there *is* a linear and direct “royal road” from problem to solution. This is very far from the truth, but the problem for beginning students is that when they see their teacher present clean and simple solutions, they think they themselves should be able in a straightforward fashion to develop solutions in a similar way. When they realize they cannot, they blame themselves and feel incompetent. Consequently they will lose self confidence and in the worst case their motivation for learning to program.

Besides teaching the students about the tools and techniques to be used for the development of programs (i.e. the programming language, programming techniques, IDE, etc.), we must also teach them about the development process, i.e. the task of using these tools and techniques to develop, in a systematic, incremental and typically non-linear way, a “good” solution for the problem at hand. An important part of this is to expound and demonstrate that many small steps are better than few large ones, that the result of every little step should be tested, that prior decisions may need to be undone and code refactored, that making errors is common also for experienced programmers, that compiler errors can be misleading/erroneous, that online documentation for class libraries provide valuable information, and that there, however non-

linear, is a systematic way of developing a solution for the problem at hand. We cannot rely on the students to learn all of this by themselves, but using an apprenticeship approach we can show them how to do it; for this purpose we use videos.

2. PRESENTING PROGRAMS AND PROGRAMMING TO STUDENTS

Different ways of presenting programs and programming to students are in use: programming on a blackboard, pre-cooked programs on slides, live programming using computer and projector, and videos showing the programming process.

Programming on a blackboard has the advantage that it is possible to create programs in dialog with the students and at a pace the students can follow; also, the teacher and the students can interact during the development of the program. This approach has the obvious drawbacks that neither are we able to run and easily modify the programs nor to use the relevant development tools.

Pre-cooked programs on slides provide a way of presenting larger and more complex programs to the students, programs that we would never consider writing on a blackboard. This approach has the drawback that we as teachers tend to progress too fast and exclude the students from taking part in the development of the programs.

Live programming in the class room/lecture theatre using computer and projector is a combination of using blackboard and slides, but with the important additional ability to run and test the program and to use the programming tools including IDE, online documentation, diagramming tools etc. This, of course, is much closer to the actual programming process than the first two approaches. However, there are still drawbacks: time in the class room is limited and this restricts the complexity of the examples that are presented; also, the presentation vanishes as it takes place; nothing is saved afterwards (except the final product and what might have been captured in the students' heads).

Videos showing the programming process are similar to live programming in the class room/lecture theatre using computer and projector, but without the limitations of live programming. In a video you can take the time needed to present as complex an example as you wish, and (parts of) the presentation can be reviewed over and over as many times as a student needs it.

The first three approaches have in common that they are synchronous, one shot events. There is no possibility for the student to go back and review (a step in) the development process if there were something he did not understand. This opportunity is exactly what is added by using videos, but at the cost of losing interaction.

3. USE OF VIDEOS IN EDUCATION

During the last years streaming video has become more and more popular and common [14, 17]. Compression techniques have been standardized and improved; bandwidth are much larger than just a few years ago (also in private homes) thus making it possible to use videos in an educational setting. Furthermore the appearances of desktop video capturing and editing tools have made it easy for people to create their own videos.

Most examples of the use of videos in teaching are so-called web casts, i.e. recordings of class room sessions and presentations in

lecture theatres. This approach is used by many universities including prominent ones like Berkeley and MIT [5, 15]. While such videos may be valuable to students whom for one reason or another are not able to attend the lecture or would like to have (parts of) it repeated, they do not significantly add new value to the teaching material.

In [16] the authors argue that “the only change over time is the way we store information – writing on stones, leaves, and cloth was changed to using slates, blackboards and now electronic media”. We strongly disagree; using technology (including videos) offers a lot of new possibilities for teachers as well as for students. With new technology, in this case computers and video capturing tools, it becomes possible to store information that represent dynamic behaviour, something which is virtually impossible to describe and represent using traditional tools and materials such as blackboards and books.

4. REVEALING THE PROGRAMMING PROCESS

The idea of revealing the programming process is not new:

Anyone with a reasonable intelligence and some grasp of basic logical and mathematical concepts can learn to program; what is required is a way to demystify the programming process and help students to understand it, analyse their work, and most importantly gain the confidence in themselves that will allow them to learn the skills they need to become proficient.

This quotation is fifteen years old [8]; nevertheless, the issue still has not found its way into textbooks on introductory programming.

4.1 Textbooks do not Address the Issue

At a recent workshop [12], a survey of 39 major selling textbooks on introductory programming was presented. The overall conclusion of the survey was, hardly surprising, that all the books are structured according to the language constructs of the programming language at hand, not by the programming techniques that we want to teach our students. The progression is bottom-up: fundamental issues (variables, types, objects and classes, etc.), control structures (sequence, selection, and iteration), methods (parameterisation, recursion), classes and subclasses, etc. The order may differ, but the overall structure is the same. In other words: learning the syntax of the programming language has priority over learning programming techniques and the programming process. This prevailing textbook approach will help the students to understand the programming language and the structure of the sample programs presented in the text, but it does not show the student how to program – it does not reveal the programming process.

We know what is needed, so how come that the topic has not found its way into textbooks on introductory programming? The best answer is that the textbook medium is unsuitable for this kind of descriptions.

4.2 Elements of the Programming Process

The programming process is characterized by the following elements: use of an IDE, incremental development, testing, refactoring, error handling, use of online documentation, and systematic construction of code from a model. All are they unsuitable for

textual descriptions, but important for the student to master. For each process element we will discuss how to address it in an introductory programming course and how videos can be used to reveal its core aspects.

Use of the IDE: We use a very simple IDE (BlueJ) that hardly needs introduction. However, a short video demonstrating the use of special facilities in the IDE makes it still easier for the students to start using it.

Incremental development: Students often try to create a complete solution to a problem before they start testing it (if ever). This, of course, is not the behaviour we want the students to exhibit; we want them to create the solution in an incremental way taking very small alternating steps of implementing and testing their code. Following this advice makes it much easier to find and correct errors and it simplifies the whole activity. This is a topic that is very difficult to communicate in a book; where do you start, how do you use your programming tool in order to incrementally develop a solution. With a video clip it is simple and straightforward to demonstrate how to behave.

Testing: We promote two simple techniques for testing: interactive testing through the IDE (BlueJ) or the creation of a special class, *Driver*, to hold one or more test methods. The process aspect of the former technique is covered under “Use of the IDE” above (see also [11]). Regarding the latter we suggest that the test code for a method is written prior to the implementation of the method (test-driven development); explicit requirements in exercises to write test code prior to implementation reinforce this.

A textbook is useful for describing principles and techniques for testing (black-box, white-box, coverage, etc.) but how to integrate testing in the development process is best demonstrated using live programming/videos.

Refactoring: When the students read a textbook they get the impression that programmers never make mistakes, that they always create perfect, working solutions in take one, and that they never have to correct and improve their programs. This, of course, is wrong; nevertheless it is the impression a student is left with after having read a textbook. In [7] it is stated that an experienced programmer should expect to use approximately 50% of his time refactoring his code. If this is the case for an experienced programmer, a beginning programmer should expect to use significantly more, so of course a student cannot expect to create the perfect solution in take one. But the students have the impression that they can, and how should they know better?

We have found it difficult to motivate the need for refactoring to students. The goal of refactoring is to create better programs in the sense of exhibiting lower coupling and higher cohesion. The students do not know when it is needed to refactor a program; they consider the job done when the program can compile and run some simple tests. But showing them the refactoring techniques “live” gives them a much better understanding of the techniques and an appreciation of the necessity for refactoring. In order to optimise motivation we often start out with one of the student’s programs, showing how refactoring can make that program much more readable and how lower coupling and higher cohesion can be obtained through successive applications of simple standard techniques.

Error handling: In order to make the students feel more comfortable it is important to show them that every programmer makes errors and that making and handling errors is a normal part of the process. It is important to show the students how errors are handled. In particular it is important to demonstrate to the students that the output from the compiler not always indicates the real error and that there are different types of errors: syntax errors, semantic errors or logic errors. Again, the videos help by being explicit and by dealing systematically with each kind of error.

Online documentation: Modern programming languages are accompanied by large class libraries which the students need to use. The documentation for Java is available online, and the students have to be acquainted with the documentation and how to use it in order to write programs.

When the students write code, we force them to write javadoc too. In order to teach them how to write and generate the documentation, we show them how to do this as an integrated part of the development process using live programming/videos.

Model-based programming: In our CS1 course we use a model-driven, objects-first approach as described in [3]. In order to do so the students need to use more than the traditional programming tools; they need to use a tool for describing the class models. The students also need to understand the interaction between the IDE and the modelling tool as well as the relation between model and code. To reinforce the importance of modelling as an integrated part of the development process it is vital to show the students the different tools in action.

5. VIDEOS IN A COURSE CONTEXT

In this section we will describe how the video mediated materials are used in an introductory programming course (a face-to-face or distance education setting).

5.1 Video Categories

We have created five different types of videos: introduction to assignments, solutions to the assignments, documentation of synchronous activities (lectures and online meetings), alternative teaching materials, and tool support.

Introduction to assignments: Many students struggle with getting started with an assignment: what is the problem, how and where shall I start, what exactly is it that I shall do? Many such questions can efficiently be addressed in a video where also fragments of a solution can be presented.

Solutions to assignments: Presentation of a solution to a programming assignment; besides presenting the solution, we also present aspects of the development process.

Documentation of synchronous activities: By capturing live programming as it takes place, the students get the opportunity to review (parts of) the process at a later stage.

Alternative teaching materials: For the core topics in the text, we create small programming problems to illustrate the use and applicability of the topic. This provides diversity in the course material supporting different styles of learning.

Tool support: We have created different kinds of videos for tool support. Like [1] we have found that, instead of creating written descriptions and manuals for these tasks, it is much easier for us

as well as the students if we create a video showing how to *do* things: just tell what you are doing on the screen while capturing it.

5.2 Student Feedback

In this year we have taught two introductory programming courses based on distance education with respectively 35 and 20 students. A detailed description of the design of this course can be found in [4].

All videos are stored on a web-server and the students can access them whenever they want and from where they want. We have evaluated the use of videos in our CS1 course. The evaluation was done as a quantitative as well as a qualitative evaluation using a questionnaire and by interviewing a number of students about their attitude towards the material. From the questionnaire we can see that more than 2/3 of the students have seen more than 50% of the videos.

During the first half of a CS1 course there were a total of 9800 hits at the website (this includes web-crawlers, but some pages are only accessed once, so we expect the web-crawlers to be responsible only for a small fraction of the hits). Of these, 1650 were hits on the videos. The students use the material (many of the students download the videos once and for all instead of repeatedly streaming them from the server, so the actual use is probably somewhat larger). The distribution of the hits on the different types of video material is as follows: introduction to assignments 28%, solutions to assignments 19%, documentation of synchronous activities 9% alternative teaching materials 21%, and tool support 23%. The interesting thing to notice is that only 9% of the students have exploited the possibility of reviewing the synchronous activities; this indicates that web casting, the most widespread use of videos [5, 15], is the least useful of the five categories.

The students have evaluated the learning outcome of the videos; the result of the evaluation is presented in figure 1.

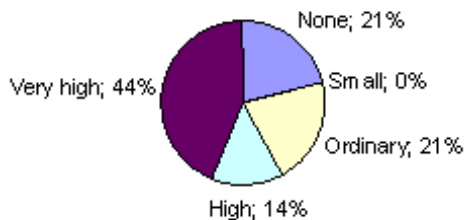


Figure 1. Learning outcome of videos

58% has indicated a high or very high learning outcome which is very encouraging.

In an interview after the course, a student characterized the use of videos as follows:

I claim that the learning potential is better with this teaching form than for traditional class room teaching; in the virtual class room I can eliminate all kind of noise and interruptions. Combined with the opportunity to review (parts of) the session, the return on investment becomes optimal.

5.3 Video Production

The creation of a video is cheap and can be done almost anywhere. There is no use for a special studio or other expensive equipment (we even created a video during an Atlantic crossing). The software for video capturing is free, and depending on how advanced post production one needs, the software is either free or very cheap. We have used Windows Media Encoder and Windows Media File Editor, both freeware programs.

The videos documenting synchronous activities are almost cost free. We do screen and audio capturing during the activity and store the file on a server afterwards.

The student can navigate in the video. In order to give the students the possibility of finding the various parts of the video, we create a simple topic→time index of all videos. We store the description in a database which implies that the students can search for the material at a later stage.

The videos supporting the textbook are a little different. Here we do a little more pre-production; mainly figuring out which examples will be useful in order to explain the concepts in the text. It is our experience that a detailed manuscript is superfluous; too detailed a manuscript tend to make the video less authentic and in the worst case plain boring. We have created approximately 30 videos of this type; it is our experience that we use one hour to prepare a 30-minute recording and another 20 minutes for post-production.

The addition of the topic→time index has added a new usage of the lectures. The students can search the material afterwards and use it as yet another part of their learning material repository. In this way the value of the lectures has expanded from something that is only useful if you are present, to a material that can be used repeatedly over time.

6. RELATED WORK

The use of videos in teaching is not new [17]. Videos are used extensively in [9], but the use is somewhat different from ours:

- All videos are very short and focused on explaining a single aspect of the programming language or programming.
- The videos are “perfect”; they do not show that it is common to make errors (and how to correct them).
- The videos do not show the integrated use of the different tools like IDE, online documentation and drawing tool.

According to our categorization (section 5.1), the videos in [9] can be characterized as alternative teaching materials.

Others use a much richer form of multimedia than plain video. One example is the learning objects discussed in [6]. The same differences as described above apply (also these learning objects fall strictly under the category alternative teaching material), and on top of that the production cost for creating these learning objects is extremely high.

7. CONCLUSIONS

Revealing the programming process is an important part of an introductory programming course which is not covered by traditional teaching materials such as textbooks, lecture notes, black-

boards, slide presentations, etc. This is just as good since these materials are insufficient for the purpose.

We have suggested videos in the form of (screen captured) narrated programming sessions as a simple, cheap, and efficient way of providing a revelation of the programming process. Furthermore we have identified seven elements that characterize the programming process: use of an IDE, incremental development, testing, refactoring, error handling, use of online documentation, and systematic construction of code from a model. For each of these elements we have discussed how to address it in an introductory programming course and how videos can be used to reveal its core aspects.

From our evaluation of the approach we know that the students use and appreciate the videos; some students even find the material superior to traditional face-to-face teaching. The creation of video mediated materials has shown to be easy and cheap as opposed to other approaches to create learning objects.

The advance of new technology in the form of digital media has made it possible to easily create learning material to reveal process elements that in the past has been implicit. The students welcome the new material which has great impact on the students understanding of the programming process and their performance in practical programming.

8. ACKNOWLEDGEMENT

The production of videos was initiated in the Flexnet project under IT University West; we will like to thank IT University West for financial support for the project. We also thank the TA's and students participating in the CS1 course for their patience, feedback and support.

9. REFERENCES

- [1] Alford, K., "Video FAQs – Instruction-On-Demand", *Proceedings of Frontiers in Education*, Boulder Colorado, 2003.
- [2] Astrachan, O. & Reed, D., "AAA and CS1: The Applied Apprenticeship Approach to CS1", *Proceedings of the twenty-sixth SIGCSE Technical Symposium on Computer Science Education*, Nashville, Tennessee, 1995. pp. 1-5.
- [3] Bennedsen, J. & Caspersen, M. E., "Programming in Context – A Model-First Approach to CS1", *Proceedings of the thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, Norfolk, Virginia, 2004. To appear.
- [4] Bennedsen, J. & Caspersen, M. E., "Rationale for the Design of a Web-based Programming Course for Adults", *Proceedings of ICOOL 2003, International Conference on Open and Online Learning*, Mauritius, 2003. To appear.
- [5] Berkeley, <http://webcast.berkeley.edu/courses/>
- [6] Boyle, T., "Design principles for authoring dynamic, reusable learning objects", *Australian Journal of Educational Technology*, 19 (1), 2003, pp. 46-58.
- [7] Fowler, M., *Refactoring – Improving the Design of Existing Code*, Addison-Wesley, 1999. ISBN 0-201-48567-2
- [8] Gantenbein, R. E., "Programming as Process: A 'Novell' Approach to Teaching Programming", *Proceedings of the twentieth SIGCSE Technical Symposium on Computer Science Education*, Louisville, Kentucky, 1989, pp. 22-26.
- [9] Gries, D. & Gries, P., *ProgramLive*, John Wiley & Sons, 2001.
- [10] Iyengar, S. & Soloway, E. (Eds.), *Empirical Studies of Programmers*, Ablex, New York, 1986.
- [11] Kölling, M. & Rosenberg, J., "Testing Object-Oriented Programs: Making it Simple", *Proceedings of the twenty-eighth SIGCSE Technical Symposium on Computer Science Education*, San José, California, 1997, pp. 77-81.
- [12] Kölling, M., "The Curse of Hello World", *Invited lecture at Workshop on Learning and Teaching Object-orientation – Scandinavian Perspectives*, Oslo, October 2003.
- [13] Linn, M. C. & Clancy, M. J., "The Case for Case Studies of Programming Problems", *Communications of the ACM*, 35 (3), 1992, pp. 121-132.
- [14] Ma, W., Lee, Y., Du, D.H.C. & McCahill, M. P., "Video-based Hypermedia for Education-On-Demand", *Proceedings of the fourth ACM International Conference on Multimedia*, 1996, pp. 449-450.
- [15] MIT, www.swiss.ai.mit.edu/classes/6.001/abelson-sussman-lectures/
- [16] Siddiqui, K. J. & Zubairi, J. A., "Distance Learning using Web-based Multimedia Environment", IEEE.
- [17] Smidth, T, Ruocco, A & Jansen, B., "Digital Video in Education", *Proceedings of the thirtieth SIGCSE Technical Symposium on Computer Science Education*, New Orleans, Louisiana, 1999, pp. 122-126.
- [18] Soloway, E., "Learning to Program = Learning to Construct Mechanisms and Explanations", *Communications of the ACM*, 29 (9), 1986, pp. 850-858.
- [19] Spohrer, J. & Soloway, E., "Novice Mistakes: Are the Folk Wisdoms Correct?", *Communications of the ACM*, 29 (7), 1986, pp. 624-632.
- [20] Spohrer, J. & Soloway, E., *Analyzing the High-Frequency Bugs in Novice Programs*, In [10].