Reducing Instructional Barriers Through Software Virtualization

I.W. Wait
Marshall University, Huntington, West Virginia, U.S.A

Abstract— **Engineering educators sometimes avoid incorporating computerized design tools into courses that might otherwise benefit from them due to the complexity of supporting software applications. Between the challenges of distributing installation files, helping students to troubleshoot hardware incompatibilities, ensuring licensing compliance, and coordinating software installation on shared lab computers, the costs of software incorporation are sometimes seen to outweigh the potential benefits. One increasingly accessible solution to many of these issues is the virtualization of software, wherein software is installed and maintained on a centralized server remotely accessed by client machines via the internet. By eliminating many of the challenges associated with localized software, virtualization can reduce both the real and the perceived costs of software integration, while preserving many instructional benefits that can arise through software incorporation.**

A case study is described to illustrate methods that can be implemented to virtualize software. This case study also highlights some of the pedagogical benefits resulting from a brief, casual exposure to engineering software that would not ordinarily justify the effort of making locally-installed software available to students. Software virtualization was utilized in a junior-level fluid mechanics course for civil engineering students. Feedback from students indicates a favorable response to virtualization as a means of software accessibility, and a favorable view of the small learning activities that were enabled by this access.

Index Terms — **Software, Virtualization.**

Introduction

In many engineering programs, software instruction is segregated into a few key courses where learning how to use the program under consideration is the primary objective of the course. Examples include Computer Aided Drafting (CAD) courses, Geographic Information Systems (GIS) courses, and programming courses in various computer languages. Except for these major software-themed classes, computer programs that may be related to the subject being taught are often not incorporated into the course curriculum, aside from a few instances of common, unspecialized utilities (e.g., Microsoft Excel, Matlab, etc.) Although many instructors are already aware of the potential benefits of exposing students to design software, some hesitate to actually implement software instruction in their courses because of access barriers, perceived inconvenience, and technical challenges.

Oftentimes, non-software-focus classes do not have dedicated desktop computers available in the classroom, which limits the potential for installation of software onto university-owned machines. In other cases, where university-owned computers are available, IT departments may have lengthy lead times or a complex approvals process for instructors to request software installation. Some instructors may overcome these barriers by choosing to make software available to students for installation on their personally-owned computers. However, distributing large software installation files can be a challenge. Likewise, the process of walking new users through potentially cumbersome and confusing license activation procedures is often enough to dissuade instructors from attempting software incorporation unless the software program is a major part of the course.

Complicating matters further is that personally-owned computer hardware is often non-standardized – students own a variety of different machines, and run a variety of different operating systems. The prospect of providing technical support to students – each who has installed software on a different computer, and may be experience a different technical problem – can be enough to dissuade an instructor from integrating software into a course.

Fortunately, the rise of cloud computing and the ease of access to server-hosted, virtualized installations of software can eliminate many of the implementation barriers that might otherwise cause instructors to feel that the cost of teaching software outweighs the learning benefits. A variety of commercial and open-source virtualization options and providers (e.g., Citrix Xen, VMware vSphere,, Microsoft Hyper-V, and others) enable any program to be managed and operated at a fraction of the inconvenience to the end-user.

Within academic instruction, virtualization has been utilized extensively in computer science settings for a variety of purposes, including unprotected network security testing for which a 'real' hardware environment would represent a security risk [1], and to simulate and investigate the performance implications of different router network configurations without actually needing network hardware [2]. Likewise, virtualization has been utilized for making multiple operating systems available to students within a single physical machine [3].

A virtualized software package can be configured to appear and function the same as if the application were instead operating in a stand-alone mode on a lab or personal computer. Additionally, since virtualization clients are often based on cross-platform tools that exist for multiple operating systems (e.g., Mac, Linux), access is enabled for students using otherwise unsupported operating systems. Engineering packages that operate only in a Windows environment, for example, can become accessible to Mac and Linux users if those users connect to a Windows-based server hosting virtualization. Rather than running the desired program on a local computer, virtualization utilizes the local machine to view and control the program that is being run remotely.

In the case-study described herein, students utilized a server-hosted, locally-controlled hydraulic design package (i.e., Bentley WaterGEMS) to conduct an in-class demonstration and out-of-class assignment to support their learning of three fluid mechanics course topics: the hydrostatic equation, energy loss in pipes, and pipe network optimization. It is uncommon to introduce a sophisticated hydraulic design package to students in a first course in fluid mechanics. Such a course has content that predominantly leans towards 'core knowledge' topics rather than application and/or design. However, targeted software exposure can support teaching of fundamental concepts by enabling students to rapidly solve many different problems and through this iteration develop a 'feel' for how equations behave. Students have a sense of accomplishment when learning how to operate industry-standard design software, and also come to develop a broader vision for the purpose of the fundamental principles that are being learned, and the applied direction for where course topics lead.

A. Traditional Challenges of Software Implementation

Although many instructors recognize the value of incorporating software instruction and learning activities into the courses they teach, there are a variety of technical and logistical hurdles that typically stand in the way. These hurdles introduce inconvenience to the instructor and to students, and serve as powerful dis-incentives to attempting to incorporate software training into courses. Among the challenges are:

Software installation must be coordinated with IT managers – In some cases, instructors do not have administrative rights to themselves install programs onto lab or classroom computers. Thus it is required to coordinate installation with IT managers, who may not be able to install new software onto machines mid-semester. In such instances, the requirement to request the addition of software far in advance of when it will be used in class may constrain the opportunity for new learning activities to be run.

Computer lab crowding – Since computer labs are generally already used for courses that do traditionally contain a software component (e.g., CAD, senior-level design courses, etc.), it may be that there is limited extra capacity in computer labs for new assignments and utilizations to be introduced.

Course is not taught in computer lab – For many core knowledge type courses, where fundamental concepts are of primary concern and software is not typically utilized, the course will often be taught in a classroom that does not include access to computers. Thus it is difficult to provide computer access to students for just one or two class periods when a brief software demonstration is desired.

Heterogeneity among student-owned computers – Among computers that run MS Windows, students generally own a wide variety of different computer types, manufactured by many different manufacturers and running a variety of different operating systems (including 32-bit vs. 64-bit). These differences inevitably lead to errors that must be resolved when students install software onto their local machines. Being thrust into the role of 'software installation technical support' can be a powerful dis-incentive against incorporating a software component into a course for an instructor with limited time or ability to troubleshoot such issues.

Licensing complexities – In some cases where a university has paid for a license to a particular program, that license may not allow students to install versions of the program onto their personally owned computers. In such cases, and where other factors limit availability to software.

Cross-platform non-operability – With the increasing popularity of non-Windows operating systems (e.g., Linux, Mac), more students own computers for which the operating system is not compatible with the software to be taught.

These challenges can be mitigated, and in some cases eliminated entirely through virtualization of software applications. For purposes of comparison, whereas introducing a software application to students previously required the instructor to commit approximately 10 hours per semester (much of which was helping students debug faulty software installations and sort through licensing issues), by implementing software virtualization, the time requirement was reduced to approximately 2 hours, including prep time before class demonstrations. In subsequent semesters, the time savings would likely be even more favorable for virtualization.

B. Benefits of Software Virtualization and Simplified Program Access for Students

"Virtualization" of software, such that it is installed and operated on a remote server to which students connect when they wish to run the program in question, can reduce or eliminate each of the difficulties identified above. During virtualization, the client computer merely acts as a 'window' to the remotely-installed and run software program, such that a user is able to see the software-generated screen, interact with the program, and issue commands to it, but since the program is actually hosted remotely, this means that it is the remote server that must have the hardware and software required to run the program. This virtualization technique simplifies academic implementation of software in a number of ways.

By making it possible for students to operate software without it actually needing to be installed on a computer, instructors no longer need to coordinate installation of software with the IT managers who control classroom and computer lab machines. This introduces flexibility to decide on which programs to utilize during the semester itself, where previously this may not have been possible.

Likewise, virtualization can overcome the problems associated with not enough computers in a classroom or computer lab, since most university students already own a PC and through virtualization can easily operate the needed program without having to install it. Since software is run without being installed, the challenges and irritations associated with installation, licensing, and cross-platform non-operability are avoided. Instead, students must only configure the small client applet that allows them to connect to the remote server, and are then

able to bring up the program as needed, even if their machine wouldn't otherwise have adequate RAM, a compatible operating system, or any of the other requirements that sometimes stand in the way of their installing software programs locally.

C. Virtualization Case Study – Student Experience

While running software virtually is significantly less cumbersome than installing it locally, there are still some initially unfamiliar steps that must be followed for use. In the fluid mechanics course described herein, the following was the workflow that students had to follow in order to utilize the Bentley Systems, Inc. hydraulic design software "WaterGEMS."

Download and install virtualization client. Students clicked on a link that was provided by the instructor. This link is what is used to trigger the virtualization of the WaterGEMS program any time the student wished to access it (whether on campus or off). If the requisite client software (in this case "Citrix Receiver") is not installed on the student machine, then the provided link automatically provides an intuitive and easy way for the user to install the client.

Since administrator rights were not required to install the client, this meant that students could access to the virtualized WaterGEMS even on computers where they have limited (i.e., non-administrator) accounts. This is a significant departure from the rights generally required to install local software.

Trigger software virtualization. Once the client software was installed and running in the background as evidenced by an icon displayed in the computer's system tray, students once again clicked on the instructor-provided access link to trigger virtualization of WaterGEMS. Upon allowing access in a security warning dialog box, the virtualized version of the program software opens, along with any pre-configured program files that the instructor wishes to have loaded into the program upon initiation of the virtualization link. Likewise supporting programs can be automatically called to load, and in this case Adobe Reader was used to display an electronic copy of the assignment on screen.

The student's view of the virtualized WaterGEMS program window was identical to the program windows available when the software is installed locally. All of the same program functionality was available, including all program menus and buttons, the option to save program files onto the students' local machine, and the ability to print to locally-connected printers. Thus the student experience when using virtualized WaterGEMS is nearly identical to using a locally-installed version of the program, and in fact users are able to switch back and forth between the two with files saved from either environment.

A brief (~3 minute) screen-capture video illustrating the software process utilized and one of the learning activities implemented in this case study was provided to students after the in-class demonstration. This video is available for viewing at:

http://www.youtube.com/watch?v=g_CzboAu63A

D. Server Specifications and Performance

For purposes of the pilot test that was conducted, the server hosting the WaterGEMS software was a cloud-based server managed by Bentley. However, university IT departments often have existing experience with virtualization servers, and could potentially be called upon to host at the university-level the applications that are to be used for instructional purposes.

The technical specifications of the remote server hosting the software (Intel Xeon @ 2.45 GHz with 7.5 GB main memory running 64-bit Windows Server 2008 R2 Datacenter), coupled with the requirements of the program(s) being virtualized and the virtualization software used, can limit the number of simultaneous users that can be accommodated via remote connection from a single server. In the case of the single server that was utilized to host virtualization for this course (which was not designed for many simultaneous users, but rather to offer student's with at-home access), approximately 15-20 users could be connected at any single time. Additional simultaneous connections above this limit resulted in users not being able to connect to the server and/or decreased performance. Depending on the number of servers used, CPU and RAM configuration of each, the virtualization software utilized, and the specification of program or programs being virtualized, the number of users that can be accommodated may be significantly more than or fewer than those mentioned above.

The initial computational load associated with making a connection to the server and beginning the virtualized programs (i.e., both WaterGEMS and Adobe Reader for each user) was significantly higher than the server resources utilized once the start-up process was finished and the program was in operation. Thus, having an entire classroom of students connect to the server at a single time, such as when an instructor begins an in-class demonstration, could lead to reduced performance and/or system difficulties if the virtualization servers are not designed to accommodate simultaneous use.

Network bandwidth is another factor that may limit feasibility of software virtualization. The virtualization software that was utilized is known to be extremely efficient with respect to connection bandwidth requirements; tests have shown that as little as 200 Kbps per user bandwidth is required, and network latency as high as the 100's of milliseconds is acceptable to maintain software interactivity. Only one student reported concerns that may have been related to insufficient bandwidth and its potential for performance degradation.

E. Student Response and Feedback

A brief survey was administered to students in-class, following their submission of the assignment, in order to characterize student opinion about the learning experience and to assess whether any technical issues limited their ability to successfully complete the activity. A total of 29 students (out of 31 enrolled in the course) completed the survey.

Table 1 – Results of student feedback on learning activity survey.

How difficult was it to learn to use WaterGEMS?	**N = 29**
3 – VERY difficult. This software should not be taught to students in my position.	0
2 – Somewhat challenging, but not too difficult for students in my position.	18
1 – It was easy.	11
In the context of better understanding the hydrostatic equation, what is your opinion about the WaterGEMS assignment that you completed? From an educational standpoint…	**N = 29**
5 - It was a very useful learning activity. It had significant positive impact on my understanding.	10
4 - It was a somewhat useful learning activity.	15
3 – Neutral	4
2 - It was not a useful learning activity.	0
1 - It was a very useless learning activity. It actually made me understand hydrostatics LESS.	0
Did you view the screen-recording video?	**N = 29**
Yes	7
No	22
To complete the homework assignment, did you use the Virtualized version of WaterGEMS, a lab computer with WaterGEMS, or did you install WaterGEMS on your home computer? (Select all that apply.)	**N = 29**
Virtualized	18
Lab computer	13
Installed on own machine	2
If you used the Virtualized version of WaterGEMS, did you encounter any problems?	**N=18**
Yes	5
No	13
How interested are you to see more of your academically required engineering software and assignments provided online (i.e., using a remote delivery 'Virtualization' system, rather than having to access software in the lab)?	**N=29**
5 - Very Interested	14
4 - Somewhat interested	11
3 - Neutral	4
2 - Not interested	0
1 - Very uninterested - would prefer that it not happen at all	0

As summarized in Table 1, 18 of 29 students (i.e., 62%) completed some or all of the assignment utilizing the virtualized (i.e., remotely-operated, server-hosted) version of the program. This is significant because students also had ready access to a local installation of the program in a computer lab adjacent to the classroom where the course is taught, such that they could have completed the homework assignment on campus without the need to configure the virtualization applet on their personal computer. However, the benefit of being able to operate the program on their own PC, without having to actually install it, was enough for 62% of students to choose this option. This highlights the ease of access to a virtualized software application, even for students who have not previously used it. Of the students who relied on the virtualized version of the software, 16 completed the assignment entirely using the virtualized version of WaterGEMS, and 2 used both the virtualized version of the program and a lab-installation of the software. 13 of 29 students (45%) completed some or all of the assignment on lab computers, and 2 students installed the full version of the program on their own computers.

Of primary importance when considering student feedback is that all students characterized the software to be "easy" or "not too difficult", with none selecting the option that the software was "VERY difficult." The hydraulic design software utilized (i.e., WaterGEMS) is actually quite sophisticated, and yet simple learning activities were designed by the instructor and conducted by the students, needing only a portion of an existing class lecture. The perception among students that the software was not too challenging to use, in spite of their being in only an introductory fluid mechanics course, highlights the potential low-hanging-fruit of integrating a small, simple introductory exercise with which to initiate student use of new software. The student feedback may also reflect the value of a live in-class demonstration where students first see the unfamiliar activity demonstrated, then they complete it themselves, and then they observe a classmate completing the steps a second time.

86% of students responded that the software homework assignment that followed the in-class demonstration was "very useful" or "somewhat useful" as a learning activity that supported their understanding of the hydrostatic equation. Thus, even though the WaterGEMS program's functionality goes well beyond calculating hydrostatic pressures, students felt that the assignment – which required them to investigate the effect of manipulating several different parameters within the hydrostatic equation – helped them to better learn the underlying principles.

Of the five students who encountered problems when using the virtualized version of the software, two reported that they were unable to connect to the software after loading the virtualization applet on their computer, one encountered unexpected program termination (possibly due to their internet connection being dropped), one student reported that the program 'didn't work', and one student reported a 'laggy' (i.e., high latency) connection when running the program virtually. These difficulties could probably be overcome over time, and in any case are a small fraction of the technical complaints and challenges experienced when local installation of the software has been attempted in previous semesters.

Conclusion

The learning activities described in this paper were undertaken as a pilot test of two principles, the first of which is enabled by the second: (1) a minimalist introduction of hydraulic design software in the early stages of a student's learning of fundamental principles, and (2) utilization of virtualization systems to simplify the steps necessary to incorporate software instruction into a course. On both accounts, this pilot test was encouraging – students reported a favorable experience using the hydraulic design software, had markedly fewer technical

issues compared to previous instances where local installation of the software was attempted, and all students indicated an interest in receiving more software through virtualization. Likewise, the instructor resources required for virtualized-delivery of software were less intensive than when software is installed locally, and thus continued future implementation of this methodology is anticipated.

Virtualization represents an approach that may significantly reduce the barriers that instructors face when they wish to make software available to students. This is particularly important for cases where the effort required to make the software available has previously outweighed the perceived benefits of doing so.

ACKNOWLEDGMENT

Brad Workman and Mike McSween are acknowledged for their valuable assistance during the pilot study described in this paper.

REFERENCES

[1] Jones, J.M. and Chou, T. "Work-in-Progress: Creating an Intrusion Detection Experimental Environment Using Cloud-Based Virtualization Technology." Proceedings of the 2012 ASEE Annual Conference and Exposition, San Antonio, Texas, June 10-13, 2012.

[2] Li, T., Thain, W.E., and Fallon, T.. "On the use of virtualization for router network simulation." Proceedings of the 2010 ASEE Annual Conference and Exposition, Louisville, Kentucky, June 20-23, 2010.

[3] Bailey, M. and Ekstrom, J. "Teaching web development with OS-virtualization." Proceedings of the 2009 ASEE Annual Conference and Exposition, Austin, Texas, June 14-17, 2009.

AUTHOR

I. W. Wait is an Associate Professor at Marshall University, Huntington, WV 25545 USA (email: wait@marshall.edu).

This work was supported in part by Bentley Systems, Inc, who hosted the software virtualization and provided technical support.

Submitted, August, 19, 2013.