AN ABSTRACT OF THE THESIS OF

Ajinkya Patil for the degree of Master of Science in Computer Science presented on August 18, 2016.

Title: Enhancing Instructors' Teaching Experience through Virtualization.

Abstract approved:

_____

Carlos Jensen

In the current education environment, many instructors make use of some type of software, such as Visual Studio or a software library like OpenGL, in the classroom. Incorrect setup and configuration on an individual's own system is a common problem when using these software tools. This thesis explores the difficulty that these setup and configuration problems create for the student learning experience as well as the instructor teaching experience. We interviewed ten instructors and four students as part of a qualitative study to gather data about the severity of this problem in preventing student learning and effective delivery of material. The participants were chosen from Computer Science, Mechanical Engineering, Statistics, and Robotics classes. The main criterion for the class selection was whether that class' usage of software came with the potential of running into configuration issues. The study shows that almost all classes did suffer from configuration issues in software tools; these issues distracted the students from actual coursework, forcing them to expend their time and energy on resolving these issues instead. It was also found that instructors and students often resorted to a "use whatever works" or a "trial and error" strategy for dealing with the issues encountered, which did not guarantee the issues' resolution. To help instructors and students resolve a substantial portion of these setup and configuration issues with software tools, we propose the design of a virtualized system which leverages an existing cloud tool OpenStack. This system modifies the services provided by OpenStack to enable easy access to preconfigured Virtual Machines which help to resolve these issues with setup and configuration.

This thesis, along with presenting the result of the study, outlines this system framework. Furthermore, it discusses the system in the context of the data gathered and the types of problems that framework could effectively handle. The lessons learned from the study and from our system design can also help inform future solutions and systems.

Enhancing Instructors' Teaching Experience through Virtualization

by
Ajinkya Patil

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented August 18 2016
Commencement June 2017

Master of Science thesis of Ajinkya Patil presented on August 18 2016

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Ajinkya Patil, Author

ACKNOWLEDGEMENTS

The author expresses sincere appreciation to my advisor Dr. Carlos Jensen for his immeasurable help and direction. To all the instructors and students who took time out of their busy schedules to answer my questions in detail. To my committee members and to all my friends, colleagues and family, who directly or indirectly helped me finish this study and get me to where I am now.

TABLE OF CONTENTS

Page

LIST OF TABLES

1. Introduction

Software configurations or settings tend to vary a great deal from system to system. The software/tool that necessarily always seems to work on one machine with one version of an operating system, may not work at all on the same machine with a slightly different operating system version.

This introduces a lot of challenges for the instructor when teaching a class with a special emphasis or reliance on certain tool(s) because they cannot be certain it will work the same for every student. Not only can they end up spending a lot of time dealing with each students individual issues, but also may end up compromising and settling for a lesser tool simply because it works. This problem can become even worse in online courses where in-person and fast help is not available.

Much of the same challenges with these software tools apply to students where they can end up spending all their time and effort on making the tool work properly on their machine.

The current workaround employed by instructors and students seems to be more along the lines of "use whatever works" than a real concrete solution. This is not a good strategy to employ in the long run and can negatively impact students and instructors overall classroom experience.

Our research goal was thus based on figuring out how to help instructors teach more effectively, by eliminating access or configuration issues with software. In other words, how to go about helping instructors teach more hands-on and engaging courses while wasting less time and reducing risks associated with technology access. Our hypothesis was that these technological hurdles, or the fear of such hurdles, limit the ambition and hands-on nature of instructors. By tackling or controlling these hurdles for the instructor they will be freed in some regard to choose the ideal teaching methods they wished for. This also leads to richer student experiences.

As a model solution for just this purpose of freeing up instructors we designed a system employing OpenStack which is elaborated upon to some length in the discussion section. We found that OpenStack since then would be insufficient to help us deal with some system requirements. The incomplete model solution still holds value because lessons can be learned from its design and the problems it could easily have tackled. These problems or requirements were gathered from the study and from our own assumptions about the features the system should have.

In the next few sections, we present the data we gathered and analyzed with solutions and discussions in the context of our system framework, following up with the solution/system we wanted to design for the instructors for dealing with these setup and configuration problems.

2. Literature Review / Related works

The system design was made from the ground up but the idea of helping instructors in an online campus has been around for some time now.

Massive open online courses or MOOCs as they are called in short have been gaining massive popularity over the years. In essence, a MOOC is a system of online courses much like our Ecampus courses, which call for participation in a traditional classroom like environment, but over the web or online. The highlights of MOOC are video courseware, and assignments being offered entirely online with students interacting with each other through online discussion boards and chatting mechanisms. Some MOOCs adopt a blend together approach of traditional face to face teaching with online lectures which seemingly performed very positively in affecting student grades.

For being offered over the web and holding its videos, assignments, etc. while also managing the massive students that could be enrolled at one point, it needs a platform structure like Canvas' "instructure" to bring everything together.

MOOCs can garner a large student population so there are two approaches that are used for such large-scale feedback and interaction. First is peer-reviews and group collaborations which is used by a type of MOOC called cMOOCs or connectivist MOOCs where a connection/collaboration philosophy is employed. Other approach is the automation of feedback in online assessments, where machine grading and feedback is also being phased in. This strategy is employed by the xMOOCs or extended MOOCs which basically means the course is in addition to something (adapted from Wikipedia).

There are different MOOC providers which have gathered over the time including popular for-profit ones like Coursera, Udemy, Udacity and non-profit ones like edX and Khan Academy. They each have their own strengths, weaknesses in how they choose to interact with the students. But they all host a platform which was either

developed or adapted from open source tools where they handle students and which can frequently cause technological problems.

We focus on the technology support and challenges that are currently being faced in that aspect here, since MOOC involves a great deal of other debate and discussion over its different facets.

One of the big challenges they face are the completion rates, which are typically lower than 10%, with a big drop in the first few weeks [2]. When asked about the reason for dropping out few typical responses that stand out are the "clunky technology" and "lack of a proper introduction to course technology and format" [3]. The MOOC guide [4] states five possible challenges that the MOOCs face, and the second one on the list is "digital literacy" for using the online materials. In such a circumstance if the tools are even more inaccessible then the drop rates could not ever hope to do better.

Here are some solutions that were proposed for dealing with these technological issues. All of these propositions try to arrive at the solution from different angles. Another thing to keep in mind is that these solutions do not necessarily try to tackle only the technological issues but still feel the importance of considering it.
[5] suggests to take the lessons offered by MOOC and apply them towards the development of a more strategic approach to online learning. Among these include the exploration of "technology options" where new platforms and services with different functions, terms and conditions for experimenting with the development of MOOCs and open online provision in institutions be taken into account which also allows for suitable innovative experimentation.

[6] concluded that for "networked learning to be successful people need to have the ability to direct their own learning and to have a *level of critical literacies* that will ensure they are confident at negotiating the Web in order to engage, participate, and

get involved with learning activities. *People also have to be confident and competent in using the different tools in order to engage in meaningful interaction*."

Another innovative suggestion tried by this study, [7], explored the use of blogs and forums *"for encouraging learners to exercise autonomy in creating their own learning networks".* They provided for rapid interaction and more personal reflection it was found. And additionally it pointed to a maturing of e-learners where once they realized the affordances and responsibility of such a setup, they did step up quite well to the role, and *"developed the platform in innovative and nuanced way, with little regard to the capabilities required or limitation of the particular media."*

Among other related works, the coding technique used in the study for data analysis, is a popular technique in Qualitative studies. [1] was used as inspiration for the codes and provided the initial technique.

3. Methodology

For the purposes of reducing software configuration issues and designing a system that would help in that regard, we made some initial general assumptions about the system and what it would entail. But to design a system/solution that would be effective and actually usable to the instructors and the students, we gathered data from them through interviews. This enabled us to find new data about problems that the students and instructors were actually facing and served also to reconfirm and adjust our initial assumptions.

We targeted specific courses from Computer Science and other departments – Robotics, Statistics and Mechanical Engineering that made use of some software or tools in class. We decided on other departments since it would help to adapt the system/solution to a larger audience and help us notice any differences between the teaching methods and needs, if any. We thoroughly analyzed all of the courses that were being offered for the term from the catalog and "classes.engr.oregonstate" web space to identify potential classes and instructors.

We shortlisted the classes based on a criteria we designed for our solution. This is elaborated upon more in later sections. We ended up with 10 classes and 4 students from those classes.

We next conducted a half-hour to hour long interview with each instructor and student. It was recorded and notes were also taken for easy analysis later on.
The interview script we used for data collection follows next, with reasoning behind each question, pertaining to what data we hoped to gather from the question and what we actually did gather.

Interview Script:

I. Instructor

Table 3.1 – Methodology – Interview

Instructor interview script

Questions	Reasoning
1. What is your favorite thing about teaching <course X>?	The first few questions like this are usually asked as part of an interview strategy where we get the subject to loosen up a little bit and get comfortable talking. We did not hope to collect anything useful from this question but we did end up getting some interesting answers which affords us some insights.
2. What is the most challenging thing about teaching <course X>?	The same is true of this question. Insights aside, the data gathered from these questions also allows us to cater our system so that the challenges/likes mentioned do get taken into account. Moreover it also allows for development of general solutions which can be applied or considered for the future.
3. What are the key technical learning objectives of this class?	This question was meant to get the instructor to mention in their own words what the most important technical learning objectives were. It was aimed to get an idea of what the instructor thought were the key things that they wished to impart from their class. The instructors did not have access to their official "learning objectives" mentioned on their course website which was the intention. We also wanted to focus on the "technical" aspect of the objectives. The data from this can be used to assess whether the instructor met their ideal objectives from the actual ones and how to go about reaching that ideal point from

Table 3.1 – Methodology – Interview (Continued)

	a solution.
4. What technical skills/technologies do your students need to learn or master as part of this course?	This question was meant to extract the important skills that were required in the class to do well. It would help to gauge what skills were deemed important and should be considered for any future solution.
5. What technical skills/technologies do you wish students could learn or master as part of this course?	This question on the flip side was meant to extract any skills or technologies that were desired to be taught in the class by the instructor. This would give an indication of any skills that the instructor was not able to bring in because of configuration or other issues. And what the future solution should probably aim to provide.
6. What tools or software (if any) do you or the students use in the class?	For getting an idea of what tools/software are required in the class.
7. What other tools or software you wish you could use, if any?	For getting an idea of what tools/software are desired by the instructor in the class.
7. (a) What are the barriers to using those tools/software?	For eliciting the problems that the instructors encountered for bringing in desired tools/software.
8. What are the technological barriers (configuring, installing tools, etc.) students in <class X> struggle with?	The main research question. The problems mainly of the technical nature with tools, software, etc., that the students faced in the class.

Table 3.1 – Methodology – Interview (Continued)

8. (a) What have you tried to do to overcome these barriers?	What things were already tried to overcome these problems.
9. How do the tools and technology you currently use affect the way you teach or the structure of <class X>?	For measuring how much of an impact or reliance, do the tools/technology have on the class.
10. If you were not restricted by the available technology, how would you teach the course differently?	If there were still any ideal way of teaching the class the instructor was missing out on. Although this was asked in relation to software configurations, the answers can be applied generally.
11. If you had a central VM (Virtual Machine) with preinstalled software tailored to your class available for students to use, how would it change your course structure and assignments?	System-specific question. For measuring how much of a relief a system like ours with access to a Virtual Machine would bring about for the course. This also gave an indication for the usefulness of a system like this.

II. Student

These questions and reasoning are more or less in line with those of the instructor. They differ in the wordings and the answers were largely dependent, it was observed, on prior student knowledge.

Table 3.2 – Methodology – Interview

Student interview script

Questions	Reasoning
1. What is/was your favorite thing about taking <course X>?	For measuring what the students liked about the class.
2. What is/was the most challenging thing about <course X>?	For measuring the thing the students found most challenging about the class.
3. What are some of the key technical learning objectives of this class?	The technical objectives from the students' perspective.
4. What technical skills/technologies did you need to learn or master as part of this course?	The required skills/technologies that the students had to learn (or apply) for the class.
5. What technical skills/technologies did you wish you could learn or master as part of this course?	The skills/technologies that the students felt would be better to be included in the class, in addition or separately.
6. What tools or software (if any) did you use in the class?	The tools/software that were required in the class.
7. What other tools or software you wish you could use, if any?	The tools/software that the students wished were used in the class.

Table 3.2 – Methodology – Interview (Continued)

a. What would you say are the barriers to using those tools/software?	According to the students, what were the barriers to bringing in those tools.
8. What are the technological barriers (configuring, installing tools, etc.) did you struggle with in <class X>?	The problems with tools/software that the students faced in the class.
a. What did you try, to overcome these barriers?	The workarounds the students employed for dealing with these problems.
9. How do you think the tools and technology that are currently used in <class X> affect the way you are taught or the structure of <class X>?	According to the students, what impact the tools used in class have on their teaching.
10. If we were not restricted by the available technology, how would you want the course to be taught differently?	The ideal way the students would like the course to be taught.
11. If you had a central VM with preinstalled software tailored to your class available for students to use, what changes would you want to see to the course structure and assignments?	Once again for measuring how useful a system with Virtual Machines would be in class to the students, according to the students. Also the changes they would like in such a scenario are also gauged.

Next step was transcribing the interview data. After that was finished, we proceeded to analyze it. We used the technique of assigning codes/tags to common themes termed “open coding”, that came to be prevalent in the transcript. This helped us to analyze effectively and group together relevant sections of important texts in the transcript for further insights and analysis.

What follows next are the tags/codes we used along with their descriptions.

I. Instructor

Table 3.3 – Methodology – Codes

Instructor code descriptions

Sr. No.	*Code*	*Description*
1.	Teaching likes	Teacher's favorite thing about class
2.	Teaching challenges	Challenges in the current way the class is taught, speaking overall about the class as a whole, not specific to any tools or skills
3.	Learning objectives	Instructor defined class-learning objectives (technical/general). Overlaps completely with skills.
4.	Skills	Any skills/technologies inferred or otherwise mentioned by the instructor, necessary or desired for the class. *Any text or speak that gives an indication of the skills that the instructor has required, will require or has desired for the class, which can be inferred.*
5.	Tools	Any mention of tools or relative connection to tools inferred or otherwise by the instructor. *Any text or speak that gives an indication of the tools that the instructor uses, has used or want to use, may not have a tool mentioned explicitly.*
6.	Barriers	The barriers for bringing in stated desired tools and technological barriers (configuration, installing, etc., with current tools) in the class
7.	Workarounds	The workarounds as stated for technological barriers
8.	Tool Impact	Impact of the tools / technology on the class, whether positive or negative, or a potential impact in the future
9.	Class changes	Instructor defined changes to their classes, if no restriction on technology, and with access to VMs. Which may be yes, no or neutral.
10.	Infrastructure changes	Changes to classes explicitly and inferentially gleaned from direct instructor responses and grouped into arguments for/against this project, project requirements, etc.

II. Students

Table 3.4 – Methodology – Codes

Student code descriptions

Sr. No.	*Code*	*Description*
1.	Student likes	Student's favorite thing about class
2.	Student challenges	Challenges students faced in the class
3.	Learning objectives	Student defined class-learning objectives (technical/general)
4.	Skills	Skills as stated necessary or desired for the class by the student, inferentially or otherwise
5.	Tools	Tools as stated necessary or desired or actually used for the class by the student, inferentially or otherwise
6.	Barriers	The barriers for bringing in stated desired tools and technological barriers (configuration, installing, etc., with current tools) in the class
7.	Workarounds	The workarounds as stated for technological barriers
8.	Tool Impact	Student defined impact of the tools / technology on the class
9.	Class changes	Student defined changes to their classes, if no restriction on technology, and with VM availability
10.	Infrastructure changes	Changes to classes explicitly and inferentially gleaned from direct student responses and grouped into arguments for/against this project, project requirements, etc. This is basically an extra code we introduced for help gathering requirements for our system.

As can be seen from the codes and their descriptions, they follow quite naturally from the questions and help group thematic objects easily. This helps to extract information quite efficiently for further analysis.

After finalizing the codes, we had another person perform inter rater reliability for validating the relevancy of the codes. We reached an agreement rate as follows –

Table 3.5 – Methodology – Agreement rates
Agreement rate calculations

Samples	Total units	Disagreements	Agreements	Agreement rate
Sample 1	69	7	62	62/69 = 0.8986
Sample 2	89	11	78	78/89 = 0.8764
Sample 3	18	1	17	17/18 = 0.9444
Totals	176	19	157	157/176 = 0.892

The final agreement rate over 3 samples with 176 total units of interview text came out to be 89.2%. We also had 2 pilot sessions before the final coding session for synchronization.

4. Results

What follows are the results that were analyzed and obtained from assigning codes to the interview transcript, broken up into the instructor and student section. Before getting into the detailed results, here are the student and instructor codes table displaying their respective sub-codes. The sub-codes helped to attach greater meaning to the text in part for their self-explanatory nature and are provided here for reference. All these codes help to extract information or gather/evaluate requirements for designing a feasible solution or the outline of such a system, so that it can be useful where and when it counts. We also take these results and discuss it in the context of our original solution/system with a more detailed explanation over in the discussion section.

Instructor –

Table 4.1 – Results – Instructor subcodes

Instructor subcode/theme descriptions

Sr. No.	Code	First order sub-codes	Descriptions
1.	Teaching likes		
2.	Teaching challenges		
3.	Learning objectives	Course-specific / technical	The objectives can be further categorized into those which were specific to the course and/or of the technical nature
		Generally applicable	The other objectives could be applied in general like for example teamwork.
4.	Skills	Required skills	The skills that the instructor mentioned were required.
		Desired skills	The skills that the instructor mentioned were desired.
5.	Tools	Required tools	The tools that the instructor mentioned were required.
		Desired tools	The tools that the instructor mentioned were desired.
6.	Barriers	Usage barriers	The barriers could be further split into usage barriers, which covered the barriers for bringing in the desired tools in the class.
		Technological Barriers	And technological ones which covered the barriers of the technological sort with current tools like installing.

Table 4.1 – Results – Instructor subcodes (Continued)

7.	Workarounds	Barrier workarounds	
8.	Tool Impact	Tools impact	
9.	Class changes	Teaching differently	In an ideal situation how the teacher would want to teach.
		VM-centered changes	The changes that would be made if a Virtual Machine option was available.
10.	Infrastructure changes	Project requirements	The requirements that was stated implicitly or explicitly by the instructor from their responses.
		Project necessity / Arguments for	The text in the transcript highlighting the need for a system that we were proposing.
		Project concerns	The concerns for the system that was proposed like allowing for admin access.
		Arguments against	The text highlighting arguments made against the proposed system.
		Other requirements	Any other requirements that could be gleaned from the instructor responses for informing any future solution.

Student –

Table 4.2 – Results – Student subcodes

Student subcode/theme descriptions

Sr. No.	Code	First order sub-codes	Descriptions
1.	Student likes		
2.	Student challenges		
3.	Learning objectives	Course-specific / technical	The objectives can be further categorized into those which were specific to the course and/or of the technical nature
		Generally applicable	The other objectives could be applied in general like for example teamwork.
4.	Skills	Required skills	The skills that the student mentioned were required.
		Desired skills	The skills that the student mentioned were desired.
5.	Tools	Required tools	The tools that the student mentioned were required.
		Desired tools	The tools that the student mentioned were desired.
		Other (used) tools	The tools actually used by the student given trouble with required tools.

Table 4.2 – Results – Student subcodes (Continued)

6.	Barriers	Usage barriers	The barriers could be further split into usage barriers, which covered the barriers for bringing in desired tools according to the student.
		Technological Barriers	And technological ones which covered the barriers of the technological sort like installing.
7.	Workarounds	Barrier workarounds	
8.	Tool Impact	Tools impact	
9.	Class changes	Teaching differently	In an ideal situation how the student would want to taught.
		VM-centered changes	The changes that the students would like to be seen made if a Virtual Machine option was available.
10.	Infrastructure changes	Project requirements	The requirements that was stated implicitly or explicitly by the student from their responses.
		Project necessity / Arguments for	The text in the transcript highlighting the need for a system that we were proposing.
		Project concerns	The concerns for the system that was proposed like allowing for admin access.
		Arguments against	The text highlighting arguments made against the proposed system.
		Other requirements	Any other requirements that could be gleaned from the student responses for informing any future solution.

Next we breakdown the codes further individually for the instructor and student section and touch up on their significance.

4.1. Instructor results

Now we will present each of the common themes that were applicable for the codes and follow that up with some significant individual answers that were discovered. We also go into detail and discuss the significance for both the cases.

4.1.1. Common/Repetitive instructor themes

We provide the common themes here at a glance and discuss each one individually by codes after.

Table 4.3 – Results – Instructor theme results

Instructor common themes by code

Code	*First order Sub-Code*	*Second order common themes / sub-codes*
Teaching likes		Fun class (x2), hands-on class (x3)
Teaching challenges		Language expertise (x2), Class Logistics (x3)
Learning objectives	Course-specific / technical	
	Generally applicable	Working together (x2)
Skills	Required skills	Git (x3), Python (x2), C language (x2), OS concepts (x3)
	Desired skills	
Tools	Required tools	Git (x2), make (x2), GDB (x5)
	Desired tools	iOS development environment (x2)
Barriers	Usage barriers	Configuration barrier (x3), Time barrier (x2)
	Technological Barriers	Configuration barrier (x4), Knowledge barrier (x2), Hardware barrier (x2)
Workarounds	Barrier workarounds	Staff help, Discussion forums, Internet, Peers
Tool Impact	Tools impact	Demo time (x2), Class structure (x4), class project (x2)

Table 4.3 – Results – Instructor theme results (Continued)

Class changes	Teaching differently	Physical HW (x2)
	VM-centered changes	Changes to assignment, -Also refer below-
Infrastructure changes	Project requirements	Offline connectivity (x2)
	Project necessity / Arguments for	
	Project concerns	Access control (x2)
	Arguments against	
	Other requirements	

Teaching likes

The most common reason instructors liked their class was found to be the *hands-on nature* of the class. We can form an easy hypothesis that wherever possible instructors would prefer the class to be as hands-on and fun as possible. *Fun* was another common response, albeit it alludes to the nature of the class that the instructors have set up. Any future solution would be best served with this in mind. Depending upon the instructors' intent it could also be possible to adapt the OpenStack services to make the class more fun overall. Hands-on was something we were actively trying to provide with preconfigured Virtual Machines in the system design which makes the tools more accessible.

On the one hand it's good that the instructor like the fun and hands-on nature of the class but it can be difficult to achieve without the right resources and setup. A good resource would need to provide the right balance and be as less obstructive as possible.

Teaching challenges

The common themes found for this code are the expected *language skills* that the students come in with, in the class and the instructors struggle for handling the two extremes in skill level. Other common one was *class management* or class logistics which can be affected by a number of different factors.

Nothing significant was discovered from these answers since they were mostly expected. But the fact that these were stated to be the most common ones might give an indication of lacking better support or tools adaptation. One hypotheses would be that instructors would like students to be mostly well versed or balanced with the language before coming in to the class. This may be related to the academic institute structuring their classes or the instructors needing students to full fill some strict pre-requisites. Another would be instructors requiring better resources or support, academic or tool wise for reducing issues with handling a large class.

Expected language skills from students are nothing that can be handled by any system or any future solution. The same can be said for class logistics but that at least can be alleviated in part by a solution that caters or makes the instructors' job easier when handling the tool and performing management actions. Automated python scripts for management like handling account creations was possible with this system.

Learning objectives

Common objectives were mostly stated to be *working together*. This may be an indication that there are lacking support in tool or otherwise for enabling this efficiently. Or it can be an ideal situation the instructor want occurring in classes.

Either way, a hypothesis could be that instructors favor better options for bringing students together to work in classes on top of any individual work.

Any future solution would need to also be designed around this idea and enable options for collaboration. OpenStack being open to adaptation through its services and in the context of tool availability inside virtual machines and on-demand access to server resources, could enable working together.

Skills

The most common skills requested by instructors involve proficiency with *Git, Python, C language and OS concepts*. This is more of a factual finding and does not have much significance other than indicating the most common and popular skills that are desired and required by the instructors.

Any future system could potentially help with this aspect. One hypothesis would be that wherever possible instructors would want to use the popular, in-demand skills going by current trends while also trying to pick one that is easier to adapt in the class. Most issues with configurations arise because of picking some skills/tools that are incompatible with most systems. In such cases, the instructor even has to settle for far fewer things than they would originally have liked. A possible future solution could help with this aspect by making sure all skills that have to be included, get along well. This way the instructor is free to choose what skills he wants, and students can focus on their coursework without grudging about spending time getting all the libraries, tools, etc. set up.

This system with OpenStack can provide for interpreters, compilers, architectures like x86 and operating systems like Windows, Mac; All of which properly configured and ready to go with the help of preconfigured virtual machines with the system.

Tools

The most common answers for tools that the instructors required from class were *Git, make and GDB.* This is may be because of the classes we chose to target. But they still show the tools that the future solution will need to provide for to make the instructors life easier.

One easy hypothesis would be Instructors always aiming for tools that are best served for both students and class adaptation. Same as in skills.

The common desired tool mentioned was the *iOS development environment.* Since Apple requires you to possess a MAC OS if you want to do any iOS development, which is not completely possible for everyone, this is not a completely unexpected

response. One significance of this response is that the instructors seemingly want to provide iOS development in their courses, which for some obvious reason they cannot currently, and at the same time make it accessible to everyone. It also shows the popularity of mobile development like iOS and the students desire to learn more about it which directly reflects the instructors desire to include it.

A hypothesis that can be proposed is that instructors would want to provide for development environments like these in their classes if possible but are held back by configuration, licensing and logistic issues.

Any future system would need to create the correct synergy between all the required and desired tools, especially the ones mentioned here since they are popular and useful among the instructors that we interviewed. With this system design like already discussed it would be easy to make these tools available to the instructors, and any later solution will need to account for that as well. Same is true for providing iOS development environment, since that would require a Mac OS virtual machine with the environment correctly configured by the instructor. This is easily made possible with the system design and OpenStack.

Barriers

Usage barriers

Configuration barrier

The major reason instructors cited for not being able to use some tool in their class was *configuration barriers*.

Quotes –

‘‘So whenever we try to introduce new tools even with other instructors we have to worry about the 100 or 1000 possible configurations that every individual machine will have so that’s why normally our choice is the like the intersection’’

CS 362 Ecampus SE II - Prof. Christi

“…like for instance Android development tools only work on LTS version of Ubuntu, they will install on other ones but they could be flaky”

CS 462 - PD - Prof. McGrath

Our original assumption/hypothesis about instructors having to limit their ambition because of (fear of) technological hurdles was absolutely confirmed by these responses. They seem to settle for tried and true tools because they have a long documented history of working.

We can form a hypothesis here that whenever possible instructors want to use the best possible tool for their class. But that they are frequently held back by "will this work?" type of questions.

Hence a system like the one we were proposing would have been able to help a great deal with this situation by managing all the configurations for them, so that instructors focus on teaching and students focus on learning.

And any other solution/system would have to involve the same considerations for their features if it is to be of any help solving this issue of instructors not being able to bring in some tools for fear of not being able to get it to work.

Time barrier

Another common barrier instructors mentioned was related to *time.* Not being able to adapt some tool or bring in some tool in class because of time limitations.

Quotes –

"So at this stage when they are about to graduate I find it difficult to introduce complex tools but if they are exposed to the tools from the beginning it's easier"

"And the tools and technologies that I can use depends upon the flow that they run through during the course."

CS 362 Ecampus SE II - Prof. Christi

"…and I didn't remove it because of tool problems there wasn't enough time to cover it really well."

CS 362 SE II - Dr. Groce

This can be easily countered by a system which mitigates this problem partly for the instructors. It can be used to familiarize the students to new tools, or structure it in a way that allows access since the class start which affords more time to the instructor.

Technological barriers

Configuration barrier

Configurations not only cause problems for introducing new desired tools in class but they also cause havoc with the tools that are already required in the class as well. They interfere to a great degree with the student experience in class.

Quotes –

"But just we cannot use and we have to go for something like VIM that is almost like very basic and least but we know that it will work just out of the box"

CS 362 Ecampus SE II - Prof. Christi

"…just that the setup and orientation in the first few weeks of VS is tough but after we get over that road bump it's pretty smooth"

CS 271 Ecampus Comp.Arch. - Prof. Redfield

There is no surprise then that this is one of the most common themes that was answered when we asked instructors what the students in class struggled with.

This also proves our original point and motivates for a system like ours which aims to handle these configurations to a certain extent for the students so they have an easier time with the course.

Any future solution would regardless have to take this into account.

Knowledge barrier

Another common theme for barriers to using current tools was the (insufficient) *Knowledge barrier*. This indicated lack of student expertise and skill level with the tool which leads to them being unable to use/install it properly and eventual frustration. The lack of knowledge could be due to a variety of reasons – lack of documentation, laziness, bad support, etc.

Quote –

"But just we cannot use and we have to go for something like VIM that is almost like very basic and least but we know that it will work just out of the box"

CS 362 Ecampus SE II - Prof. Christi

This can be easily countered with a system which removes this equation from students' hands. Then the students need only concern themselves with how to go about using the tool to perform their task.

Hardware barrier

Another common theme in barriers was the problem with hardware. Incompatible hardware systems be it a MAC causing problems with running tools or Windows not being able to run the tool. This barrier indicates that hardware more often than not is an essential component in any class which deals with software. And it often leads to more problems than it solves. It calls for a solution which handles the intricacies of compatibility between the software and the hardware it runs on, and also for its availability in its absence.

Quote –

"So the frequent things that come up are like I said the MAC users who for some reason can't get it working properly."

CS 271 Ecampus Comp.Arch. - Prof. Redfield

This is also easily mitigated with a system like ours which deals with all hardware availability by virtue of virtualization and compatibility for the students.

Any future solution would also have to take this into account since they are bound to run into this issue.

Workarounds

The workarounds tried for dealing with the technological barriers was found to commonly include *getting help from staff, peers, using discussion forums and the internet.*

It can be observed that none of these methods are a guaranteed solution to the problem faced by the student. And if nothing works out, students and instructors

would often resolve to the "use whatever works" kind of strategy. This can have a lot of negative consequences for all concerned.

A system which takes care of most of the technological barriers would arguably also help students not spend too much time seeking help unnecessarily for paltry and easily handled issues.

Tool Impact

Tools were commonly found to impact *demo time, class structure and the class project.*

Quotes –

"Oh yeah! I have VS demo in the class where I step through and debug the code."

CS 271 Ecampus Comp.Arch. - Prof. Redfield

"They do in the sense that I could do projects for example in the case of physical hardware the last project involved writing a driver a user level driver for a USB rocket launcher because they could plug it in"

CS 444 OS II - Prof. McGrath

One more consequence of having configuration issues and troubles with tools were found in this code in the form of demonstration time. The instructors may have likely spent more time than was perhaps necessary demoing the tool and how it works. It might also have been extra sessions which were apart from the regular coursework.

It was also found to impact the class structure in most cases and the class project. This indicates the dependence of the class on tools and thus motivates the fact that everything has to go right with the tools, for a class to run smoothly, which is not always the case.

A portion of time can be cut from demoing with a system that is already set up with the tool, but how the tool works can be an important lesson for the students, which cannot be helped with easily with the use of any system.

Over dependence on the tools can also be managed just as well with a system like above. Plus there can be greater confidence on the part of the instructor for things

running smoothly. Thus they can include any tool without fear and teach without being held back.

Class changes

Most instructors when asked how they wanted to teach differently wanted to include some or the other kind of *hardware* in the class which would enhance it.

Quote –

"The only thing I would change the class is to have everything on the Arduino"

CS 444 BendCampus OS II - Prof. Rubin

Since hardware cannot be justifiably provided with any software system or solution, this knowledge can go towards informing any future solution which can make this possible.

Even simulated hardware would not be that useful since it is not as powerful or effective.

Quote –

"…so if I had that remote Kernel Debugger the fifth project would become a RKD (remote kernel debugging) project rather than a USB driver project"

CS 444 OS II - Prof. McGrath

When talking about changes in context of Virtual machines, most common answer was the *changes to the current assignments.* Virtual machines setup would make the assignment more accessible/challenging and would allow the Instructor to experiment.

This does make a case for a system that we were designing and its usefulness. Any future solution would also serve better with a system that kept this in mind, and allows for robust and complete assignment design.

Infrastructure changes

The only common answers found for this section dropped under the project requirement and project concerns sub-code themes.

Quotes –

‘‘They actually have to get their hands on it which means that a lot of their learning is going to take place offline where I can't see it.’’

CS 362 SE II - Dr. Groce

“I wouldn’t want them to have the access to do that”

CS 444 OS II - Prof. McGrath

Some instructors wanted *offline connectivity* to be accounted for from our system design and other instructors were concerned over *access control* in the underlying structure. Offline connectivity was not something we designed for from the system since it was being offered over the internet, but there were some extra steps that could have been undertaken as a possible solution to this.

Any future solution would also need to make sure that offline connectivity is not something that is ignored when designing. This is mostly because students more often than not can work at locations with no internet access. Not providing for it may end up wasting the students’ time when they would rather get some work done.

Access control will also be need to be accounted for, for granting varying degrees of permissions. Otherwise, it could lead to students messing up other students work, breaking the system, wiping up any important data, etc. The system we had enabled role based access control which could be set up quite efficiently for dealing with control permissions.

4.1.2. Individual instructor themes

Before we talk about the significant individual themes which came forth from our analyzed coded sections which we find very useful for consideration from any future system design and how it came to influence/change our design, we look at a special code “infrastructure changes”. Since this code was designed specifically for gathering requirements for our system, it’s worth talking about the individual themes for this code the most after discussing the common ones. It not only also covers most others codes individual themes but also can be applied generally for a system that is not our own. Here are the themes at a glance with a more involved discussion after.

Table 4.4 – Results – Instructor theme results

Instructor individual themes by code

Sr. No.	*Code*	*First order Sub-Code*	*Individual sub-codes / themes*
1.	Teaching likes		Helping students understand
2.	Teaching challenges		Teaching online, Platform configuration
3.	Learning objectives	Course-specific/technical	Subject specific
		Generally applicable	Nothing significant
4.	Skills	Required skills	Version control system – SVN and Git, JUnit, commercial tools like QTP, Visual Studio w/ MASM plugin, code coverage tools, Simulations, Statistical tool R, MATLAB, Linux, concurrent programming, QEMU, physical hardware, virtual machines, CAD tools, TI code studio, MS project via Office 365 SharePoint.
		Desired skills	Eclipse, parallelism, OS pipelining, GDB, gcov, remote debugging in Linux kernel, Windows OS other tools
5.	Tools	Required tools	Cygwin, UNIX, VMware, Ubuntu with G++, gcc, Edgecam, Delmia, Solidworks, Catia, MS project, OpenCV, LabView, Robot OS, Perl
		Desired tools	Industry focused tools, spin model checker, CBMC, architectures like x86, ARM Virtual machine, Windows OS, CNC, Espirit cnc, Siemens technomatics, hosting platforms
6.	Barriers	Usage barriers	Entry barrier, Tool/platform preference barrier, Effort barrier, Improper usage barrier, Scaling barrier, Money barrier and academic university investment/structure barrier, Logistical barrier, Hardware barrier, Hosting barrier
		Technological Barriers	Knowledge barrier, Tool usage barrier, Installation barrier, Tool limitation barrier, Resource limitation barrier, Technological barrier

Table 4.4 – Results – Instructor theme results (Continued)

7.	Workarounds	Barrier workarounds	Familiar tools and previous experience
8.	Tool Impact	Tools impact	
9.	Class changes	Teaching differently	Industry tools, Ecampus goals, Assignment changes, Teach with Windows OS, Integrate class parts
		VM-centered changes	
10.	Infrastructure changes	Project requirements	Access control, Working tandem, VM Storage, Remote Kernel Debugging, Providing configurable/customizable resources for lacking campus resources, Providing ARM (different architecture/OS) VMs, Providing scalable as-needed resources
		Project necessity / Arguments for	Running tools - CBMC; VS; etc., Missing relevant background, Bring different class parts together, Alleviate logistical issues, Introductory classes, Spending time on tool issues, Introducing new tools, Providing configurable resources, Providing disposable VMs, Providing configured tools
		Project concerns	Processing, Working tandem, Storage, Instructor configuration control
		Arguments against	Offline connectivity, Centralized resources validity/usefulness
		Other requirements	Arduino simulator, Scalable measures, Providing configurable/customizable resources for lacking campus resources

Infrastructure changes

For discussing the data gathered or results from this code effectively we break it down into separate sub-codes/themes. Some themes may be repeated as a result but given the context it should still make sense.

Project requirements

This theme was applied for direct responses which indicated what the instructor wanted from the system. Some instructors asked for things specifically while some were gathered from different responses.

Access control we already talked about in context of project concerns. But for this theme it applies more towards the level of permissions that would be required from any system. This holds especially true for a system which hands of resources in the hands of the students. Without an established fine grained layer of levels each establishing how much access or control the students are granted the system could crash quite frequently.

Working tandem was also desired to be present from the system. Basically features that should enable or elevate student interaction and working together. This may be simultaneously or otherwise.

Quote –

"See that's the biggest issue, the Linux kernel build tree is large, to do a full build takes to the order of tens of GB of storage, which means they can't be run on the network file system."

CS 444 OS II - Prof. McGrath

Virtual Machine storage is another requirement that was found. For some classes the storage requirements can go up very high, for these reasons having a system which does not allow for students to have access to more storage as desired will not be very useful. This provision could be made with OpenStack and our system design.

Quote –

"…so if I had that remote KD the fifth project would become a RKD (remote kernel debugging) project rather than a USB driver project"

CS 444 OS II - Prof. McGrath

Remote kernel debugging was one requirement from a specific class. It was desired to enable debugging of the kernel remotely for enhancing an aspect of the class. Although this may not be as generally applicable, it still is an important requirement

that speaks to the adaptability and enhancing abilities of a system. Both of which are important for a useful system. Our system could be adapted to provide this as well.

Quote –

"Different web frameworks, the department has limited web interface, the departments web framework collection is small. You get JS, PHP, CGI but very limited CGI and that's it they didn't have Django, no rails, none of the more popular platforms"

CS 462 - PD - Prof. McGrath

Another requirement that was bought forth, which is pretty prevalent in its lack but there are not many different ways to handle, was *providing configurable/customizable resources for lacking campus resources.* Some systems in place at campus are old and would require considerable resources to update. In some projects for some classes the students demand the latest libraries/tools/frameworks or hardware resources all ready and setup with some libraries which cannot be easily provided. In such cases a system much like ours which makes up for this lack and is easily expandable to include and provide such resources to the students can be of use a great deal. Some cases also call for the resources to be configurable and customizable by the students themselves or already tweaked for them and ready to go.

Quote –

"Ideally I would give each student 2 pieces of Hardware, one ARM and one x86 and have them work on it."

CS 444 OS II - Prof. McGrath

Providing ARM or access to different architectures, operating systems was another such project requirement. Some classes suffered terribly because of the inability to introduce an operating system easily while others were stuck trying to figure out how to make a tool work properly. Some instructors could have enhanced the class with the inclusion of different operating systems. So for a system to enable the inclusion of architectures and operating system will be useful for the instructors and allow them experimentation. Much like what could be possible with this system.

Quote –

"…so if they want sort of the one of the unsupported web frameworks because that's what they wanted to do their project with, then that's where we... that's the most cause of needing a VM actually."

CS 462 - PD - Prof. McGrath

Next requirement to come forth was *providing for scalable as-needed resources.* In some classes students had need for resources which were specific to their project. This can include a server, storage, etc. A system which can provide on that would be useful. For some resources like storage, or even virtual machines they would not only have to be made accessible but in some cases they were required to be scalable as well because of student demand. OpenStack makes this aspect easily possible.

Project necessity / Arguments for

Wherever the instructor responses arguably favored for a system solution like ours, we grouped them under this theme. At some places they even speak to its necessity.

Quotes –

"…and it depends on whether they are coming from an engineering background or stat background."

ST 443/543 - Applied Stochastic Models - Prof. Emerson

"CBMC is something that I might teach"

CS 362 SE II - Dr. Groce

"So the courses I think you might get a lot more utility from something like this are probably ST 511 512 513 which are, they are introductions to applied data analysis and the students have a lab section where they are supposed to learn to use R"

ST 443/543 - Applied Stochastic Models - Prof. Emerson

"…yeah I think I would try to make it more integrated"

ME 413: COMPUTER-AIDED DESIGN AND MANUFACTURING (Karl Haapala)

The first and most obvious reason was for *running tools – CBMC, VS, etc.* The solution we were proposing would easily let instructors have tools available for students to work on, without much prior setup or knowledge from them.

Another argument for the necessity of such a system was when students were *missing relevant background.* Even in such cases our system would be useful because it

would already take care of this for the students. They would not need to have had any exposure to the tool.

Next argument in favor for the design of the system included the ability to *bring class parts together*. In situations where the instructor has a class split into parts, a system like this could help the students to see the bigger picture and bring them together cohesively.

This system could also *help alleviate some logistical issues* where possible, by handling or automating a part of it for the instructor.

This system would be especially useful in *introductory classes* where the students might not need to know or learn about the tool in detail. A working version of the tool all ready to go will let the students focus on learning the course content. For introductory classes in Statistics where students need to get the programming language "R" working is such an example.

Quote –

"And honestly the most challenging bit is finding one that works and gets out of the way, if that makes any sense, because most of the time the students are spending so much time dealing with the tool that they lose sight of the fact that they are there to learn a concept not to learn a tool. Yeah and so that's by far the most challenging bit - is getting the tool to fade into the background so that they can just do their project."

CS 444 OS II - Prof. McGrath

As already discussed this system would enable for students to *spend less time haggling with the tool and its setup configurations*. Another reason why the system would be useful and speaks to its necessity.

It will be just as easy to *introduce new tools* with this system as it is for adapting to current structure.

The system could also *provide configurable resources and disposable virtual machines*. Very useful in situations where they are needed in class.

All these reasons are strong and favorable for our system/solution. Any future design could also account for these. They help to serve as a strong reason for taking this development further, besides.

Project concerns

Processing was one such concern that was raised about our system and weather it could handle multiple students accessing the system and working, especially on due dates. With server load balancing and efficient algorithms we deigned it possible to handle such a scenario with our system.

Would the system be able to handle *storage* distribution was another such concern. Since the underlying implementation, OpenStack, we choose to use at the time allowed for control over the storage components, we did not think this would be a big issue.

Another concern worth mentioning was the level of *instructor configuration control*. The system was being designed especially around this, so instructors were allowed as much control as they would need for their classes but more control or options were going to be abstracted away or hidden. This allowed the instructor sufficient control over the resources and classes and they could be customized in their quotas assigned, with additional access to more, if required.

Arguments against

One main argument that could be made against the design of this system would have to be *offline connectivity,* allowing students to work the system without an internet, locally. This system at the time had made no provisions for this and had not even considered this option. The way this system was being designed, making the system work offline would have required some more work but would not have been impossible. Any future solution could learn this lesson early and design it around this requirement.

Quote –

"…but people just don't use it, they want to do their own thing and what I provide generally isn't good enough to be all the useful to them"

ROB 421/521 - Applied Robotics - Dr. Hurst

Some issues were raised around the *validity or usefulness of centralized resources* like the one we were proposing. Although these concerns were valid, they only were a disadvantage in a small number of cases. In one argument, since the class only used one tool/software, the system would have caused a bigger overhead than was necessary. This is a valid concern if the class remained static but a counter argument would be that the system could enable the use of 2 tools/software and allow for bigger experimentation if ever there was a need.

These arguments although valid and fair against the system do not outweigh the advantages that it would have bought.

Other requirements

These requirements were grouped in this category for future systems that could provide them. As the system that was going to be designed at the time, we could not reliably account for this, but we still wanted to list and gather them, so that it does not go to waste.

One such requirement for a class was an *Arduino simulator* which simulated the hardware for Arduino circuits. This could be an entire project by itself but would be very useful in Operating systems classes where cost was an important issue, and hardware like Arduino would be difficult to provide for a large class size.

Quote –

"…yeah it involves a good deal of that (programming). We actually have that relatively well managed now. We could probably do more ambitious things if it were more scalable"

CS 362 SE II - Dr. Groce

Other requirement that was touched on had to do with *scalable measures* in classes. This could be tools, infrastructure and other technical things. Using scalable

algorithms and providing resources properly, it can be relatively managed with the system design.

Teaching likes

One instructor mentioned he liked that the course *helped students understand.* That is definitely something that can be accomplished a lot better with the help of the system because it allows for different approaches to be tried out with minimal effort.

Teaching challenges

Teaching online was one of the challenges that was found. It's mostly because the students and instructors are not face to face and communication can become a real issue. Even in such circumstances since the system was being designed to be accessible online through an endpoint or uniform resource locator (URL) it could potentially also help to alleviate some of these issues in teaching online. Any technical issues or technological support could be managed with the system to make instructors and students life easier teaching online.

Platform configuration was another challenge instructors faced. The problem was finding a platform either hardware or software that worked well after proper setup and configuration, and then faded into the background. That can be well taken care of with this system as the configuration and setup will only be required to be done once by the instructor and this system will just place that in the hands of the students, essentially getting out of the way of coursework and assignments. Hence we can make another case in favor of the system.

Skills

Required Skills

Instructors mentioned skills with *version control system – SVN and Git, JUnit, commercial tools like QTP, Visual Studio w/ MASM plugin, code coverage tools, Simulations, Statistical tool R, MATLAB, Linux, concurrent programming, QEMU, physical hardware, virtual machines, CAD tools, TI code studio, MS project via*

Office 365 SharePoint. The good thing with this system is that it is able to provide any tool barring licensing or other issues easily with the system for instructors to focus on just choosing the tools they want and students to just focus on working and their skills with these technologies.

Desired Skills

For this code, there were skills desired for *Eclipse, parallelism, OS pipelining, GDB, gcov, remote debugging in Linux kernel, Windows OS other tools.* The same argument can be made for these results which speaks to the necessity for a system like this one.

Tools

Required Tools

Much of the same tools and arguments as were mentioned for skills apply along with *Cygwin, UNIX, VMware, Ubuntu with G++, gcc, Edgecam, Delmia, Solidworks, Catia, MS project, OpenCV, LabView, Robot OS, Perl.*

Desired Tools

Industry focused tools, spin model checker, CBMC, architectures like x86, ARM Virtual machine, Windows OS, CNC, Espirit cnc, Siemens technomatics, hosting platforms. The desired tools mentioned here can also be mounted on a Virtual Machine with our system and provided to the instructors for modifying/configuring and supplying to the students.

Barrier

Usage barrier

Quote –

"…no no tools that they know, so until recently they were asked to use VIM and C programming so that was the standard and if now I suddenly introduce eclipse that introduces a whole new layer of complexity and that is apart from the complexity of the course itself."

CS 362 Ecampus SE II - Prof. Christi

Entry barrier is the barrier when there is too much learning that needs to be performed for a tool which the instructor wants to bring in to the class. This is what ultimately prevents them from adapting it in the class. With a system like ours, the learning for setup and configurations is already completed. It makes it much easier to prevent this barrier from occurring.

Quote –

"But it's like if they cannot install it on their machine they are not going to do it they are not going to take that extra step to install Citrix and probably figure out a way to connect to the university server and do the things"

CS 362 Ecampus SE II - Prof. Christi

Tool/platform preference barrier is also something that shows up for students. Most of the time Ecampus and other students, develop a preference for using a certain tool and platform which prevents instructors from bringing in the tools that they might want to include. There is always the risk of students not using it even if the option is available. Our system includes the option to have multiple tools available or have the students try out the tool that the instructor wants to add.

Effort barrier ties in closely to the above barrier, where students do not want to expend effort looking at other options or using them. We can remove or try to manage this barrier by allowing for less effort to be expended with the use of our system depending on how the instructor decides to use it.

Quote –

"I used to have students complain about not getting SVN to run that they installed some GUI to SVN and it didn't work well."

CS 362 SE II - Dr. Groce

Improper usage barrier is where the students do not use the tool the way it was designed, is effective to be used. There is not much that can be done to overcome this barrier with the system.

Scaling barrier is another such barrier that is preventing instructors from using a particular technology in the class. This barrier can be managed with efficient algorithms and the proper setup, performed for the system.

Quote –

"We don't even speak the word Windows because it's not free and it's not what the university is invested in and by that I mean both emotionally and institutionally invested and not even necessarily from a monetary standpoint."

CS 344 OS I Ecampus - Prof. Brewster

Money barrier and academic university investment/structure barrier prevents instructor from using their desired tools because of monetary issues like licensing or being forced to use a structure or operating system which the university has already in place. Both of the barriers are hard to tackle for letting instructors use the tools they want to bring in to the class even with a system like ours.

Quote –

"…yeah, mostly just cost, licensing, relearning, restructuring and updating the lab, training the TAs, etc."

ME 413: COMPUTER-AIDED DESIGN AND MANUFACTURING (Karl Haapala)

Logistical barrier i.e. cost of software licensing, relearning with new tools, restructuring course, updating labs, training TAs, incomplete knowledge of newer tools is another reason which prevents instructors from using new tools. Not much can be done for this barrier with the system but any future system could potentially learn this and navigate around it.

Hardware barrier is when the lack of hardware availability prevents the use of certain tools in class. This shouldn't be much problem for the system since it can make available different architectures and operating systems through a disk image on a virtual machine.

Quote –

"…and I think I came up with a working, workable solution that involves remote building machine. And where that gets hosted is an issue,"

CS 462 - PD - Prof. McGrath

Hosting barrier is the absence of a proper hosting infrastructure that restricts instructors' ambition for newer tools in class. This can be made available with the system since it can handle server resources.

Technological barrier

Knowledge barrier prevents instructors from changing the current tools because only the tools that have been used before and people have exposure with can be used effectively in class. In such case the system can be used to provide the necessary exposure.

Tool usage barrier is another case of improper usage of the tool in the class with current tools. Since this barrier seems to be very prevalent, adequate steps can be taken to manage this to an extent. Unfortunately, the system will not be of much use for this barrier prevention.

Installation barrier talks about problems with tool installations and setup. Since this can be easily handled for the student with the system, it can be used for preventing this barrier.

Quote –

"…and that's another thing I would really like to bring into the class, it's that perspective, the comparison perspective. I think I can do that with QEMU but it's really slow."

CS 444 OS II - Prof. McGrath

Tool limitation barrier speaks about the current tool limitations which prevents it from being used effectively or conveniently. The native speed or usage of the tool is not something that can be handled with the system. But if it is provided on the system, it could be configured and optimized to run the way it wouldn't have been possible previously.

Quote –

"Delmia had some licensing problems in the number of licenses that could be active at one time. The labs are limited to 20."

ME 413: COMPUTER-AIDED DESIGN AND MANUFACTURING (Karl Haapala)

Resource limitation barrier speaks to when the resources currently present are lacking or restrictive and affect the class adversely. Even in such cases since the system has virtualized services and is digital, this barrier can be mitigated.

Quote –

‘‘For the CAD they do it on their own machine and they are also available on Citrix, and then with EDGECAM and with DELMIA those licenses are tied to specific lab computers’’

ME 413: COMPUTER-AIDED DESIGN AND MANUFACTURING (Karl Haapala)

Technological barrier is concerned with software and tools that are required in the class, and when they have to be split between some software being available only in labs and some on the personal devices. Since this service and system is available online and is virtualized, all software and tools are available over a centralized structure in the same place.

Workarounds

Familiar tools and previous experience was one workaround instructor tried for getting past issues with current tools. They restricted themselves to only familiar or previously used tools since it was much easier to use and get up off the ground. This restricts the ambition. With the system, it is possible to provide alternatives and work out a strategy that works best.

Class changes

Teaching differently

Quote –

‘‘A lot like and every time it’s like, even in our ecampus instructors meeting we have these discussions and that is one of the biggest challenge that we are facing is like we would like to use better tools, tools that'll be readily used in the industry’’

CS 362 Ecampus SE II - Prof. Christi

Industry tools was one of changes where the instructor wanted to teach tools used in industry more to give exposure and experience that would help in the industry. This can be easily managed by providing the tool with the system.

Quote –

"We feel the goals for the ecampus students are different from the on campus ones so it affects the decisions we make"

CS 362 Ecampus SE II - Prof. Christi

Ecampus goals forces instructors to employ a different strategy when it comes to teaching and tools. All this can be supported and enhanced with the use of the system.

Assignment changes would be bought about by the instructors if they had no restriction on technology. This can be effectively supported with the system.

Quote –

"We don't use windows, it's kind of baffling. It's kind of backwards."

CS 344 OS I Ecampus - Prof. Brewster

Teach with Windows OS was one answer by an instructor if they had no restriction on technology. Operating system can be easily supplied on the virtual machine with the system.

Quote –

"…yeah I think I would try to make it more integrated, so if there was a tool like SIEMENS TECHNOMATIX which integrates with a CAD and CAM software, if I could integrate all in one tool, that might be helpful,"

ME 413: COMPUTER-AIDED DESIGN AND MANUFACTURING (Karl Haapala)

Integrate class parts was one change which was sought. With the proper setup, it would be possible to come up with a strategy involving the system that would enable the instructor to combine separate class parts to give that big picture view for the benefit of the students.

4.2. Student results

Now we touch up on the student side of the argument. This is also broken up like above into its own sections as follows.

4.2.1. Common/Repetitive student themes

Here we present the common student themes at a glance and follow up with discussion after.

Table 4.5 – Results – Student theme results

Student common themes by code

Sr. No.	*Code*	*First order Sub-Code*	*Second order common sub-codes*
1.	Student likes		
2.	Student challenges		
3.	Learning objectives	Course-specific/technical	
		Generally applicable	
4.	Skills	Required skills	Unix/tool skills (x2), scripting language skills (x2), C prog. skills (x2), OS concepts (x2)
		Desired skills	
5.	Tools	Required tools	
		Desired tools	Industry centric tools (x2)
		Other (used) tools	

Table 4.5 – Results – Student theme results (Continued)

6.	Barriers	Usage barriers	
		Technological Barriers	Tool configuration barrier (x3), insufficient tool/configuration knowledge barrier (x2), tool usage difficulty barrier (x2)
7.	Workarounds	Barrier workarounds	Peers (x2)
8.	Tool Impact	Tools impact	
9.	Class changes	Teaching differently	
		VM-centered changes	
10.	Infrastructure changes	Project requirements	
		Project necessity / Arguments for	Tool setup/configuration issues (x3)
		Project concerns	
		Arguments against	Experience setting up tool (x2)
		Other requirements	

The common themes were mostly influenced by the classes we got responses back from but they still go towards informing what the system will need to cover to be useful for most cases. Here we look at each code individually.

Skills

Required skills

The most common required skills that came forth were the *skills required for tools specifically UNIX in one class, script writing skills, C programming skills and Operating system concepts*.

Quotes –

"I did not have much experience using UNIX and I enjoyed learning more details about how the UNIX system worked and the commands that were available to users."

"I also needed to learn BASH scripting"

"…as well as additional C programming skills involving processes and networking features which I had not learned before this course."

Alex Samuel Fall 2015 - Email CS 344 OS-1 EC - Prof. Brewster

All these technologies or libraries can be set up very conveniently with our system for the students, a UNIX operating system, Python (or any other script language) and C interpreters/compilers and some special provisions can even be made for teaching OS concepts. All they would need to focus on then would be actually just using these for enhancing their own skills or abilities. By using the tool on the system we can make sure that they only focus on the skills and not on the tools. We can't increase their skills for them by them just using the system. We can't provide skills with the system. We can provide the base to work on.

Thus with this system and likely with any other future system, it would have to be possible to provide on such skills which are required in the class for it to be useful. Since the requirements for other similar classes is not likely to be different, the usefulness can even be extended across more classrooms barring special conditions. These can be assumed to be a representative sample in the meanwhile.

Tools

Desired Tools

Common responses here included a focus on *industry centric tools.* Essentially skills with tools that were or would be useful from a software industry standpoint after the student graduates.

Quote –

"I think it was useful to write some of those programs I have written in other languages but it would have given me a better understanding of what kind of things you would do in industry with assembly."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Except for exceptionally complex or resource heavy tools, it would be possible to provide on such with our system. This is an especially common response from students and its usefulness is pretty overt. For any future solution this would also have to be made possible like it is with this system.

Barriers

Technological barriers

Barriers students mostly faced with current tools included -

Tool configuration barrier

The commonality of this barrier again goes to show the prevalence of configuration problems with tools and supports our original hypothesis and assumption.

Quote –

"I had issues with VISUAL STUDIO at first, getting that set up was a hassle"

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Students were faced with a lot of problems and consequences not getting the tool to setup and run properly. This again, as the basis, of why we were designing the system is taken care of for the student, so that they can just focus on doing their work.

Insufficient tool/configuration knowledge barrier

Another fairly common barrier that came to pass here was the limited knowledge with the tool and its different configurations that students possessed which caused them problems using the required tools in the class.

Quote –

"One thing I could not understand was opening VS from a specific file ".sln", that was the one that was configured."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Since we handle the tool setup and configurations for the student with our system design, it helps to alleviate this problem.

Tool usage difficulty barrier

This was another common response which students gave that speaks to the problems that the students can have just using the tool properly because of a variety of reasons.

Quote –

"Yeah, getting it running was fine but getting it running the way I wanted to was difficult."

Morgan Shirley Winter 2014 CS 362 SE II - Dr. Groce

Along with setup and configurations, some other complex steps can also be taken care of for the student which would make this problem manageable but not completely unavoidable. It would be possible to make a switch easily with the system and provide options. There also seems to be an emphasis on making it run a certain way so that all the issues get resolved. This also causes undue difficulties with using the tool. With the system set up properly, this would be less of an issue because we would try to perform all the right steps already for the student.

Workarounds

The most common workarounds to problems with current tools were asking for help from *Peers*.

With the system design it can be made possible to enhance this aspect if there should be a problem that needs additional help.

Infrastructure changes

Project necessity / Arguments for

As already discussed and proven a very strong reason for developing a system like this that bubbled forth from the responses was unsurprisingly *Tool setup/configuration issue.* That is the original reason for taking on this study and project.

Quotes –

‘‘So it was kind of a pain to get VS on my computers and then halfway thru the course my computer broke and so I had to go through the whole process again, with a different computer that wasn’t mine.’’

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

“Yeah, getting it running was fine but getting it running the way I wanted to was difficult.”

“…not from what I saw, none of the peers that I asked had that problem so it was more a problem with how I was setting up gcov.”

“So stuff like gcov its useful to learn yourself but if it’s some sort of commercial tool/software or some big graphical interface software that would take a long time to set up just right but isn’t particularly difficult to id say that would be really helpful to have in a VM.’’

Morgan Shirley Winter 2014 CS 362 SE II - Dr. Groce

The undue consequences and results for this major problem as already discussed above are very severe. It makes another argument in favor of this system design.

Arguments against

One common argument made against the system was that of explicit hand holding and the students’ need to *experience setting up tool.*

Quotes –

‘‘That would be really useful but I can also see the appeal of figuring things out for ourselves because that would be the case in industry.’’

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

‘‘Well I would want the experience of setting it up but I think that its fine if after the first Assignment or 2, the VM was provided and it went like even if you could not set it up previously or you did set up I made some tweaks here that runs the way I want it too, I wouldn’t want to be robbed of the experience of having to struggle with setting it up myself cause that’s what we would have to do if we wanted to use it elsewhere.’’

Morgan Shirley Winter 2014 CS 362 SE II - Dr. Groce

‘‘I am actually opposed to this level of hand holding.’’

Daniel O'Farrel Fall 2015 - Email CS 344 OS-1 EC - Prof. Brewster

Since this system was meant to optional and not required to be used, some of these concerns do not apply. In courses where such experience is desired the students will have the option to do either or. It can act as a backup if nothing works and student is

nearing the deadline. Also in some courses such knowledge or experience does not add any value, thus this concern does not hold weight.

4.2.2. Individual student themes

Here are the individual results of each student code at a glance. Since there are only 4 students, we provide all of it without culling. But we only go on to discuss the most important or significant.

Like we did in instructor we will look at the "infrastructure changes" individual results first and then follow up with others.

Table 4.6 – Results – Student theme results

Student individual themes by code

Sr. No.	*Code*	*First order Sub-Code*	*Second order individual sub-codes*
1.	Student likes		Perspectives, Lecture structure-topics, learning tools, specific class assignment
2.	Student challenges		Problems with tools-applying course concepts, deciphering large codebases, learning new script language, forking concept
3.	Learning objectives	Course-specific / technical	Related to 271, Related to 362, Related to 344, Related to 344
		Generally applicable	
4.	Skills	Required skills	Computer architecture concepts, testing concepts, software design skills, basic python skills
		Desired skills	Hardware device skills-industry relevant skills, skills with testing frameworks, Advanced C programming, skill application towards a complex assignment, advanced python skills
5.	Tools	Required tools	VS-MASM-Irvine library, SVN-C-makefiles-gcuv, flip engr. server-PUTTY-VIM,
		Desired tools	Linux-NASM, test frameworks
		Other (used) tools	Sublime text-PUTTY-CISCO VPN-DSynchronize, Notepad++, Google chat

Table 4.6 – Results – Student theme results (Continued)

6.	Barriers	Usage barriers	Tool preference barrier, habit barrier, university structure barrier
		Technological Barriers	Improper tool usage, improper tool setup, canvas slow loads and time zone
7.	Workarounds	Barrier workarounds	Retracing-specific step workaround around config. knowledge barrier-reconfiguring/tool restart, Online-manual--Instructor, workaround for problematic parts of class structure / problematic tools in class, steps automation for tool-BASH textfile info save for easy access-PC left running-instructor help
8.	Tool Impact	Tools impact	Distracts from coursework-class structured around VS-Tool complexity(overpowered)-enjoyable after figured out, tool learning usefulness, tools help teach (and learn) relaxed manner-tool usage in assignments benefitted students-led to class enjoyment and to get more out of class, logistical tool impact, tool usage technical satisfaction, tool made assignment easier-tool usage difficulty-PC left running-Canvas message board issues load times time zone delay
9.	Class changes	Teaching differently	Class topics-measures for enhanced understanding, measures for industry exposure/industry tools, assignment changes-more feedback in class, Change in discussion forum
		VM-centered changes	Changes supporting working together (pair prog.), logistical/demonstration changes
10.	Infrastructure changes	Project requirements	Features for working together(utilizing centralized resources)-tool availability, easy message board
		Project necessity / Arguments for	Class enjoyment after issue resolution, counter against tool config-*tool setup issues*, *getting tool running right way*-*improper tool setup*-good to have options-advanced software avail.-*set up difficulty*, better assignments
		Project concerns	VM/system setup concerns
		Arguments against	*Figuring things out bc useful later as exp. in industry*, class taught well no changes-*experience setting up tool*-VM setup difficulty, *hand holding comes in way of gaining knowledge or learning how to gain knowledge*
		Other requirements	Lab component, hardware availability

Infrastructure changes

We start with infrastructure changes since this code again was specially made to collect system design parameters from entire transcript.

Project requirements

The student oriented requirements for the system included *features for working together so that it utilizes a centralized resource like our system, making sure different tools are available as required and perhaps an easy message/discussion board.*

Quotes –

"Cause personally I learn best when in a group. And then if you had like this centralized thing it might be nice if there was a really easy way to kind of share snippets of code to help debug and stuff. Yea just working together."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

"So stuff like gcov its useful to learn yourself but if it's some sort of commercial tool/software or some big graphical interface software that would take a long time to set up just right but isn't particularly difficult to id say that would be really helpful to have in a VM."

Morgan Shirley Winter 2014 CS 362 SE II - Dr. Groce

We already covered these requirements to a certain degree in the instructor section, and here it means the same thing. Students wanted features that would encourage or make it possible for students to get together and work to a greater degree than what was possible currently. Things like pair programming. Some students could have had a much easier time with the assignment or the class in general had they access to some tool. Other students had issues with the chosen message board system, e.g. Canvas in this case.

Our system should be able to handle working together in the context of the base it was standing on. So we could easily enable sharing of resources between students and add feature for pair programming, to enable working together. Making tools available and ready to go was one of the main features we were designing around, so this

requirement would have been easily met. Message boards like Piazza among a host of others are already solid services. We were not planning to provide a separate message board because it would have meant stretching the system out too thin. But for a system like ours it could have been easily expandable and added in the future, if it became necessary.

These things also go to inform any future solution about possible features and necessary enhancements.

Project necessity / Arguments for

Here we look at some more reasons, student side, of how and why the system or the idea being designed was necessary and useful.

Quote –

"Yes! And I enjoyed the class the way he had set it up after I had figured out the setting up and some recurrent issues that popped up, which by then I could just go and take care of."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Some students went on to state that they *started enjoying the class after they finally resolved all the tool issues*. If the enjoyment was affected so much by just issues with tools alone, it really pushes the need for some system which at least attempts to take care of such. It would at least go on to reduce some stress from students and instructors alike.

As already discussed so often this system was being designed as an effective *counter against tool configurations.* Since tool configuration issues are such a prevalent threat a system like this would really go a long way to help both concerned parties.

The same can be argued in favor of the system when it comes to *tool setup issues.*

This system was also being designed to provide options, so it speaks highly to its importance when it was also mentioned in the interview that *having/providing multiple options* is a very good thing.

It was also possible to make *advanced software available*. One more area where the system was necessitated and its importance validated.

Another area found in the responses was for *better assignments or better already existing assignments.* So this is one more area where the system would be of good use.

Project concerns

Quote –

"If we just had the tools in class in the VM it won't be that useful just cause the effort of downloading, installing, running the VM is probably about the same as just configuring the software by itself."

Morgan Shirley Winter 2014 CS 362 SE II - Dr. Groce

One student expressed concerns over the *setting up of the system and Virtual Machines* and the difficulty one can get into while using the system among other aspects. Although the concern is valid, most of the settings for the system would already be handled and the interface was purposely being made user friendly with the students and instructors in mind. There would be some "getting up to speed" machinations involved but they would be easy enough and would be a one-time deal.

Arguments against

Quote –

"I don't know if I have any thoughts there, I thought it was taught pretty well."

Morgan Shirley Winter 2014 CS 362 SE II - Dr. Groce

One student felt that the *class needed no changes* and that it was handled just the right way. For this scenario, one can argue against designing and usefulness of the system. But for obvious reasons this is only a small minority.

Other argument was for *Virtual machine and general system setup* like already mentioned, which the student thought would be of the same complexity as setting up

tools anyways. For one and as already discussed over in concerns, it was not being designed to be such and second the benefits far outweighed any risks and effort.

Other requirements

Quote –

"Would have liked a lab component to the class to mess around with circuits."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

One student felt the class would be better suited with a *laboratory component* where they could work on circuits directly so they can apply their knowledge gained in Computer Architecture, Assembly language.

One student felt that *different hardware* would have been nice to have been made available to get real hands on experience in the class.

Quote –

"Would have liked to have worked on some hardware device, and to write programs for that, instead of just writing programs that we have already written in higher level languages."

Student - Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Student likes

One student mentioned *perspectives* as one of reasons he liked the class. With system design like ours it would not be hard to bring more of that in the class. Another student mentioned that getting to *learn the tools* in the class was their favorite. This also can be improved upon by the use of our system which makes it easy to provide on tools.

Student challenges

Problems with the tools in the class was one reason student faced challenges. It only goes to show the necessity for a system like ours and the overarching need for dealing with this issue in general.

Skills

Required skills

Computer architecture concepts, testing concepts were required skills to learn or master in some classes as felt by the students. These specific concepts can be provided upon in a more accessible or easier way with the system because it can provide different testing frameworks and architectures for students to explore and play around with which may not always be possible traditionally.

Desired skills

Hardware device skills is something that can be provided at least in a simulated manner. *Industry relevant skills* are so highly desired, with our system adaptation this should be possible. *Project complexity* was something that the students wanted as well. With this system and with any system being considered in future all this and more should be possible. This system allows for apt experimentation if it was missing before to fine tune assignments.

Tools

Required Tools

All the tools mentioned here *Visual Studio, MASM plugin, Irvine library, SVN, C, make, g-cov, putty, vim* and more could be easily provided at all the right configurations irrespective of platform or prior settings. Students will just login and start working.

Desired Tools

Even some desired tools wished for – *Linux, NASM plugin, and test frameworks* can also be easily provided.

Other (used) tools

These tools which the students resort to use as replacements because some issues inevitably happened with current tools, can also be provided for. For example *Sublime text, Cisco VPN, DSynchronize, Notepad++ and Google chat.*

Barriers

Usage barriers

Quote –

"Convenience and it maybe just for me cause I tend to run Linux, so it's just easy."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Tool preference barrier was one of the barriers that prevented instructors from bringing in some of their desired tools. Over the course of the term the students and instructors get used to some tools, from other classes, previous experience, peer recommendation, etc. This prevents perhaps other better tools to be adapted because a preference has been established which is hard to break. This is not a big issue for a system we were proposing since the tools our audience preferred, and the tools they didn't know about could be brought in and used without much hassle.

Quote –

"Plus a lot of the other work we do in classes runs on Linux servers here, so it's just familiar and comfortable, and I know how to get around."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

A *habit barrier* was another such similar one where people got habituated with their tools and platforms and always fell back to them. These can be solved much the same way.

Along the same lines as above, *University structure barrier* was a barrier where the university choose the infrastructure which influenced what the instructor could not change or do and had to be forced to use a particular platform. For example the flip engineering server with UNIX. Although the entire barrier could not be removed, it

can be managed to a certain degree with this system. The system can build on the university infrastructure and provide different operating systems, tools, etc. where the traditional format could not.

Technological barriers

Quote –

‘‘So in gcov, when a file is used by multiple binaries, I had a lot of trouble getting that to work’’

Morgan Shirley Winter 2014 CS 362 SE II - Dr. Groce

Improper tool usage was a barrier with current tool in classes which made students and instructors life difficult. The students did not use the tool correctly which caused issues. This may be by intention, oversight, misuse, abuse or any number of other reasons. This is partly taken care of by the system as it may pertain to potential oversight usage scenarios by having the proper settings in place already. For others there is not much that can done except to have proper safeguards in place.

Quote –

‘‘The entire setup process for visual studio with MASM is just kind of hacked together, and it’s weird. It’s a little confusing to follow.’’

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Improper tool setup was a barrier where students could not setup the tool properly. This may be during the actual installation or afterwards when using the tool for a specific purpose. This is also taken care of partly for the student, tool installation, students would not need to be bothered with because it will be present in the system, and tool setup for a task could be taken into account to a certain degree by performing these for them like above.

Workarounds

Quote –

‘‘And they could not figure it out either and so I kept trying from the beginning, on how to set everything up, from the website which has the procedure listed for setup. Eventually I got it working.’’

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Retracing was a workaround mentioned for overcoming tool not working as expected or failing in some regard, where the students retraces the steps they read or were told to get the tool working a certain way. This is a common workaround for students but with the system since the tool is already properly configured they would not have to perform this repeatedly.

Quote –

‘‘One thing I could not understand was opening VS from a specific file ".sln", that was the one that was configured. That was the only way I could get my code to actually run on my computer.”

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Some students *performed some specific step around a barrier* because they lacked the configuration knowledge to identify the problem. Since this configuration will be taken care of for the student, this will not be much of an issue.

Reconfiguring / tool restart was another workaround employed where students kept tinkering with the tool till it worked or restarted it continuously. The same solution with our system should apply.

Quote –

‘‘I struggled with writing my programs using Vim on the OSU server because I thought the program was difficult to use. I decided to write my programs locally using Sublime Text and then transfer my files to the OSU server for compilation. With DSynchronize I was able to sync my files with the server automatically any time I made changes to a file.”

Alex Samuel Fall 2015 - Email CS 344 OS-1 EC - Prof. Brewster

Some students had a *workaround for problematic parts of class structure / problematic tools in class*, where they choose to use a different tool for perhaps getting the job done easier or used a tool they had previous familiarity with. Another cause for a system like ours being designed.

Tool Impact

Quote –

“So I guess it would take my mind off the tool and more on the coursework”

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

One consequence of using the tools in class was that it *distracted from coursework.* This is another reason in favor of our system.

Another *class was structured around Visual Studio,* which indicated the reliance of the class on the tool and getting all the things just right. This system was designed to mitigate the issues that might arise from such a scenario.

Quote –

“Overall I think VS is a little overpowered for our level.”

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

One student felt the choice of the *tool was too complicated (overpowered)* for the class. Different options can be tried out with the help of the system in such a case. At least with the configurations and setup taken care of, the focus can be just on using the tool for the task.

One student claims the class was *enjoyable after all tool issues were sorted*. With the use of the system it is hoped that this is achieved much earlier for the student.

Class Changes

Teaching differently

One student wanted a change in *class topics.* With this system in place all kinds of different experimentations could be performed.

Quote –

"I don't know how relevant all that is to what I'll be doing after college since that might be a bit more electrical engineering but I think it would be interesting to get slightly deeper understanding of that."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Measures for enhanced understanding is another thing a student requested for in the class. And the same can be said as above for improving or enhancing the overall class.

Measures for industry exposure/industry tools is something we have talked about extensively already.

Assignment changes was another request for a change in the class. If it has anything to do with tools or infrastructure it could probably be possible with the system.

VM centered changes

Quote –

"Beneficial to do a little more pair programming, not like co-authors but more like just watch each other's back. You encourage each other more, help each other to understand the algorithm, like what's going on."

Billy Buffum Fall 2015 CS 271 CA&AssemblyL - Dr. Redfield

Changes supporting working together (pair programming) is also something that would be possible to change in class with the help of a Virtual machine and this system in place.

Quote –

"I think the only change I would like to see would be in the lecture videos. Specifically, if the instructor would be demonstrating examples of code, I would like to see the instructor using the VM in the video itself."

Alex Samuel Fall 2015 - Email CS 344 OS-1 EC - Prof. Brewster

Logistical/demonstration changes where instructors changed the way they demonstrated something in class which the students had problems with, could also be brought about with the system depending on the use case for example tool or setup demonstration could be handled easily with an apt tool in the Virtual Machine.

5. Discussion of system design

In this section we finally present the system we had under design. It could not be completed unfortunately because of the complications we faced for the platform we had decided to base it on.

We talk about the design of the system, the original assumptions, how and why we decided to collect more data for informing the system design, along with the technical details of the platform and how it could not satisfy our specifications due to the technical shortcomings.

It is hoped that this will give a better understanding of the overall system, why it was being made this way and suggest better strategies and/or lessons for any solution or system that is decided to be built in the future. The lessons learned from this attempt should still be worthwhile.

5.1. Platform discussion

Before we head into the specifics of the system design, some proper context and discussion of the platform i.e. OpenStack, would serve well, and how we came to use it.

We started from the problem of configurations and how they are adversely affecting the instructors' ability to teach a certain ideal way. Most of the times it seemed the instructor had wanted to provide some tool in class but had to decide against because of the issues with platform configurations, software configurations among a host of other settings that would need to be handled and setup just right. We observed this was obviously holding the teachers back and had them settling with tried and true methods or safer approaches. Even all the tools that were currently used in the class however were not foolproof and had their own set of problems and configuration issues.

We sought to help instructors with that. OpenStack was almost successfully at the time being used in one class. So we started to look into that. We realized that through preconfigured Virtual Machines made accessible through OpenStack, we would be able to provide the tools instructors wanted to put in the class at the right configurations. All we had to do was make the exact copy of instructors' disk image containing the tool at the right configuration already, which was what the students would want and the instructors wanted to have them setup on their machine and provide it through OpenStack Glance service to the student. It would also allow experimentation to see what would work best.

We started with that base but after evaluating the classes offered for the term, we found that OpenStack would not only be useful for a lot of classes in this scenario but it would also be useful in classes where students needed infrastructure resources for their classwork or research work. Because OpenStack as a service would even be able to provide resources like storage, disposable virtual machines, etc. basically infrastructure support as necessary.

This essentially broke down our requirements from the system into three categories – Student as User, Student as Admin and Student as Explorer, which are discussed in the System design section.

Here we take a look at OpenStack services which we were looking into and how it fell short.

5.2. OpenStack discussion

OpenStack is essentially an operating system for your Cloud infrastructure. This basically means it gives you proper control over the servers through its different services. The core services offered by OpenStack are as follows –

Swift – This service is for managing object storage.

Keystone – This service is for authentication and authorizations.
Nova – OpenStack compute layer. Manages lifecycle of instances.
Neutron – For managing networks for OpenStack instances.
Cinder – This is used for block storage management.
Glance – OpenStack image service. Manages virtual machine disk images.
Horizon – OpenStack dashboard user interface into managing the Cloud.

Our audience mostly consisted of students and instructors. Handing them an OpenStack API (Application Programming Interface) to work with would be too complicated and would give them unnecessary power over the resources. Thus we decided instead to focus on Horizon. It would be much easier for people to use and would be widely accessible. We set out to modify the user interface provided by Horizon and adapt it to our needs. As it was provided out of the box, it had too many features the audience did not need or care for. There was also risk of someone causing havoc with the system.

For a certain class, the super admin, who oversaw the servers would create the necessary space for the class. Automate the adding of students, set the proper permissions and quotas and send off account logging details to the instructor and the student. The audience would just need to log in and work off on our modified user interface. The intention was to make it easier for students to spin off Virtual Machines and request resources within the context and limit of the class. And for instructors to manage it, without much hassle.

At first OpenStack made it considerably difficult to set per user quotas and access permissions because of the way it was structured. It also turned out that the instructor could not be given proper control over his class without making him the super admin and presenting him the admins interface, which was not practical. When they eventually fixed that through later versions and updates, there were some other issues that cropped up. OpenStack use of domains was one. Domains were added later and had a finer grained hierarchy and control, set one level up from the super admin. It

enabled compartmentalization of resources which suited our needs. So we changed our design to have a class be a domain and instructor be a domain admin. But the domain admin permissions were set to that of the super admin which bought us back to square one. It also wasn't supported in the API completely and properly which restricted what horizon could do with domains. Horizon also natively did not support domain scoped tokens which essentially made it useless for domains and domain role based access control. This was not fixed even later, till their recent Mitaka 2016 release. One of the main developers on the project thought it was to be replaced by Hierarchical Projects and gave up on domains till recently.

Horizon did not have the same level of control or features that could compete with the API which was found later on, even though internally the calls were getting converted to actual API calls which is how Horizon did half the things. But it severely restricted what we could and wanted to accomplish with Horizon.

We tried to work around this but the differences in the API versions themselves with no backwards compatibility and the restrictive and unrealized potential with Horizon itself we had to abort the plan going forward, since the future did not look like it would bring improvement or be acted on as soon, as we wanted to go around building this.

5.3. System design

Here we present the system design document. Some of it is presented as is, to give an idea of our approach with the design and where we were going with it. It also indicates the progress we were making and eventual hurdles we ran into.

I] Motivation for the way document is structured:

Before creating a hierarchy and classifying the population it is important to observe why the system will be designed. It will mainly be designed to help students and instructors perform an activity (for e.g. working on Operating Systems assignments) in a better and more convenient manner than what they have currently available to them.

The students and also the instructors will perform these activities every time in relation to a particular **subject/course** that they have signed up for/are teaching. *Since it can be observed that the system will have to change and adapt heavily to the courses that are offered and not so much on the instructors and students alone, we felt it was necessary to list all of the subjects in CS (and later non-CS) who may use/need it, first! Before moving on in the hierarchy to students and instructors.*

A separate system can also be designed for **research** purposes, but this is a different use case since it will require the system to be radically different and need-based. The same can be said for some **special subjects**, courses like **Senior Design project** for which the resources requested will be need-based.

Following are the list of courses in CS both at the grad and undergrad level where this system may be useful (for all terms, both grad and undergrad). The current focus is on CS classes, instructors and students because they are the most immediate population that will need this system.

II] List of CS courses: (NOTE: For some subjects the use may not be overtly/quickly apparent, for these we will need to talk with the instructor directly to determine usage.) All the information for this was gleaned from the classes.oregonstate web space.

Legend:

Student as user - Signifies that the student will just use the VM provided with preconfigured images, already setup with the Tools/SW that will be used in the class, with minimal to no interaction of the student whatsoever with OpenStack.

Student as admin - Signifies that the student will interact with OpenStack services to spin-off VMs/instances (also networking and Databases) from the assigned quota. This is a superset of the above case.

Student as explorer - Signifies that the student will have varying needs from OpenStack services. (Can either be handed off resources or request for some)

	Student	**Student**	**Student**	**Not**	**Tools/SW**

	as user	as admin	as explorer	relevant	used
WINTER					
	UnderGRAD				
CS 161 (Section 001) INTRO TO COMPUTER SCIENCE I (Parham Mocello, Jennifer) **[OnCampus]** CS 161 (Section 400) INTRO TO COMPUTER SCIENCE I (Alcon, Tim) **[Ecampus]**					C++, any IDE
CS 162 (Section 001) INTRO TO COMPUTER SCIENCE II (Rooker, Terry Lee) **[OnCampus]** CS 162 (Section 400) INTRO TO COMPUTER SCIENCE II (Rooker, Terry Lee) **[Ecampus]**					C++, any IDE
CS 165 ACCELERATED INTRO TO COMP SCI (Alcon, Tim) **[Ecampus]**					C++, any IDE
CS 195 INTRODUCTION TO WEB AUTHORING (VanLonden, Pam) **[Ecampus][Exams/Workshops on Campus]**					Dreamweaver, ShiftEdit, Chrome

CS 225 (Section 400) DISCRETE STRUCTURES IN CS (Ehsan, Samina) **[Ecampus]**					
CS 261 (Section 001) DATA STRUCTURES (Metoyer, Ron A) **[OnCampus]** CS 261 (Section 400) DATA STRUCTURES (Ehsan, Samina) **[Ecampus**]					C, any IDE (VS 2012, Xcode)
CS 271 (Section 001) COMPUTER ARCH & ASSEM LANGUAGE (McGrath, D Kevin) **[OnCampus]** CS 271 (Section 400) COMPUTER ARCH & ASSEM LANGUAGE (Redfield, Stephen James) **[ECampus]**					NASM Compiler with Linux, Windows, Mac
CS 290 (Section 400) WEB DEVELOPMENT (Wolford, Justin D) **[OnCampus]** CS 290 (Section 001) WEB DEVELOPMENT (Scaffidi, Christopher Paul) **[ECampus]**					
CS 312 LINUX SYSTEM ADMINISTRATION (Jensen, Carlos) **[OnCampus]**					

CS 325 (Section 001) ANALYSIS OF ALGORITHMS(Borradaile, Glencora) **[OnCampus]** CS 325 (Section 400) ANALYSIS OF ALGORITHMS(Schutfort, Julianne Marie) **[Ecampus]**					
CS 340 INTRODUCTION TO DATABASES (Sarbaziazad, Ameneh) **[ECampus]**					MySQL (or any other) DB
CS 344 (Section 001) OPERATING SYSTEMS I (Chaney, Raymond Jesse) **[OnCampus]** CS 344 (Section 400) OPERATING SYSTEMS I (Brewster, Benjamin C) CS 344 (Section 501) OPERATING SYSTEMS I (Rubin, Marc Joseph) **[OnCampus]**					
CS 352 (Section 001) INTRO TO USABILITY ENGINEERING (Burnett, Margaret M) **[OnCampus]** CS 352 (Section 400) INTRO TO USABILITY ENGINEERING (Azarbakht, Amirhosein) **[Ecampus]**					CogTool, Balsamiq (ecampus)

CS 362 (Section 400) SOFTWARE ENGINEERING II (Christi, Arpit Moses) **[ECampus]**					Debugging tools, IDE
CS 362 (Section 501) SOFTWARE ENGINEERING II (Rubin, Marc Joseph) **[OnCampus]** CS 362 (Section 001) SOFTWARE ENGINEERING II (Groce, Alex) **[OnCampus]**					Debugging tools, Linux VM (Marc Rubin)
CS 361 (Section 001) ^SOFTWARE ENGINEERING I (Dig, Daniel) **[OnCampus]** CS 361 (Section 400) ^SOFTWARE ENGINEERING I (Ahmed, Iftekhar) **[Ecampus]**					
CS 372 (Section 001) INTRO TO COMPUTER NETWORKS (Noroozoliaee, Mohammadjavad) **[OnCampus]** CS 372 (Section 400) INTRO TO COMPUTER NETWORKS (Redfield, Stephen James) **[Ecampus]**					Networking resource

CS 381 PROGRAMMING LANGUAGE FUND (Smeltzer, Karl) **[OnCampus]**					Haskell, Emacs, Vim
CS 419 (Section 002) ST/ COMPUTER GRAPHICS (Zhang, Eugene) **[OnCampus]**					OpenGL, C, IDE (Can be put in both categories)
CS 419 (Section 004) ST/ OPEN SOURCE (Jensen, Carlos) **[OnCampus]**					
CS 419 (Section 005) ST/ DEFENSE AGAINST DARK ARTS (McGrath, D Kevin) **[OnCampus]**					
CS 419 (Section 400) SELECTED TOPICS/COMPUTER SCI (McGrath, D Kevin)(Software Projects) **[OnCampus]**					
CS 419 (Section 001) ST/ CRYPTOGRAPHY (Rosulek, Michael) **[OnCampus]**					
CS 440 DATABASE MANAGEMENT SYSTEMS (Termehchy, Arash) **[OnCampus]**					MySQL DB
CS 457 COMPUTER GRAPHICS SHADERS (Bailey, Michael) **[OnCampus]**					OpenGL, GLSL,

CS 462 SENIOR SOFTWARE ENGIN PROJECT (McGrath, D Kevin) **[OnCampus]**					
CS 476 ADVANCED COMPUTER NETWORKING (Hamdaoui, Bechir) **[OnCampus]**					
CS 480 TRANSLATORS (Parham Mocello, Jennifer) **[OnCampus]**					gforth
CS 496 MOBILE/CLOUD SOFTWARE DEVEL (Wolford, Justin D) **[OnCampus]**					May use some tools
	GRAD				
CS 519 (Section 001) ST/ADV APPLIED CRYPTOGRAPHY (Yavuz, Attila) **[OnCampus]**					
CS 519 (Section 002) ST/ADVANCED SYSTEM SECURITY (Bobba, Rakesh) **[OnCampus]**					Networking resource, OS
CS 519 (Section 003) ST/ PERSONAS METHODS IN HCI (Burnett, Margaret M) **[OnCampus]**					
CS 519 (Section 005) ST/NATURALANGUAGE PROCESSING (Fern, Xiaoli Z) **[OnCampus]**					

CS 519 (Section 006) ST/ INTERCONNECTION NETWORKS (Chen, Lizhong) **[OnCampus]**					
CS 520 GRAPH THEORY WITH APPLS TO CMP (Nayyeri, Amir) **[OnCampus]**					
CS 531 ARTIFICIAL INTELLIGENCE (Tadepalli, Prasad) **[OnCampus]**					
CS 536 PROBABILISTIC GRAPHICAL MODELS (Wong, Weng-Keen) **[OnCampus]**					
CS 540 DATABASE MANAGEMENT SYSTEMS (Termehchy, Arash) **[OnCampus]**					
CS 551 COMPUTER GRAPHICS (Zhang, Eugene) **[OnCampus]**					OpenGL, Graphic tools
CS 556 COMPUTER VISION (Todorovic, Sinisa) **[OnCampus]**					OpenGL, Graphic tools
CS 557 COMPUTER GRAPHICS SHADERS (Bailey, Michael) **[OnCampus]**					OpenGL, Graphic tools
CS 570 HIGH PERFORMANCE ARCHITECTURE (Lee, Ben)					

[OnCampus]					
CS 576 ADVANCED COMPUTER NETWORKING (Hamdaoui, Bechir) **[OnCampus]**					
CS 581 PROGRAMMING LANGUAGES (Walkingshaw, Eric T) **[OnCampus]**					Coq SW
SPRING					
	UGRAD				
CS 419 (Section 002) ST/ALGRTHMS F/COMP MOLEC BIO (Hendrix, David A) **[OnCampus]**					
CS 419 (Section 005) ST/ NETWORK SECURITY (Yavuz, Attila) **[OnCampus]**					Networking resource
CS 419 (Section 001) ST/ INTRO INFO VISUALIZATION (Metoyer, Ron A.) **[OnCampus]**					D3.js, vis tools
CS 419 (Section 004) ST/SCIENTIFIC VISUALIZATION (Bailey, Michael) **[OnCampus]**					vis tool
CS 419 (Section 400) ST/SOFTWARE PROJECTS (McGrath, D Kevin) **[OnCampus]**					

CS 434 MACHINE LEARNING & DATA MINING (Fern, Xiaoli Z) **[OnCampus]**					Tools
CS 472 COMPUTER ARCHITECTURE (Chen, Lizhong) **[OnCampus]**					Instances, Networking
CS 475 (Section 001) INTRO TO PARALLEL PROGRAMMING (Bailey, Michael) **[OnCampus]**					OpenCL, tools
	GRAD				
CS 519 (Section 004) ST/ DISTRIBUTED SYS SECURITY (Bobba, Rakesh) **[OnCampus**					Launch multiple VMs and test multiple security protocols
CS 519 (Section 003) ST/ VECTR & TENSOR FIELD VISUL (Zhang, Eugene) **[OnCampus]**					Vis Tools
CS 519 (Section 005) ST/ COMPUTATIONAL GEOMETRY (Nayyeri, Amir) **[OnCampus]**					
CS 523 ADVANCED ALGORITHMS (Borradaile, Glencora) **[OnCampus]**					

CS 533 INTELLIGENT AGENTS & DECISION (Fern, Alan Paul) **[OnCampus]**					Tools, SW
CS 569 ST/STATIC AN/MOD CHECK SFT ENG (Groce, Alex) **[OnCampus]**					
FALL					
	UGRAD				
CS 419 (Section 001) ST/NETWRK MTHDS IN BIOINFOMTCS (Ramsey, Stephen A) **[OnCampus]**					
CS 419 (Section 003) ST/GEOM MDLNG IN COMP GRAPHICS (Zhang, Eugene) **[OnCampus]**					Graphic, Vis Tools, OpenGL
CS 491 CS SIM & GAME PROGRAMMING (Bailey, Michael) **[OnCampus]**					Tools can come pre-installed
	GRAD				
CS 519 (Section 003) ST/ HCI RESEARCH METHODS (Jensen, Carlos) **[OnCampus]**					
CS 519 (Section 001) ST/ DIGITAL IMAGE PROCESSING (Todorovic, Sinisa) **[OnCampus]**					Matlab, Tools

CS 527 ERROR-CORRECTING CODES (Bose, Bella) **[OnCampus]**					
CS 534 MACHINE LEARNING (Fern, Xiaoli Z) **[OnCampus]**					Machine Learning Packages- WEKA,LibSVM, SVM-Light, LibLINEAR, Vowpal Wabbit, Open source machine learning software, Mallet, MLDemo
CS 550 INTRO TO COMPUTER GRAPHICS (Metoyer, Ron A) **[OnCampus]**					OpenGL, VS 2012 IDE
CS569 FIELD STUDIES IN SW ENGR **[OnCampus]**					
CS 583 FUNCTIONAL PROGRAMMING (Walkingshaw, Eric T) **[OnCampus]**					Haskell

Quick analysis (Some courses are slash courses and maybe repeated; any unique course is taken into account) -

	Student as User	Student as Admin	Student as explorer	Not relevant	Courses (listed)
Winter- ugrad	13	8	6	4	41(14 Ecampus)
Winter- grad	4	4	4	6	15

Spring- ugrad	4	2	1	1	8
Spring- grad	2	1	0	3	6
Fall- ugrad	2	0	1	1	3
Fall- grad	4	0	1	3	7
TOTAL	29	15	13	18	80

III] Identified populations/stakeholders:

From the above discussion the following populations can be identified easily.

- Students
- Instructors

These populations can be categorized by **subject** as -

- CS Subject
  - Instructors
    - Ecampus
    - on-campus
  - Students
    - grads
      - Ecampus
      - on-campus
    - undergrads
      - Ecampus
      - on-campus

III. a] Discussion for these populations and categorization:

Making the classification based on the courses that are actually taught seems to be the correct way to go about this. Because we can then easily reason about them and their implementation. For e.g. -

- Subject - CS Operating Systems

  Both may need to have some adjustments made because of the campus and the instructors need.

  - Instructors / TAs -
    - Ecampus

      Will include OpenStack as part of the curriculum and will need to have the option to control the resources and images to upload - if this is done right the instructor will not even need to touch his own dashboard because his only job is to make the image available then the

students will use this system for VMs and real hacking, the instructor may need to make changes to the asst. based on the system he's given or based on the asst. we may need to change the system. May involve additional use-cases for the instructor where he may need to step in to take some admin/mgmt. actions.

- On-Campus

  As the system is delivered through the cloud (online) there will be no changes for on-campus as well.

- Students
  - Ecampus

    Ecampus students will login through the dashboard and create instances.
  - On-Campus

    Will be the same

IV] Requirements categorization:

As discussed briefly above the requirements for each group of subjects are broken down into the categories as below. What follows is a discussion for each of these categories -

IV. a] Observations for "Student as User":

This category will handle cases where student just needs a VM with preconfigured images which he can just start using right away. Since this scenario will only handle tools/SW being provisioned to the students, and students interacting only through an endpoint, it does not cover instance sharing or students working in groups. It is assumed that the endpoints will be generated for each student →
For this scenario there are 2 stakeholders - *Instructors, Students*

The *Superadmin role* in this would be to create project/tenant, set quota for it. Within the project, create users then create instances for them from the image provided by the instructor (this can be done in different ways but here I am just listing one way). Then provide a list of endpoints to the instructor. This will all be automated

with a python script (After creating the project/tenant, the script can be run either by the Instructor or the Superadmin). The least a Superadmin would have to do is to create project and set default quotas for them based on requirement.
The *instructors' role* in this would be to just supply those images with tools, SW already setup and configured, after which instances would be created from them with predefined flavors and reasonable quotas already set as above. An endpoint for accessing the above instance will be communicated with the student.

The *students' role* would be to just access the instance and start working.

An example scenario for the above use case using CS519/419 - Information Vis:

Setup - Windows OS; Use D3.js library for visualization; Use a tool like Jimbo for collaborative work;

Current method actually used in class - Setup D3.js to work on your system. Setup Jimbo to use on your system. Correctly configure these tools to work with your native OS. Start actual work of the class/assignments.

How OpenStack can help - Instructor provides a Windows Glance image with D3.js and Jimbo correctly configured for direct use.

Admin/TAs - Create 'x' number of instances from above image. Choose flavor for instances from the default quota assigned by SuperAdmin. Get and communicate the endpoints for the created instances.
Students - Receive endpoints for accessing VM and start actual class work/assignments.

IV. b] Observations for "Student as an Admin":

This scenario is for students who will need to interact with OpenStack services and spin off virtual machines for themselves. As mentioned before this scenario acts as a super set for the one above. Which signifies that in addition to the

students being in charge of their services, we would also be able to provide them preconfigured images through Glance if the class so desires. This will be based off a strict template.

For this scenario there are 2 stakeholders - *Instructors*, *Students*

The *Superadmin role* in this scenario would be to create the project/tenant and set quotas for them. Depending on how the instructor structures his course, in order to restrict instance sharing, the Superadmin may need to create a project for each user when all the students are working alone on an assignment or create a project with some specific team member size or create just a single project for an entire class. Since there will be multiple assignments, in order to make submissions and grading easier, each assignment will be a project/tenant and will be based off above. The users will be created and assigned to the project based on the requirement, their quotas will also be set accordingly. Next the Superadmin will relay the assigned username/password for the students to access their Student dashboard.

The *instructors' role* in this scenario would be to monitor the resources for the student and make occasional changes if a need arises. Supply preconfigured images if need be.

The *students' role* would be to receive their username/password and access the Student dashboard. Create instances within the quota set for them, from the dashboard.

An example scenario of this use case using a real class CS344 Operating system:

Setup - Linux OS;

Current method actually used in class - Ship Linux image to students and require them to have it up and running with everything configured correctly.

How OpenStack can help - Instructor provides a Linux Glance image.

Students - Login to horizon, use the Linux image to create however many instances they require within their assigned quota and start working on the assignment.

What follows is an implementation specific discussion for this category which is not thoroughly completed as of yet. It is only meant to portray an outline.

IV. b] [a] Categorization of subjects for "Student as an Admin":

From the listing of the courses it can be checked that the resources the classes require usually may fall into Instances (plain Virtual Machines), Databases and Networking. All of which can be provided through OpenStack.

We will have **three cases** that we would need to consider; all classes that will have a *similar requirement from the system* will be put into one category, classes that will need *tweaking based on the use cases*.

What follows is the discussion of the similarities and differences that will need to be considered and prioritized and discussed upon for requirements and implementation.

The most effective way to go about this would be to list the subjects based on a particular resource and check whether their implementation will be the same. And the same for a case by case basis.

Case I

- CS Subjects - Networking resource

  List of subjects with similar requirement -

  Intro to Computer Networks

  Advanced comp networking

  CS 519 - Applied Cryptography

  CS 519 (Section 002) ST/ADVANCED SYSTEM SECURITY

  - Instructors
    - E_Campus - Requirement/Implementation

      Not enough data to form a general common usage scenario for networking resources. [http://docs.openstack.org/admin-guide-cloud/content/arch_overview.html]

      -*Implementation*: Networking resources can be provided by OpenStack Neutron which can be installed separately.

        - On_Campus – Req./Implementation [To be expanded]
    - Students
        - E_Campus – Req./Implementation [To be expanded]
        - On_Campus – Req./Implementation [To be expanded]

- CS Subjects - Database resource

  List of subjects with similar requirement

  Introduction to Databases

  CS440 DBMS

  CS 540 DATABASE MANAGEMENT SYSTEMS (Termehchy, Arash)

    - Instructors
        - E_Campus – Req./Implementation

          Will access the database resource setup through OpenStack. This will have more managerial aspects.

          Implementation - Trove is OpenStack DBaaS infrastructure, but seems to be still in development. Some features like quota management have not been completed yet. [Link: https://wiki.openstack.org/wiki/Trove#trove-api]
        - On_Campus – Req./Implementation

          Same
    - Students
        - E_Campus – Req./Implementation

          Will access the same database resource for assignment purposes.
        - On_Campus – Req./Implementation

          Same

- CS Subjects - Operating systems(instances)

  List of subjects with similar requirement

  Linux system admin

  Operating systems

CS419 Defense against the Dark Arts

Mobile/Cloud SD

- Instructors
  - E_Campus – Req./Implementation/Use-case
    The instructor uploads the image that the students will use, and will need to have admin actions to manage resources if needed.
    Implementation- Almost completed.
  - On_Campus – Req./Implementation
    Same
- Students
  - E_Campus – Req./Implementation
    Launch instances with given images through the dashboard.
  - On_Campus – Req./Implementation
    Same

- CS Subjects - Graphics resource

  List of subjects with similar requirement

  Shaders

  Computer Graphics

  Computer Animation

  - Instructors
    - E_Campus – Req./Implementation
      Provide additional resources for Graphical applications which can include number of cores, more RAM etc. For this the instructor will only play a resource monitor role.
      Implementation - Should not be any different than above (because all we want is some system which grants more resources e.g. cores, RAM to the requester) so almost completed. Each individual request will be common on this side i.e.; Case I. Can also come with a tool installed in the image.
    - On_Campus – Req./Implementation [Same]

- Students
  - E_Campus – Req./Implementation

    Get the resources as requested from the system. This is not the system that will handle requests on a case-by-case basis. That will be covered by Case II. This system design will handle common requests uniform to all students.
  - On_Campus - Req/Implementation [Same]

Case II

Any variations to the above model of resource consumption for which the system will have to be tweaked (will depend on the typical nature of the class and the instructor), maybe on a need-based criteria. This also brings into question the feasibility aspects of such a system and to what extent these can be covered.

These will be gleaned from the above Case and decided upon for feasibility of inclusion.

IV. c] Discussion for "Student as an Explorer":

This will cover classes such as Senior Project and research needs for graduates, for which there is no common use-case and no variation to the common mode because a request issuing from such a source will be highly independent of each other and the system will have to be designed so as to allow variations for these resources to be made available as needed by the requester. And as such does not have any template that can be applied.

5.4. Prototype plans

Here we present the ongoing plans we had for the prototype, presented as is to give an idea of where we were going and trying to accomplish.

Domain admin functionality is currently not possible in Horizon because of the lack of domain scoped tokens support/presence in Horizon. Current work to include them in Horizon so as to enable this will be completed by milestone Kilo 2 [**EDIT**- Version: kilo-3 Code name: k3 Expected: 2015-03-19]. Although this can be done and is already achieved by some people through the CLI using curl and REST APIs.

In the absence of this support in Horizon, which is essential for this project, we can try to simulate the domain admin by making use of what's currently possible and present. What is currently possible is the use of Admin and member roles which we can assign to teacher and student respectively. But the main drawback of assigning teacher as Admin in absence of a domain admin is that the teacher becomes the super admin and can do pretty much anything.

There is a way to restrict/remove things from the teacher-admin Horizon in order to prevent the teacher from having too much power and making unwanted changes. Plus the teacher does not care for nor require some of the additional things that are given to an admin by default.

The list of things that should change and that we changed or experimented with from the admin-teacher horizon, are listed below:

For teacher-admin horizon:

- **Project dashboard**
  - Nothing needs to be removed (Confirm with teacher)

- **Admin dashboard**
  - Removed the “Hypervisors” and “Host Aggregates” panels from the Admin dashboard.
  - Everything else looks important to include.
  - Tweaks here to make it teacher-friendly.

- **Identity dashboard**
  - For Domains panel, only list the domains the admin is part of and hide the others from view.
    - No fine granularity of roles – only admin and member
    - Horizon has implemented something called the RBAC (Role Based Access Control) for the data that is displayed to the user. Based on that the data presented can be restricted based on the roles someone has, since only admin and member are possible right now and we need something like domain admin this does not seem to be possible just yet.
  - For Projects panel, only list the projects which the admin is supposed to manage.
    - This will also require a finer granularity for the roles.
  - For Users panel, create the option to add users here easily as opposed to creating one by one. Also list only those users here which the admin-teacher has to manage as opposed to listing everyone.
    - This is a UI related work and can be done.
  - Removed Roles panel from the Identity dashboard.

For student-user horizon:

- **Project dashboard**

- Only thing to change right now is the instance sharing that is possible between all members of a particular project. This can cause a lot of issues. To prevent this we want to restrict users to only the instances that they create.

Things to discuss / work on:

- The UI - Horizon
  - Brainstorm how the interfaces are going to look like and what needs to be there and what doesn't
  - What would be the consequences of having every user in their own project?!
    - This would solve the quota problem
    - This would also solve the instance, volume sharing problem.
    - We would need a script that would generate all the projects, set quotas for them and put the users in it in a particular domain. The users would be supplied by the teacher in a CSV or some similar format to the super admin in charge of the actual hardware resources, the admin would run the script and assign the resources and set the teacher as the admin of the domain.
      - Script would need to be run by admin – would be Linux + OpenStack curl commands
        - Take a CSV from the teacher containing all students names
        - Read from the CSV, get the required data
        - Create a class domain
        - In the domain above create a project/tenant for every student
        - Assign flavors and quotas for each project
        - Create and assign each student-user to their own project

6. Conclusion

Software configuration issues as discussed abundantly, do indeed have the capacity to stir up disastrous consequences. But much of those issues can also be managed with the right system setup and preparations.

One such system was discussed involving a cloud tool called OpenStack, and the drawbacks which held back the full completion of the design. The system was also discussed in the context of the data found and how it could potentially help instructors once realized.

A lot of requirements and safeguards were collected which can effectively go towards the building of a new system which learned from this study. The major issues and findings that stand out for both the instructor and student are, the barriers with tool configurations; the haphazard trial and error workaround approach that goes towards handling these; how the tools can end up being a crutch for the class due to their overreliance and complexity, and negatively impacting the students; And the fact that students want more industry exposure in classes and why it can be a problem to implement.

Finally with virtualization or through any new emerging technology in the cloud, we can reach a stage where the instructors can cherry pick the design for their class, try out different things and teach without hurdles.

Bibliography

OSU course catalog –
http://catalog.oregonstate.edu/CourseDescription.aspx?level=grad

Instructor class webpages –
http://classes.engr.oregonstate.edu/

OpenStack official webpage –
https://www.openstack.org/

[1] - *Cory Kissinger, Margaret Burnett, Simone Stumpf, Neeraja Subrahmaniyan, Laura Beckwith, Sherry Yang, and Mary Beth Rosson. 2006. Supporting end-user debugging: what do users want to know?. In Proceedings of the working conference on Advanced visual interfaces (AVI '06). ACM, New York, NY, USA, 135-142. DOI=10.1145/1133265.1133293*

[2] - *Catropa, Dayna (24 February 2013). "Big (MOOC) Data". Inside Higher Ed. Retrieved27 March 2013*.

[3] - *P. Adamopoulos, "What Makes a Great MOOC? An Interdisciplinary Analysis of Student Retention in Online Courses", ICIS 2013 Proceedings (2013) pp. 1–21 in AIS Electronic Library (AISeL)*

[4] - *"Benefits and Challenges of a MOOC". MoocGuide. 7 Jul 2011. Retrieved 4 February2013.*

[5] - *Yuan, Li; Powell, Stephen; Olivier, Bill (2014). "Beyond MOOCs: Sustainable Online Learning in Institutions". Cetis publications. Retrieved 31 January 2014.*

[6] - *Kop, Rita "The challenges to connectivist learning on open online networks: Learning experiences during a massive open online course", International Review of Research in Open and Distance Learning, Volume 12, Number 3, 2011, accessed 22 November 2011*

[7] - *S.F. John Mak, R. Williams, and J. Mackness, "Blogs and Forums as Communication and Learning Tools in a MOOC", Proceedings of the 7th International Conference on Networked Learning (2010)*