Broadening Accessibility to Computer Science for K-12 Education

Mark K. Gardner
Virginia Tech
Blacksburg, Virginia USA
mkg@vt.edu

Wu-chun Feng
Virginia Tech
Blacksburg, Virginia USA
feng@cs.vt.edu

ABSTRACT

Enrollments in computer science and computer engineering have decreased dramatically since the dot-com bubble burst in 2000 even though it is projected that nearly three quarters of all science and engineering jobs in the future will be in these fields. Meeting this demand will require a substantial effort to inspire and motivate students as early as in the elementary school years. The challenge is to provide motivational access to computer science training, particularly for women and minorities in disadvantaged areas.

We consider three approaches for broadening accessibility to software that teaches computer science fundamentals: (1) virtualization, (2) self-contained live CDs, and (3) web browser-based deployment. While the work is ongoing, initial results are quite promising. Next steps include a more formal and much wider deployment to 3rd–5th grade classes in ten elementary schools in rural Virginia.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education; K.3.2 [**Computers and Education**]: Computer and Information Science Education

General Terms

Human Factors, Languages, Legal Aspects

Keywords

K-12, Storytelling Alice, Virtual Computing, Live CD

1. INTRODUCTION

The bursting of the dot-com bubble in 2000 led to a massive hemorrhage in students majoring in computer science (CS) and computer engineering (CE). After peaking in 1999-2000, interest in CS as a major dropped by a whopping 70% between 2000 and 2005 and has since flattened to a point where just over 1% of incoming freshmen indicate CS as their probably major [9]. The above numbers are in spite of the fact that the U.S. Bureau of Labor & Statistics projects that 71% of all science and engineering jobs will be for computer scientists and engineers [8]. Thus, it is imperative that we train the next generation of computer scientists and get them started early, in particular, in elementary school and middle school [13].

One way that many school districts have addressed the above is to improve physical accessibility by increasing their number of computers, thus reducing the student-to-computer ratio. (The U.S. national average is 3.1 students per computer [5].) However, improved physical access does not necessarily translate to improved access to educational software because we argue that school districts largely use the computers as glorified electronic typewriters, e.g., Microsoft Word and Microsoft PowerPoint, that have access to the Internet via a web browser.

Furthermore, increasing the number of computers in schools can dramatically increase the overall cost to the schools due to the acquisition and recurring operational costs for the computer but also the software that runs on it. Thus, from a socio-economic standpoint, the divide between the "haves" and "have nots" continues to grow. For instance, at a rural K-8 school in Virginia, the student-to-computer ratio is *over three times worse* than the national average. As a consequence, there exists a need to be able to deliver educational software to economically disadvantaged areas, which in many cases, means being able to reach out and engage underrepresented minorities.

On a related note, focusing on K-8 students also provides the opportunity to engage, and more importantly, retain the interest of females in science and engineering as it is typically in the middle-school years that females decide that science and engineering is "not cool" [6]. This is circumstantially supported by the fact that 39% of the information technology (IT), i.e., computing, workforce was female in 1980; in 2007, it had dropped to 13%.

To address the need for earlier computer science training, both to interest more students in pursuing careers in computing related fields and to attract more women and minorities into the profession, we propose broadening accessibility to computer science software at home and at school. Specifically, we address the issue of making it easier to gain access to the software while simultaneously eliminating impediments to deployment so teachers have the software available for teaching and students have the software accessible to them at home for experimentation. We also briefly discuss several issues that should be considered in making such software accessible.

ITiCSE 2010 Bilkent, Ankara, Turkey

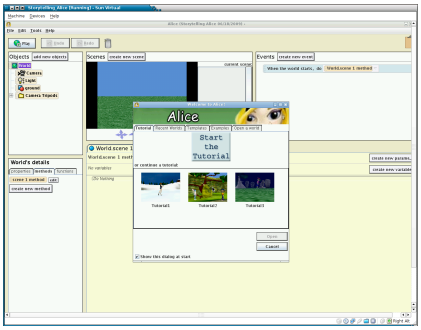

Figure 1: Storytelling Alice Running in VirtualBox.

2. INCREASING ACCESSIBILITY

In this study, we consider three ways to increase accessibility of software and materials to inspire future computer scientists.

2.1 Virtualization

Students need access to hardware, software, and training in order to be prepared to function well in today's society. One successful solution to the problem is accomplished by providing remote access to resources and software through virtualization. The outreach of universities utilizing the Virtual Computing Laboratory (VCL) model [1], developed by North Carolina State University, has been shown to be effective in increasing access to computing resources and software, particularly for under represented minorities. Virtualization enables experts in a field to create virtual machines on behalf of others, dramatically reducing the effort required to provide teachers with materials to support the education of students in disadvantaged regions.

We are developing a framework for deploying virtual machines for use in K-12 schools in rural Virginia. We call the framework SERViCE – a Scalable, Extensible, and Reliable Virtual Computing Environment [7]. It has been used to teach fundamental computer sciences concepts to elementary age students at two local schools using Storytelling Alice (shown running in a virtual machine in Figure 1).

Previous versions of SERViCE utilized VMware Server as the virtualization technology. As we have begun to consider other topics of instruction, we quickly realized that only supporting a single virtualization technology would inhibit the organic growth of a community to create and share virtual machines for education. An approach that accommodates more homogeneity is needed.

By re-factoring and extending the SERViCE framework, we are able to support many, if not all, virtualization options. The framework serves as a prototype of how to support heterogeneous virtualization technologies in other virtual computing environments, such as NC State's VCL.

Currently, support for VMware Server and VirtualBox is implemented and undergoing testing. Support for Xen, KVM, and Qemu is in the design or implementation phases. Other technologies will be supported on an as needed basis.

Experience Report: Our experience using virtualization to teach computer science via Storytelling Alice [11] to K-8 students suggests that slow software startup times inhibit the engagement and attention of the students.

The time between user invocation in a virtual computing environment and when Storytelling Alice is ready to use depends on four broad factors: network communication time, time to prepare a virtual machine, time to start the virtual machine, and time to start Storytelling Alice. In the results that follow, we do not measure the network latency as it can be highly variable and is outside our control. The latencies associated with the remaining operations were measured in seconds. The test platform is our deployed SERViCE environment, a desktop system with an AMD Athlon 64 X2 3800+ processor and 2 GB RAM running the SERViCE framework using VirtualBox.

Initial measurements on a virtual machine with a fresh installation of Windows XP, including the Java runtime environment and Storytelling Alice, show a latency of 175 seconds, of which 144 seconds is spent cloning the virtual machine from the master image. Additional experiments indicate that the time to clone the image is proportional to the size of the image. Thus, the best way to decrease the startup latency is to reduce the time it takes to clone the image. While a Windows installation can be stripped down somewhat, doing so is tedious and time consuming.

Another way in which to reduce the time to clone the virtual machine image is to configure VirtualBox to use a shared copy-on-write image. Copy-on-write does not copy the data. It only sets up some data structures in order to access the pre-existing data. The startup latency after configuring VirtualBox to utilize a copy-on-write image is 29 seconds, a *six-fold improvement* in performance.

To summarize, smaller images can be cloned faster and hence exhibit lower latency. However, copy-on-write eliminates much of the impact of size. Copy-on-write images provide the additional benefit of preventing "configuration creep". Thus, the use of copy-on-write is highly recommended.

2.2 Live CD

While extremely attractive, virtualization many not meet the needs of all the students in rural areas, primarily because it depends upon adequate network connectivity which are less likely than in metropolitan areas. One attractive alternative is the creation of *live CDs* which contain everything needed to run the software.

Figure 2 shows that a live CD is an installation of an operating system, support software and education applications on a read-only compact disk. Everything that is required to run education applications is included on the CD and configured to run without user intervention. Although expertise is required to create a live CD, only general computer skils are required to use one. Inserting the CD into the CD-ROM drive and rebooting the computer is all that is required to gain access to the packaged software.[1]

One of the advantages of using a live CD is the fact that it is a read-only medium and hence cannot be inadvertently misconfigured. This alone dramatically reduces the amount of support required to ensure that the software works as it should in a computer lab or home environment. It is also forgiving of the students' explorations and encourages them

[1]The computer BIOS boot order must also be set to try the CD-ROM before the hard drive. While permanently changing this setting is not overly difficult, it can be intimidating. Fortunately, many manufacturers, such as Dell, use BIOSs that can temporarily boot from a CD-ROM if a specific key is pressed during the boot sequence.

Figure 2: Anatomy of a live CD.

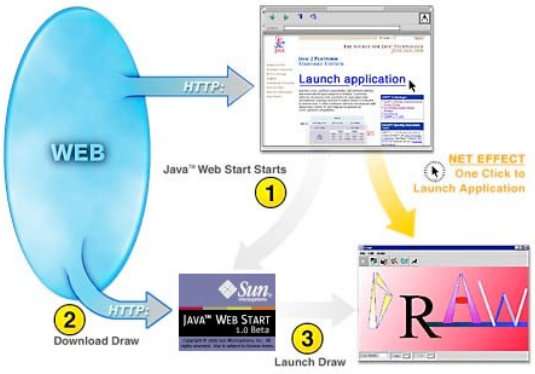

Figure 3: How Java Web Start Works [12].

to take chances which enhances learning. Any mistakes can be reverted by rebooting the computer.

Because it is read-only, however, a live CD does not allow a student to save partial work. There are at least three solutions to this problem. First, the live CD can be configured to mount the computer's hard drive so students can save their work as before. Alternatively, the live CD can be configured to allow their work to be saved to a network file share. The former is probably more appropriate for live CDs intended to be taken home while the later may be more appropriate for school use. Finally, the complete environment can be installed on one partition of a USB flash drive instead of a compact disk to create a live USB drive. A second partition can then be used to store the student's data.

Experience Report: We have created a live CD containing Storytelling Alice to teach elementary and middle-school students computer science concepts. We chose Storytelling Alice because the story board format engages the students in those age groups better than traditional approaches, especially for female students.

Storytelling Alice can be freely distributed but previously only ran on Windows which cannot. We have ported Storytelling Alice to Linux to avoid any issues of copyright infringement. We are in the process of folding those changes back into the mainline software distribution. We are also porting the changes made to create Storytelling Alice back onto the mainline Alice 2 distribution.

Concerning startup latency, the time it takes to boot a live CD is longer that the time it takes to boot a copy-on-write virtual machine. Depending on the speed of the computer and the CD-ROM drive, it can be as much as 1–3 minutes. The speed of the CD-ROM drive appears to be the limiting factor.

2.3 Java Network Launching Protocol

The Java Network Launching Protocol (JNLP), also known as "Java Web Start" (JWS), enables applications to be provisioned over the web, simply by clicking a link on a web site or even a desktop icon, as shown in Figure 3. JWS, defined via an XML schema, specifies *how* to launch JWS applications via a set of rules that define how to implement the launching mechanism.

We considered using JWS to make the software accessible since our application of interest, Storytelling Alice, is written in Java. It appears to be a natural fit. However, there are two problems.

First, Java Web Start requires a reasonably current Java runtime environment from Sun to operate successfully. We discovered that many of the machines likely to be available to students had old versions of the runtime that would not work for JWS. Installing any software can be intimidating, particularly for elementary and middle-school students or their teachers, because software installation often has unintended consequences.[2] Also, learning is greatly enhanced by self-guided discovery and hence the software needs to be available at home. Asking rural parents, many of whom are not computer literate, to download and install the Java runtime from Sun's web site is a high barrier to entry. The difficulty in installing the Java runtime environment is the primary motivation for creating a live CD with Java preinstalled.

Second, although Storytelling Alice is written in Java and hence JWS appears to be a natural fit, it requires the software to have a particular architecture which Storytelling Alice does possess. Modifying Storytelling Alice to use Java Web Start as a deployment mechanism requires a significant overhaul of the system, is going to take a while. It will also require significant effort to maintain since a major restructuring would cause it to be distinct from the mainline code base.

As a proof of concept, we rapidly prototyped such a system. The prototype is fragile but does serve as a proof-of-concept. It will require more effort to become a robust solution. However, we plan to further develop this approach as it easily makes software available via the click of web page link for those machines with access to the World Wide Web and a reasonably up to date Java runtime.

3. ISSUES TO CONSIDER

Increased accessibility of educational software to train future computer scientists requires more than the selection of deployment technologies. Three other important issues to consider are the proper licensing of the software to be deployed, the curriculum that will be used, and building a community to share expertise, implementations and curriculum.

[2]In many rural schools, system administration is performed by teachers. Even when a dedicated system administrator is available in a school district, self-contained software without unintended consequences is greatly appreciated.

3.1 Licensing

One of the challenges in making software for computer science more readily available is the issue of licensing. Software cannot legally be distributed by anyone other than the license holder unless authorized to do so. As a result, there are at least three types of licenses that must be kept in mind: (1) educational software licenses, (2) licenses for the support software, and (3) licenses for the operating system upon which all the software runs.

The need to obtain licenses for commercial software has a tendency to dampen its use to create self-contained live CDs or virtual machine images for wide spread deployment. Acquiring the legal right to distribute commercial software can become an expensive proposition. Alternatively, open-source software can be used for which permission has already been given in the license.

It is because of licensing issues that we have decided to exclusively use open-source software for the images we produce. The main educational software we use, Storytelling Alice, is freely available. It is written in Java and requires a Java runtime environment to execute. An open source version of the Java runtime environment, called OpenJDK, is available from Sun.

Storytelling Alice originally required the Windows operating system, which we cannot distribute. To solve this problem, we ported Storytelling Alice to Linux. We are in the process of making the changes available to the Alice developers for incorporation into the main distribution.

Finally, we have restricted our attention to open source virtualization technologies for the creation and execution of virtual machine images. Our preferred choices right now are Sun's VirtualBox or RedHat's KVM.

3.2 Curriculum

The primary focus of our teaching efforts has been on Storytelling Alice, created to motivate middle school students, particularly young women, to be interested in computer science [10]. It is an extension of the Alice educational software which greatly simplifies the steps that students must go through to create a 3-D "story," thereby making computer science more approachable to children at a younger age. Since we are focusing our teaching efforts on elementary- and middle school-age children, it is an appropriate choice. Experience teaching children as young as third grade validates the choice.

Although there are several books for teaching programming using Alice, for example [4], we are not aware of any materials for teaching programming with Storytelling Alice. We are in the process of developing such a curriculum with the aid of some exceptional undergraduate students.

We are also preparing to teach 3rd–5th grade students in ten elementary schools during the first quarter of 2010. The teaching opportunities are being organized under the auspices of the Computer Science Community Service (CS^2) club at Virginia Tech, which perform various computer service activities for the community. Several of our undergraduate assistants are officers or members.

Also ongoing is our effort to port Storytelling Alice back to pure Alice. Storytelling Alice branched off from Alice 2.0. Since then, bugs have been fixed in Alice and additional features have been developed. We are working with the main Alice 2.x developer to make sure our changes, both with Storytelling Alice and with the Linux port, will be included in the main distribution for the benefit of all.

3.3 Community

One of the by-products of the open source software movement is a renewed realization of the power of a group of like-minded individuals working for a common goal. Large businesses, such as IBM, have contributed to and leveraged the collective might of open-source volunteers. We envision a community surrounding the development and use of educational software which would bring together like-minded technologists and educators support discussion, development, and deployment of educational software for the benefit of students, teachers, and us all. Towards this end, we announce the Free Repository of Educational Software and Curriculum Archive (FRESCA) `http://service.cs.vt.edu/fresca`.

The goals for FRESCA are to serve as the gathering place for education professionals and technologists, to provide a catalog of available software pre-packaged as live CDs or virtual machine images, and to build a community for developing lesson plans and other educational materials needed to make effective use of the software.

It is envisioned that all of the software and lesson materials referenced from the FRESCA site will be freely available, probably under an open source license (or under a Creative Commons license in the case of lesson materials). However, it is also possible that versions of commercial software and materials that cannot be freely distributed may also be made available under a "free for academic use" license.

4. RELATED WORK

The SERViCE framework is most similar to the Virtual Computing Laboratory at North Carolina State University [1]. The source code for the VCL is open-source software and the first author is a registered developer on the project. A pilot deployment of the VCL software at Virginia Tech is also underway. However, the authors continue to extend the SERViCE framework as it is a much smaller, less complex code base from which to try various ideas that may or may not prove useful to the VCL project.

Cloud computing also leverages virtualization to enable economies of scale and dynamic reprovisioning of physical resources amortized across many organizations. Viewed in a certain way, VCL or SERViCE can be considered clouds specialized to education. While there are definitely some vocal proponents advocating cloud computing, even for education [2, 3], it is still to early to tell whether generic cloud services will adequately meet the specialized needs of academic organizations. Important services, e.g., integration with student records, are not provide by cloud vendors. And it is still an open question in the minds of many whether confidential information should even be processed in the cloud.

There are many sites hosting virtual machine images. The largest is VMWare's Virtual Appliance Marketplace which hosts images intended for its products. Many images are available at these sites. However, few of them are geared towards supporting education. It is due to the lack of a site focused specifically on education that we developed FRESCA.

5. CONCLUSION

While science and engineering jobs are projected to grow dramatically, especially jobs related to computing, enrollments in computer science and computer engineering pro-

grams have flattened or decreased. In addition, enrollments of women and minorities in the field continue to decrease. Since studies have shown that the middle-school years are very important for inspiring students to pursue careers in computing, we are developing ways to expose students to computer science fundamentals earlier by increasing accessibility to computer science educational software.

We have explored three different approaches for making it easier for teachers and students to gain access. With the rapid maturation of the technology, virtualization is an effective way to make software accessible. First, we have developed a virtual computing environment called SERViCE that has been used to teach computer science fundamentals to elementary students using the Storytelling Alice application. Students are excited about and wish to continue their explorations at home.

Second, the student's desire to take the software with them led to the development of a live CD with everything needed to run the software at home without the need to install or configure anything. Inserting the software disk into the CD-ROM drive and rebooting is all that is required to enter the learning environment. A very important side benefit to the use of a read-only deployment mechanism like a live CD is that students are able to freely experiment knowing that they can correct their mistakes no matter how bad by rebooting the machine. Initial results looks encouraging. We will have more data to report this spring after we teach 3rd–5th grade students at ten local elementary schools.

Third, we have explored the use of the Java Network Launch Protocol, also called Java Web Start, as a means of deploying the software. We have a prototype working but have not yet used it in teaching. The plan is to improve the prototype this summer and begin using it as a deployment mechanism in the fall.

We note that these delivery mechanisms are not exclusively for computer science software, but can be used to make software from any other disciplines more accessible to teachers and students.

Finally, we are in the process of developing a curriculum for Storytelling Alice for use in the teaching of classes this coming spring.

6. ACKNOWLEDGMENTS

We express our gratitude to Caitlin Kelleher for developing the Storytelling Alice environment, for providing invaluable feedback, and for her inspiring remarks at Kids Tech University 2008. We also express our gratitude to the Diversity Committee in Department of Computer Science at Virginia Tech for supporting this work.

We would like to thank Whitney Edmister of C-Tech^2 for providing the opportunity to reach out to 30 female high school juniors and seniors this past summer and get them excited about computer science.

Finally, we gratefully acknowledge the assistance of Adam Herr (Xen; FRESCA), David Mazary (port to of Storytelling Alice 2.0 Linux; FRESCA), Heshan Lin (mentor), Thomas Scogland (enabling infrastructure and logistics), and Gabriel Martinez (Storytelling Alice live CD). We also acknowledge the ongoing work of Michelle Datoc, Scott Fernandez, and Sahil Talwar in creating a Storytelling Alice K-12 curriculum.

7. REFERENCES

[1] S. Averitt, M. Bugaev, A. Peeler, H. Shaffer, E. Sills, S. Stein, J. Thompson, and M. Vouk. Virtual Computing Laboratory (VCL). In *Proc. 1st Int. Conf. on Virtual Computing Initiative (ICVCI-1)*, pages 1–16, Research Triangle Park, NC, May 7–8 2007. IBM Corp.

[2] T. Bittman. Cloud Computing and K-12 Education. `http://blogs.gartner.com/thomas_bittman/2008/11/26/cloud-computing-and-%k-12-education/`, 2009. Last checked 2009-10-13.

[3] K. Bunchuck. Cloud Computing for K-12 Schools. `http://bunchuck.blogspot.com/2008/10/cloud-computing-for-k-12-schools.h%tml`, 2009. Last checked 2009-10-13.

[4] W. Dann, S. Cooper, and R. Pausch. *Learning to Program with Alice*. Prentice Hall, 2005. ISBN-10: 0131872893; ISBN-13: 978-0131872899.

[5] Education Week. Technology Counts 2008 — STEM: The Push to Improve Science, Technology, Engineering, and Mathematics, 2008. Online suppliment containing statistics for Virginia available at `http://www.edweek.org/ew/articles/2008/03/27/30dsr.h27.html`.

[6] K. A. Frenkel. Women and Computing. *Communications of the ACM*, 33(11):34–46, Nov 1990.

[7] M. K. Gardner and W. Feng. Towards a Virtual Ecosystem for K-8 Education. In *Proc. 2nd International Conference on Virtual Computing Initiative (ICVCI-2)*, Research Triangle Park, NC, May 15–16 2008. IBM Corp.

[8] D. E. Hecker. Occupational Employment Projections To 2014. *Monthly Labor Review*, 128(11):70–101, Nov 2005.

[9] Higher Education Research Institute (HERI) at the University of California at Los Angeles (UCLA). Low Interest in CS and CE Among Incoming Freshmen, Feb 2007.

[10] C. Kelleher. *Motivating Programming: Using Storytelling to make Computer Programming Attractive to Middle School Girls*. PhD thesis, Carnegie Mellon University, 2006. Technical Report CMU-CS-06-171.

[11] C. Kelleher and R. Pausch. Using Storytelling to Motivate Programming. *Commun. ACM*, 50(7):58–64, 2007.

[12] Sun Microsystems, Inc. Technical Articles and Tips: Java Web Start Architecture. `http://java.sun.com/javase/technologies/desktop/javawebstart/architectu%re.html`. Last checked 2010-01-11.

[13] A. Tucker, F. Deek, J. Jones, D. McCowan, C. Stephenson, and A. Verno, editors. *A Model Curriculum for K-12 Computer Science*. Final Report of the ACM K-12 Task Force Curriculum Committee, Oct 2003.