KINGSTON UNIVERSITY

Faculty of Computing, Information Systems & Mathematics

WLab: Providing E-Learning Tools for ICT Students using Virtual Machines as Learning Objects

Paul Neve

A thesis submitted in partial fulfilment of the requirements of Kingston University for the degree of MSc in Informatics

September 2010

Table of contents

List of figures and tables

Glossary of terms used

The descriptions below explain the meaning of the terms with reference to the WLab system:

Active Directory	Microsoft product that provides an electronic directory of individuals, groups and other entities.
ActiveX	A framework for creating software components that are independent of a particular programming language. Only available on Microsoft operating systems.
AD	See **Active Directory**.
Administrative privileges	User permissions that allow someone to perform administrative tasks on a server, such as reconfiguring its services, changing its network settings, adding additional software, etc.
Agile	Software development methodologies that subscribe to the Agile Manifesto [1]: "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value: **Individuals and interactions** over processes and tools **Working software** over comprehensive documentation **Customer collaboration** over contract negotiation **Responding to change** over following a plan"
Backend VM	In WLab, any virtual machine that exists on the virtualisation server. They may have been created by a system administrator, using the tools available outside of WLab, or they may be a **Tutor VM**.
Backlog document	Technique used in Scrum that documents all deliverables/features. A *product* backlog is a complete set of deliverables for the project; a *Sprint* backlog is a subset used to document those to be delivered during a given **Sprint**.
BDUF	See **Big Design Up Front**.
Big Design Up Front	Any software development methodology which subscribes to the traditional approach whereby most, if not all design work is carried out before any development takes place.
Crystal Clear	**Agile** software development methodology written by Alistair Cockburn.
Cygwin	Cygwin provides a Linux-like environment and suit of tools for Windows operating systems. See www.cygwin.com.
DCOM	Distributed Component Object Model. Provides a means of communication between Windows-based networked computers.
Dependency injection	A programming technique designed to reduce tight coupling and interdependency between software components.
DMZ	A "demilitarised zone"; in computer networking, an area of a network that is exposed to the external internet but isolated from the internal network, or allowed access to the internal network in only a highly controlled fashion.
DSDM	An **Agile** software development methodology. See www.dsdm.org.
eDirectory	Novell's solution that provides an electronic directory of individuals, groups and other entities.
eXtreme Programming	An **Agile** software development methodology.
Hyper-V	Microsoft's **virtualisation solution**.
IIS	See **Internet Information Services**.
Internet Information Services	Microsoft's web server application.
J-Interop	A **Java** library for sending **DCOM** requests.
Java	Programming language by Sun Microsystems that uses a **virtual machine** (the Java Virtual Machine or JVM) to execute applications written for it. Java applications can thus run on any platform which has a version of the JVM available.

Java servlet	A **Java**-based component for a web application. More formally, a class that conforms to the Java servlet API.
JiBX	A Java library that binds **XML** data to Java objects.
JSP	Stands for JavaServer Pages, which generates **Java servlets** dynamically using a simplified scripting language.
Jumpgate	An application that accepts traffic at a network address and forwards it to another. See http://jumpgate.sourceforge.net/.
JWBem	Provides Java classes for accessing **WMI**.
JXPlorer	An open source **LDAP** browser application written in Java. See http://jxplorer.org/.
KUOLE	Bespoke **VLE** written in-house at Kingston University.
Lab	A WLab learning object that describes and provides the environment for a practical workshop activity. Labs are divided into **lab stages**. The lab stages, taken in order, describe the path one takes to complete the lab activity and provide additional complementary learning content.
Lab stage	A WLab object that describes a milestone point of a **lab**. They include a **Tutor VM** that provides a working environment and starting point for the activities of this part of the lab, and one or more **resources**.
LDAP	The *Lightweight Directory Access Protocol.* Provides a protocol for accessing and searching an electronic directory of individuals, groups and other entities.
Learning object	“Any digital resource that can be reused to support learning” [2]
Lubuntu	A streamlined Linux distribution, based on Ubuntu, designed to work in a small memory footprint using a minimum of system resources. See www.lubuntu.net and www.ubuntu.com.
Metaphor	A technique from **eXtreme Programming** where a software system is described in a paragraph or two. “Entity metaphors” are established to give a common terminology for key objects of a system. The purpose is to establish a common lexicon across all project team members and alleviate any potential misunderstandings between (for example) developers and users.
MVC	See **Model-View-Controller**.
Model-View-Controller	A software design pattern which separates the general application and domain logic from the presentation layer, which reduces coupling and dependencies and encourages more modular code.
Mono	Open source implementation of **.NET**.
MoSCoW	System of prioritisation commonly used in software development where each deliverable and/or feature is given one of four statuses, i.e. “**M**ust have”, “**S**hould have”, “**C**ould have”, “**W**on’t have”. Ordinarily, a project is expected to deliver all M- and S-status items (although an **Agile** project may well respond to change by altering priorities throughout).
.NET	A software framework for the Windows family of operating systems that provides a set of libraries for common programming problems, and a virtual machine designed to execute programs written in the framework.
N-Learning	Rote-based, memorisation of quantifiable facts. Defined by Max Boisot in [3].
OER	See **Open Educational Resource**.
Open educational resource	A learning object or objects, made available under a distribution licence that permits free distribution and use, often distributed via a publicly accessible repository web site.
Prototype	In the context of **Agile** software development, a deliverable usually supplied at the end of a **timebox** which describes a new aspect of functionality. In early stages, prototypes might take the form of screen mockups, or a skeleton version of the application with only sample data displayed. Later, they take the form of fully operational and working software.
RDP	Microsoft’s *Remote Desktop Protocol.* Used to deliver the console of a computer to a remote terminal.

Resource	A link to a web page or other HTML content. Resources provide additional learning context during a **lab stage**, and might be used to give instructions to the student, references to other pertinent learning material such as textbooks, lecture notes, etc.
Root privileges	See **administrative privileges**.
S-Learning	Learning based on flashes of insight and leaps of comprehension; “practical”, intuitive learning such as that which takes place in a workshop environment. See Max Boisot’s work [3].
Samba	Solution for UNIX-based operating systems that provides Windows-compatible network access.
SAN	A **storage area network**, a means of attaching data storage devices to servers over a network so that they appear to the operating system on the server as locally attached devices. Several servers can access the same SAN, meaning that SANs are often used in server clusters where many servers must access the same storage/data.
Scrum	An **Agile** software development methodology. See www.scrumalliance.org.
Shovelware	Content that has been uploaded to a VLE with little consideration as to the pedagogic benefits offered by the VLE’s features, and that is often meaningless when taken out of its original context (e.g. the PowerPoint slides of a lecture minus the verbal component of the lecture itself). Defined by Teo and Gay [4].
Spring (framework)	An open-source application framework for **Java** which provides a number of solutions for common programming issues, including **dependency injection** and the **MVC** pattern.
Sprint	A **Scrum** term meaning a period of development activity focused on a defined set of goals.
SSH	A network protocol that provides a secure channel between client and server.
Student	In WLab, an individual who will perform the activities implied by **labs** for learning purposes.
Subversion	A revision control system and repository, often used for software projects.
SVN	See **Subversion**.
Sysprep	A tool, provided by Microsoft, designed to provide a “clean” instance of a Microsoft operating system as if it had been only just installed.
System administrator	An individual or group of individuals who is responsible for the installation, configuration and subsequent support of the WLab system.
Terminal Services Gateway	Microsoft’s solution for routing **RDP** traffic securely over the internet and through firewalls.
TightVNC	An implementation of **VNC**. See http://www.tightvnc.com.
Timeboxing	Technique whereby tasks are divided into pre-defined, constant periods of time. Each set period should result in a quantifiable set of outputs.
Tutor	In WLab, an individual who will design **labs**.
Tutor VM	In WLab, A virtual machine explicitly created by a tutor for use in a **lab stage**. Tutor VMs are cloned from existing VMs.
Unified Process	A software development methodology. Notable for the manner in which it spreads out all of the discrete tasks of developing an application over the period of the entire project (so, for example, development – or “implementation” – will occur in varying degrees all the way through the project).
Use cases	A technique used in software development to analyse user requirements. They describe a system’s behaviour as it responds to user requests, input and/or activity.
Virtual Learning Environment	An electronic solution that supports teaching and learning. VLEs usually operate using web technologies and provide a variety of tools for distributing learning content and facilitate communication and collaboration between tutors and students.

Virtual machine	A complete simulation of a computer system, including its hardware, software, user data and configuration state. The computer system might be a simulation of a real, physical machine (e.g. the simulations of desktop PCs provided by **virtualisation solutions**) or it might be a machine that has no physical analogue, such as those used to provide execution platform for programs written for them, e.g. the Common Language Runtime in .NET [5] or the Java Virtual Machine [6]. In WLab, a virtual machine that simulates a desktop PC is used to describe an entire computer environment for a given **lab stage**.
Virtual PC	A Microsoft product for the running **virtual machines**.
Virtualisation	The technology of running and delivering **virtual machines**.
Virtualisation server	A server which has the ability to run and deliver **virtual machines** to users.
Virtualisation solution	A product which runs on a server in order to make it a **virtualisation server**. Examples are **Hyper-V** or the various products offered by **VMWare**.
VLab	A previous project at Kingston University that delivered a virtualised lab environment.
VLE	See **virtual learning environment.**
VM	See **virtual machine**.
VMWare	A software company and/or a suite of products by that company that deliver **virtualisation solutions.** Also used as a catch all term to mean "any one of the company's virtualisation solutions".
VNC	*Virtual Network Computing*. An open-source solution, originally written by staff at Olivetti and later at AT&T, which can deliver the console of a computer to a remote terminal.
Windows Management Instrumentation	An interface for performing management tasks on Microsoft Windows based computers through scripts and/or code.
WMI	See **Windows Management Instrumentation**.
wsname	A tool written by David Clarke (see http://mystuff.clarke.co.nz/MyStuff/wsname.asp) that provides a means of changing the computer name of a Windows-based operating system via a script.
XML	eXtensible Markup Language. Provides a standard means of encoding documents and data in a text-based form that can easily be read by software and (to some degree) by humans.
XML Schema	Defines the structure, semantics and the data stored within an XML document.
XP	See **eXtreme Programming.** Or, alternatively, might refer to Windows XP.

Abstract

When teaching ICT subjects, active learning is a crucial component: students must have the opportunity to engage in hands-on workshops to be able to put what they learn into practice. Delivering these workshops to distance learning students is problematic in that they do not have access to institutional equipment and thus the requisite computer environment may not be available to them. Even when delivering workshops to students on-campus, issues may arise. Institutional equipment is usually configured to a generic "one size fits all" template designed for simple office-type tasks. This may be ill suited to the needs of ICT teaching, particularly for advanced courses where students may also need "root" or administrative access to servers.

Virtualisation is frequently used to overcome these issues, and allows for the delivery of a simulated computer environment to students either on- or off-campus without the difficulties involved in modifying "real" institutional equipment. Several projects implement such simulated environments for ICT teaching; however, thus far such projects have been very narrow in scope, providing only for the needs of a specific domain, e.g. IT security. Additionally, they fail to avail themselves of the pedagogic opportunities inherent in the characteristics of virtual machines.

WLab introduces a new approach to teaching via virtualisation for ICT. At its core is an innovative composite learning object. A lab is divided into a series of **stages**. Each stage contains a **virtual machine** state, and one or more links to static learning content objects (**resources**). These stages can be considered milestone points along the student's activity path through an exercise. The virtual machine in each stage provides both the working environment and the initial starting point for the activities of the stage. The resources provide a learning context, and link the practical aspect of the lab stage with the teaching goals of the activity, by referencing other appropriate learning materials, previous workshop stages, etc. Each stage builds on and introduces new techniques and learning content.

A hybrid approach that borrows techniques from several different Agile methodologies was used in order to accommodate the needs of a one-developer project. Open technologies were used during development wherever possible to maximise accessibility and minimise vendor lock-in. The result is a cross-platform web application that both delivers the composite learning object to students, and provides a user-friendly authoring environment for tutors.

The logistical and pedagogic advantages of such a system are significant. Practical workshops for ICT courses are liberated from the limitations imposed by distance, scheduling and outdated facilities. Students can quickly contrast and compare their work with either a past or future stage, a learning technique not easily replicated in a conventional on-campus workshop scenario. Workshops can be distributed to other tutors and/or institutions in a fully self-contained form. The concept of an ICT practical workshop exercise as a complete, portable open educational resource is now a reality.

Acknowledgements

The author would like to extend profound gratitude to the following people:

David Livingstone
for his support throughout not only this work, but the entire MSc programme

Luke Hebbes
for his support throughout this work, and for providing an alternative technical perspective

Graham Alsop
for assistance in negotiating the world of academia, and pointers on pedagogy

Adam Hobbs and all of the CISM IT support staff at KU
for going above and beyond to provide technical resources and assistance

Miroslav Novak
for his kind understanding during the family emergency that delayed this work

and most importantly

Jean and Bob Neve
without whose support and love it would have been absolutely impossible for the author to successfully undertake an MSc programme

1 Introduction and background

1.1 Why electronic delivery of ICT courses is problematic

In many academic environments, virtual learning environments (VLEs) provide an electronic means of delivering handout-style learning content[1] to students, e.g. PDFs, Word documents, HTML pages etc. Many VLEs provide interactive facilities such as forums, instant messaging, wikis, and so on; the intent is to provide an electronic analogue of the usual two-way student-tutor dynamic, where a student increases their understanding of the static material delivered by posing questions. Additionally, VLEs are often used to deliver learning content to distance-learning students.

Unfortunately, VLEs are rarely used to best effect, with many tutors using the VLE solely as a means of distributing lecture notes. Others feel obligated to populate their area of the VLE with *something* even if they aren't comfortable with the technology, and simply refactor existing static content with little or no consideration of any pedagogic advantages offered by the software – Teo and Gay refer to this as "shovelware" [4]. Often, such shovelware takes the form of PointPoint slides originally intended to complement the verbal element delivered by the tutor during a lecture. However, Williams' point that a learning object only has useful meaning if delivered in an appropriate context [7] is particularly pertinent. Shorn of the verbal component the slides are less useful and in extreme cases may even be meaningless. (This issue is discussed further in *A General Approach to E-Learning for ICT Students* [8].)

Even if a tutor is able to completely approximate a lecture via a VLE – perhaps by including not only PowerPoint slides, but also an audio component or other replacement for the spoken word aspect – it would not overcome the fundamental shortcomings of lecturing as a teaching method. It has been established that students' attention spans during lectures waver after a relatively short time [9]. Most best practice guides now advocate the inclusion of "active learning", defined by Bonwell and Eison [10] as "instructional activities involving students in *doing* things and *thinking* about what they are doing" (our emphasis).

Boisot distinguishes between two types of learning, Neoclassical or Newtonian learning (N-Learning) and Schumpterian learning (S-Learning) [3]. Mellor and Mellor's simplification of Boiset's wordy discourse summarises by stating that N-Learning consists of rote-based memorisation of quantifiable facts, whereas S-Learning involves flashes of insight and leaps of comprehension [11]. Both kinds of learning are important, but S-Learning is key in practical subjects like ICT if a student is to be able to successfully apply the materials being taught to create new, original pieces of work. Lecture style teaching conveys N-Learning easily, but is less suitable at promoting S-Learning where other approaches might be more appropriate.

One such approach is to use workshops, where a student is set a practical activity designed to exercise the skills they have acquired. Unfortunately, with respect to ICT courses, VLEs cannot deliver an analogue of an on-site workshop where a student would sit at an institutional computer in a lab and attempt a practical task. This presents a serious problem for distance learning students. In many cases it is assumed that such students have access to IT equipment, and that this equipment can be configured according to the needs of the workshop exercise. These assumptions can lead to support issues that are highly distracting for the student. If the student does not possess or cannot acquire suitable equipment, they may miss essential components of the course.

Beyond those posed by distance learning, ICT also presents other problems when it comes to running workshops for more advanced courses. In many such cases, administrative or "root" level privileges on a server are required in order for a student to practice the skills being taught. Students cannot be given such privileges on a production server, and it would be infeasible to allocate each student a server of their own.

Gaspar et al [12] describes the creation of isolated labs containing workstations with full administrative privileges, so as to allow students the root-level access they need for their studies without impacting the rest of the campus network. Such an approach requires significant

[1] Such learning content will be referred to as "static" content from henceforth.

resources. If such a lab were established as a permanent fixture at an institution, it would mean that the facilities go unused and thus wasted when such courses are not being run. Few institutions would tolerate or indeed, be able to afford such waste. The alternative is that an existing lab room is used. Prior to the course, the laboratory would need to be isolated from the institutional network, and all its workstations reconfigured. At the conclusion of the course, a similar exercise would be required to reverse the changes. This would require considerable staff time, effort and resources.

The ideal solution to these issues would be a VLE that, alongside the other learning content being delivered to the student, provides a simulation of a workstation or server. Such a simulation would be configurable to suit the needs of a particular workshop exercise without affecting “real” equipment or facilities. Assuming that this was being delivered via a standard web-browser, as with existing VLEs, this would overcome both on-site logistical issues and those presented by distance learning students. While no such idealised VLE yet exists, many academic institutions have made use of virtualisation to provide these simulations.

1.2 Introducing virtualisation

There is a perception that virtualisation is a relatively "new" technology. However, as far back as 1974, Popek and Goldberg defined a virtual machine (or VM) as "an efficient, isolated duplicate of the real machine” [13]. Their subsequent assertions will be familiar to anyone familiar with modern virtualisation: the environment provided should be "essentially identical" to the original machine and programs running in the VM should suffer only minor performance degradation. Such principles had already been implemented a decade before [14] in the form of the IBM S/360-67 and the CP-40. At the time, running concurrent applications using time-sharing was beginning to gain momentum, but the S/360-67 was the first hardware to provide a true, virtualised environment.

Shortly after the turn of the century VMWare released the first versions of their server-side products [15]. These enabled organisations to reduce the amount of physical servers required to run their infrastructure by allowing a single piece of hardware to run multiple virtual "servers". Competitor products to VMWare opened the marketplace further, and it is now rare to find an organisation with a large IT investment that does not make some use of virtualisation. This would seem to be supported by the heavy growth in sales of virtualisation solutions up until 2008 [16] at which point sales began levelling out. (One might surmise, as server virtualisation became more commonplace, with a larger install base, the opportunities represented by new sales were reduced to a minimum, and subsequent sales were comprised primarily of upgrades for existing customers).

1.3 Virtualisation in education

The education sector is no exception to these trends, and a white paper by IBM [17] describes the growth of virtualisation in education. However, it is fair to say that the case studies cited are only indirectly related to pedagogy. Of more interest are projects that directly use virtualisation to deliver learning content, and that move towards the proposed idealised VLE for ICT.

In 2002, with virtualisation on desktop PCs still in its relative infancy, McEwan discussed how Christchurch Polytechnic Institute of Technology was able to use a combination of products to deliver operating system- and network-related subjects [18]. VMWare Workstation was used for the OS courses, where the need was simply to be able to quickly and easily switch between different OSes. The solution for the networking courses was more novel, where User Mode Linux or uml[2] [19] was used to create a virtual network. This network never actually existed; it was comprised of virtual hosts running on the same physical hardware. However, it allowed a lecturer to demonstrate the concepts to students, and allowed students to experiment with various aspects of networking, such as subnetting, routing, firewalls and so on. Its virtual nature also meant that it did not run the risk of causing damage to the production network, through error or malice on a student (or lecturer's!) part. In 2006, Bullers et al used a similar model to

[2] The acronym for User Mode Linux is not capitalised when written, apparently to avoid confusion with the Unified Modelling Language.

McEwan to deliver “three advanced courses covering system and network administration, information security and database administration” [20].

One commonality between Bullers’ and McEwan’s work is that in both instances, the virtualisation products were used solely as a means of solving one of the logistical problems in advanced ICT pedagogy, i.e. that it is both physically and financially impractical to give every student their own server and workstation machines. Xen Worlds [21] has a similar infrastructural focus. In contrast, V-NetLab [22] takes a step further forward. A web interface is provided from which students can create and connect to a virtual network comprised of several virtual machines. The VMs that comprise the virtual network are created on demand from a base image, prepared in advance by the tutor.

1.4 Virtual machines as learning objects

V-NetLab’s base VM images might be considered a learning object in their own right. Consider: these images contain information, both overtly in the form of discrete files and folders stored within the VM’s virtual hard disk, and also more subtly in the form of the VM’s configuration, the operating system environment and so on. Wiley defines a learning object as “any digital resource that can be reused to support learning” [2]. V-NetLab’s base VM images certainly fit this definition.

Kingston University has used virtualisation as a vehicle for delivering such learning objects for some years now. Initially, a McEwan-like approach was used where virtual machines were prepared in advance of a course, and Microsoft Virtual PC used to run them. A student could use an institutional machine, or take the VM image home on an external hard drive and install and run Virtual PC on their own equipment. Later, the VLab project centralised the VM learning objects onto a Hyper-V server and provided a student-facing web interface. Students were allocated a VM on this server, and a web interface allowed them to stop and start their VM. Connectivity both on campus and at remote sites was provided via Microsoft Terminal Services gateway. VLab provided no ability for VMs to be created on demand, meaning tutors were required to both design an initial exemplar image then manually create clones of it for each student on the course. However, as with V-NetLab, these exemplar images can be considered learning objects.

As noted previously, a learning object must be presented in an appropriate context in order to be meaningful. Virtual machine-based learning objects are no exception. With regard to VLab this will be illustrated by way of an example, the Electronic Commerce Technologies MSc module delivered at Kingston University in December 2008. Via VLab, each student was provided with a virtual machine containing Windows Server 2008, IIS, MS SQL Server and Visual Studio. The context of the VM was established by conventional lectures in combination with printed handouts (which were also available in electronic form). During a workshop session, the handout described the practical activities that the student was expected to engage in within the VM. These activities were designed to reinforce the learning context delivered by previous lectures.

The SOFTICE project leverages virtualisation to deliver computer security courses [12, 22]. As is the case with VLab, its authors acknowledge the importance of accompanying static resources if the virtualised environment is to have any meaning as a learning object. On SOFTICE’s project wiki there is an area devoted to “pedagogical resources” [24]; these resources take the form of “worksheet”-type documents.

In both the case of VLab and SOFTICE the association between virtual machine and static learning resources is manual. The current use of VLab assumes that a student will be given or told where to find these resources – the VLab system itself does not provide any link or reference, although it would be possible to include the materials within a VLab virtual machine or host them on the VLab web server itself. Similarly, with SOFTICE, a student wishing to use the published “pedagogical resources” must use a separate web browser to locate and view them. In both cases, the solution falls short of the ideal outlined previously, where the VLE itself would deliver both the practical workshop environment and the static learning materials side-by-side.

Within their target subject area (i.e. IT security) the SOFTICE team have engaged in considerable discussion about the pedagogic advantages of virtual machines as learning objects

[12, 22]. In the latter source, they introduce the concept of dividing a lab assignment into steps: a "briefing" step establishes a base level of knowledge, an "exercises" step defines what is expected of the student, and a "solved" step gives exemplars. SOFTICE's lab assignment is thus a collection of these steps along with complementary "pedagogic resources" as exemplified on their wiki. A single step in isolation – which one assumes includes a pre-defined virtual machine state, although this is not explicitly stated – is meaningless without the accompanying pedagogic resource that guides the student in its use and places it into context. Indeed, a step further might be to consider the lab assignment – i.e. the collection of "briefing", "exercises" and "solved" steps – as a single learning object.

One shortcoming in this approach is that it does not consider *progression*. The steps are presented as *faits accomplis* that are only indirectly related. For example, [22] describes the "solved" step as "an example of the kind of work the student is expected to do". Thus it is *not* the solution to the "exercises" step suggested by its name; rather it provides exemplars of *similar* work intended to direct and give students a clue to the right direction. Referring to these as "steps" is something of a misnomer, as they do not represent a logical progression of a learning task from start to finish.

An alternative approach would thus be to make the individual "steps" precisely that – i.e. steps one takes when performing a task from start to finish. The benefit of this approach is that the resulting learning object effectively becomes a "roadmap" that represents the path taken by an expert user in executing the activity. A novice can follow this roadmap and in doing so, assimilate the tacit knowledge inherent in the expert's execution of the activity.

1.5 Introducing WLab: a new paradigm for a virtualised ICT lab environment

In [8] a composite learning object was described that would – among other things – contain several discrete virtual machine states. These states or "stages" taken in a particular order, would describe a path that a student would take in order to complete a lab activity:

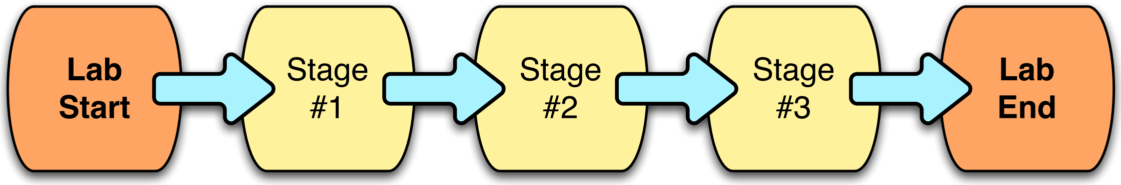


Figure 1 - the path of a lab exercise

Thus, the idealised VLE for ICT learning postulated previously can now be extended to one that not only presents a simulation of a practical environment alongside its associated static learning content, but one that implements this staged path through a lab exercise. Such a VLE would implement a composite learning object as shown in Figure 2:

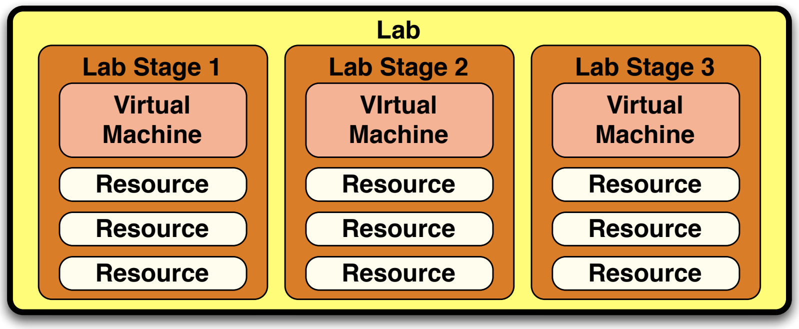


Figure 2 - the structure of a 3-stage lab

Here, a "lab" can be defined as any given workshop activity. This activity is comprised of "stages" which contain a virtual machine and one or more "resources". The virtual machine would be pre-configured to provide the starting point for the student to begin the practical work required of them for this stage of the exercise. The "resources" are accompanying, context-setting learning content, provided via PDFs, Word documents and/or links to HTML content. The *differences* between the stages describe additional learning content – in many cases, this will be the path one takes in order to navigate the activity from start to finish. Nothing precludes adopting a non-deterministic approach when designing such labs, where one stage's end point is not necessarily the next stage's start point (a la SOFTICE's approach). However, the advantage of the deterministic flow described in Figure 1 is that it simulates a real-world task more closely. This means that the tutor could author a lab by simply *doing* the task, and, from the student's perspective, it provides both a complete exemplar and interactive step-by-step guide to achieving a desired result.

The WLab[3] project will implement such a composite learning object, providing the requisite tools for tutors to author them, and for their delivery to students. This represents a natural evolution of the previous work at Kingston University in ICT workshop delivery, incorporating the lessons learned from past work as a foundation for what follows. Thus while in many respects this is a brand new project, it also represents a continuance of previous work in ICT workshop delivery. This is illustrated in Figure 3:

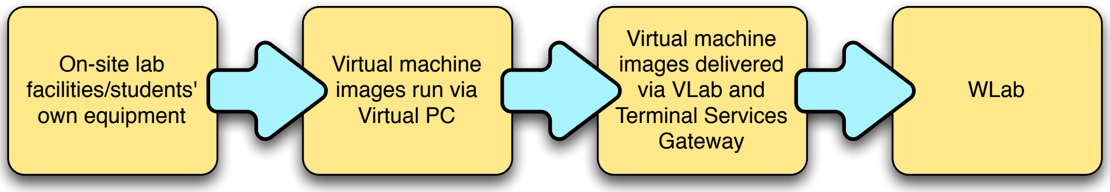

On-site lab facilities/students' own equipment
Virtual machine images run via Virtual PC
Virtual machine images delivered via VLab and Terminal Services Gateway
WLab

Figure 3 - the evolution of ICT workshop delivery at Kingston University

[3] WLab – derived from V++Lab – i.e. the next incremental step!

2 Project preparation

2.1 The kick-off workshop

A Kick-Off Workshop was held on Thursday 6th May, with the following individuals present:

Luke Hebbes – user and product owner
David Livingstone – user and project supervisor
Paul Neve – developer and project manager

The concept of the staged lab exercise was discussed in depth and agreed to be highly desirable.

A number of technical shortcomings were identified with the existing VLab solution. Firstly, creating virtual machines is a manual process; for a class of 20 students, 20 VMs must be manually configured. Tools such as Microsoft System Center Virtual Machine Manager can simplify this process [25], but it is still a non-trivial task. Secondly, VLab makes use of predominantly Microsoft technologies; Hyper-V is used as the virtualisation backend, the student-facing web interface that allows them to start and suspend their lab sessions is a .NET application hosted on IIS, and the students' VMs are delivered via Microsoft Terminal Services Gateway. This combination provides a means for both remote and on-campus students to access their lab sessions – but only if they are running a Windows operating system. This presents a serious accessibility issue.

On the current VLab system, in order to minimise server load, students are prevented from starting their VMs if the number in use by other students exceeds a defined maximum. Additionally, so as to prevent students from "hogging" the system, they are given a limited duration during which their virtual machine may be run. After this period, their virtual machine is automatically suspended and the student cannot restart it until a set time has elapsed. However, this means that students may have difficulties logging in at busy times. Consequently, the ability for a student to schedule a guaranteed period of lab time via some form of booking system was seen as key.

The original intent in the project proposal was to integrate with the KUOLE VLE, developed in-house at Kingston University. However, at the kick-off workshop this was de-scoped. It was agreed that introducing dependencies on third parties and other projects introduced an unpredictable complication, especially as KUOLE is now a PhD project. Instead, a simple non-functional requirement was introduced: WLab data should be stored as simple XML according to a defined schema. This will make any future integration a very simple matter; the KUOLE developer can simply utilise the schema to immediately start reading and writing WLab data.

2.2 Project aims

Based on these discussions during the kick-off workshop, the following aims were defined:

- Implement the composite learning object previously described, i.e:
    - A lab exercise contains several stages
    - Each stage contains a virtual machine state which provides the practical environment of this stage of the lab exercise
    - Each stage also contains associated static learning resources intended to complement and place the practical environment into an appropriate context
- Provide for automatic provisioning of virtual machines to students
- Provide a web-based application that delivers these learning objects to students
- Provide a web-based application that allows tutors to author these learning objects.
- Provide a facility for students to book a time slot where access to the system is guaranteed
- Provide a solution that is cross platform – both in terms of client access, and in terms of the server side application itself.

Another point to consider is the importance of providing a solution that is genuinely usable in the real world, and that is accessible to the uninitiated. Much of the literature describing other virtual lab environments has targeted the research community rather than a potential user community, and as a result most projects have struggled to acquire a user base outside of their authoring institutions. SOFTICE is an exception in that their project wiki [24] does provide material targeted at individuals wishing to actually *use* the solution (e.g. as installation details, virtual machine images and so on). However, the relative inactivity on the SOFTICE wiki (the last updates took place in November 2008) suggests that it has fallen into disuse.

Thus, the final aim of the project will be to:

- Provide instructional materials targeted at users wishing to deploy and use the system. The following categories of users can be anticipated:
    - Students
    - Tutors
    - System Administrators
    - Developers
- Provide an on-line presence for the project which can serve as a foundation for a user future community for collaboration and knowledge sharing.

2.3 Project planning

2.3.1 The five development phases

At the initial kick-off meeting, it was agreed to take a five-phase approach to development. Each individual phase provides a complete, coherent solution in its own right that improves upon what existed before it, and provides institutional value. The ideal is to implement all of the phases before the end of the project. However, if this is not possible, the staged approach will still produce worthwhile ouputs.

The proposed development phases are as follows:

Phase 1	Reverse engineer current VLab functionality using open technologies
Phase 2	Implement scheduling/booking functions
Phase 3	Dynamic virtual machine creation: • When a student logs in and attempts to undertake a lab exercise, if required, the system will automatically create a new virtual machine for them from a "base" image.
Phase 4	Introduce the composite learning object: • Introduce the concept of a "lab" that consists of multiple virtual machine objects with complementary URI resources • Enhance student-facing UI to allow comparison and synchronisation (i.e. compare their current work with the exemplar VM state, or sync themselves with the exemplar if they are stuck)
Phase 5	Management interface for tutors: • Additional web interface to be created to allow tutors/administrators to create lab exercises via a user-friendly web-based UI (the assumption is that, up until this point, such authoring will have been done via the virtualisation backend's own tools, and by editing appropriate XML files to create the WLab objects required)

Table 1 – the five phases of development

It should be noted that the above describes only the phases anticipated for *this* specific project and period of development effort. Future work may go beyond these phases.

2.3.2 Project scheduling

There are a number of external dates and deadlines that influence project planning:

1st July	**Interim report due**
6th July	**Poster Presentation**
23rd September	**Final due date**

Additionally, a proposal for a poster presentation has been submitted and accepted for the ALT-C 2010 Conference (see http://www.alt.ac.uk/altc2010/). This introduces another date:

7th-9th September	**ALT-C**

The project schedule is shown in Figure 4; this has been derived from the requirements of the hybrid agile approach (see 3.1), the deliverables required for the dates outlined above, and the outputs of the project kick-off workshop.

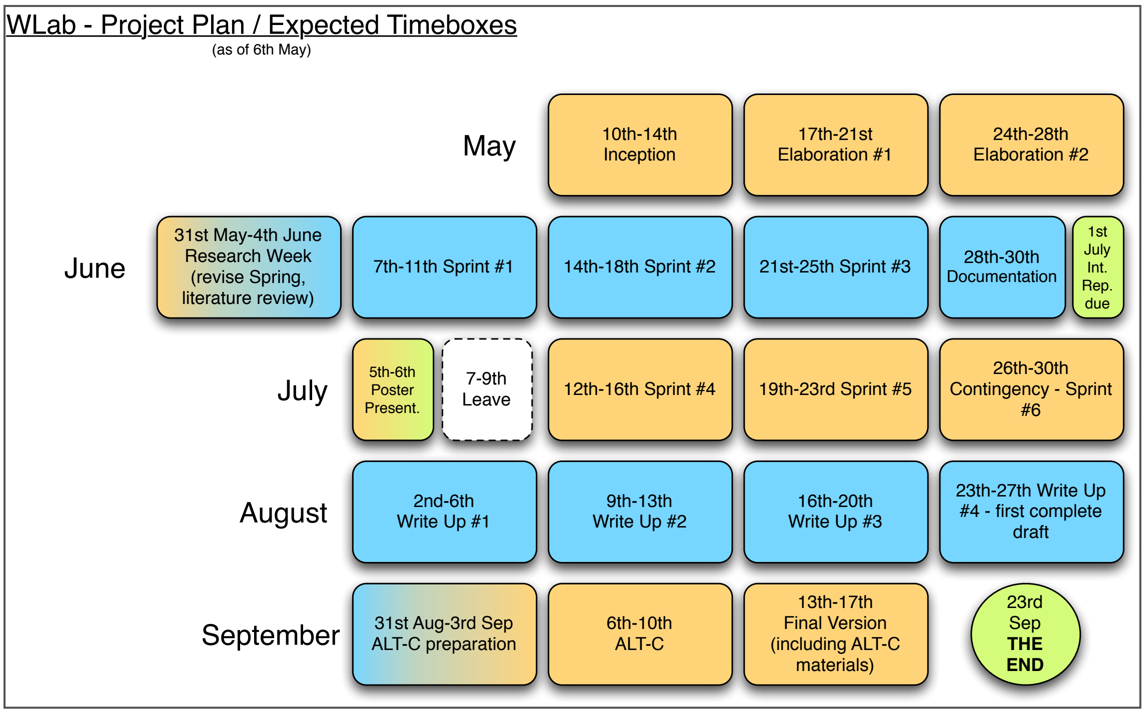


Figure 4 – project schedule

3 Methodology

3.1 Development and Project Management Methodology

The team working on this project is very small – i.e. one developer, two users. The developer will be expected to handle most if not all project management and the users will fulfil what in formal methodologies would be considered business-level roles, e.g. product owner.

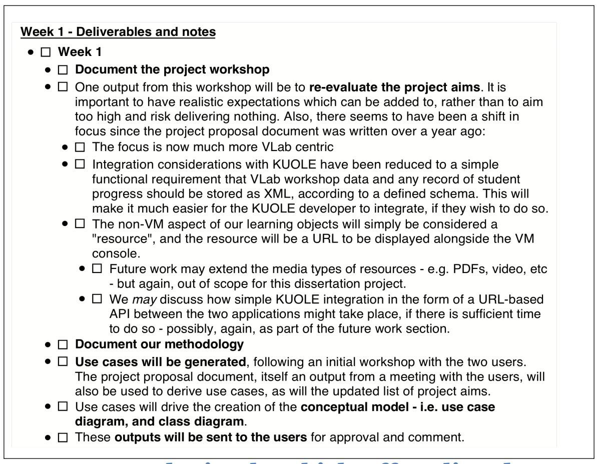
This small team would seem to lend itself to an agile approach, and there is a wealth of literature that suggests that agile methodologies work best with small teams, although perhaps not quite as small as this. Alistair Cockburn's Crystal Clear suggests a team size of 8 or less, "four of which probably have to be distinct people [26]. Scrum defines three distinct roles, "but the team in Scrum is seven, plus or minus two people" [27]. DSDM Atern is even more heavyweight, with a raft of artefacts suggested by its product set [28] and 12 distinct roles [29]. While a single individual could conceivably and is often expected to perform several roles in such methodologies, no established Agile methodology seems precisely suited to such a minimally-staffed project.

The intention is therefore to cherry-pick various techniques and principles from different approaches, adapting them where necessary for the purposes of this project.

3.1.1 Technique #1: Timeboxing (from DSDM Atern)

Atern's concept of timeboxing will be crucial as a means of task and time management [30] There are 19 weeks available to the project; this is a fixed, immutable period that cannot be altered. This time will therefore be divided into 19 week-length timeboxes. Not all of these timeboxes will be devoted to actual software development; indeed, it is anticipated that a larger number will be concerned with what might be referred to the more "prosaic" aspects of an MSc dissertation project.

Atern indicates that a timebox should be divided into five stages – **kick-off**, **investigation**, **refinement**, **consolidation** and **close out**. While the formality of such clearly defined stages within the timebox is probably excessive here, given the single developer involved, the expectation is that analogues of the **kick-off** and **close-out** stages will occur. At the start of each week, while nothing so formal as a meeting will occur, an hour or so will be devoted to producing an outline document detailing the expected week's tasks and deliverables. An example is shown in Figure 5. Similarly, the final activity of the week will be to evaluate the deliverables actually produced during the timebox and compare them against those anticipated in the outline document. This approximates the formal acceptances mandated by Atern.


Week 1 - Deliverables and notes

- ☐ **Week 1**
  - ☐ **Document the project workshop**
  - ☐ One output from this workshop will be to **re-evaluate the project aims**. It is important to have realistic expectations which can be added to, rather than to aim too high and risk delivering nothing. Also, there seems to have been a shift in focus since the project proposal document was written over a year ago:
    - ☐ The focus is now much more VLab centric
    - ☐ Integration considerations with KUOLE have been reduced to a simple functional requirement that VLab workshop data and any record of student progress should be stored as XML, according to a defined schema. This will make it much easier for the KUOLE developer to integrate, if they wish to do so.
    - ☐ The non-VM aspect of our learning objects will simply be considered a "resource", and the resource will be a URL to be displayed alongside the VM console.
      - ☐ Future work may extend the media types of resources - e.g. PDFs, video, etc - but again, out of scope for this dissertation project.
      - ☐ We *may* discuss how simple KUOLE integration in the form of a URL-based API between the two applications might take place, if there is sufficient time to do so - possibly, again, as part of the future work section.
  - ☐ **Document our methodology**
  - ☐ **Use cases will be generated**, following an initial workshop with the two users. The project proposal document, itself an output from a meeting with the users, will also be used to derive use cases, as will the updated list of project aims.
  - ☐ Use cases will drive the creation of the **conceptual model - i.e. use case diagram, and class diagram**.
  - ☐ These **outputs will be sent to the users** for approval and comment.

Figure 5 – sample timebox kick-off outline document

3.1.2 Technique #2: The "Inception" and "Elaboration" project phases (from the Unified Process)

The Unified Process – regardless of which flavour is chosen – names the initial project phases **inception** and **elaboration** [31, 32]. It also eschews the traditional "Big Design Up Front" (BDUF) paradigm and suggests that the activities of software development straddle an entire project. Figure 6, taken from [32], illustrates this; for example, some system implementation activity takes place as early as the inception phase, and similarly, system design continues right up until the end of the project (albeit in smaller amounts).

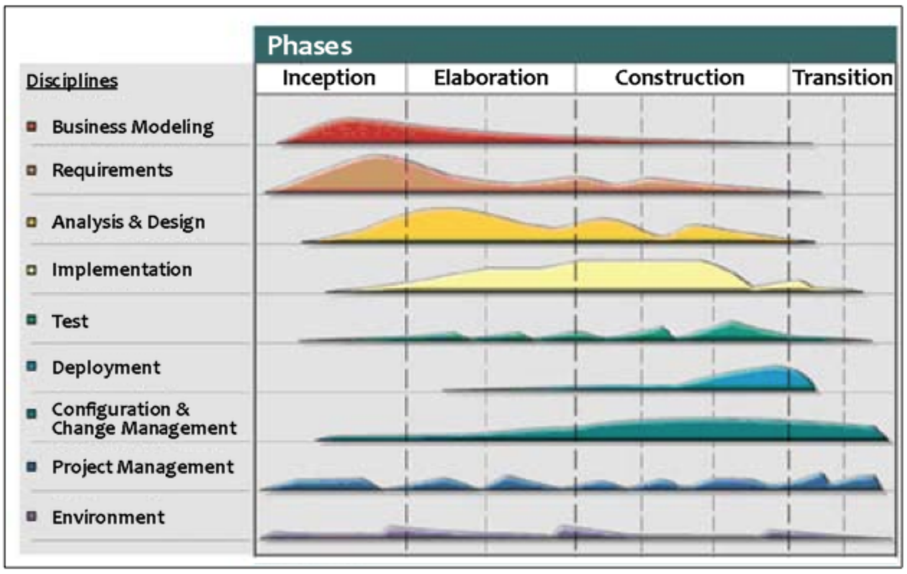


Figure 6 – the Unified Process

Nevertheless, the lion's share of modelling and design activity takes place in the early stages of the project. The intent is to adopt a similar approach here. The first timebox will be devoted to the inception phase, and the subsequent two timeboxes the elaboration phase. During the inception phase the following activities are expected:

- Finalise methodology
- Document use cases, based on kick-off workshop and other available information. These will be prioritised according to the MoSCoW system [33, 34].
- Produce conceptual model. This will consist of a skeleton class diagram and use case diagrams.

and during elaboration:

- Produce first "prototype" in the form of mockup screenshots
- Produce first draft of design model (i.e. complete class diagram with all attributes and relationships, at least for the data model)
- Produce product backlog document

The "prototype" and design model will each have two iterations, allowing the users to influence proceedings correct any early misconceptions. A product backlog document *a la* Scrum [27] will be prepared during the final days of the elaboration stage,; this will be used later to help order the development cycle.

It may appear that a large amount of design is being done up front, drifting very close to the dreaded BDUF. However, it should be noted that the design model that emerges from the elaboration phase will be used to create an XML schema, itself an implementation artefact. Similarly, the "prototype" of mockup screenshots is a form of implementation, establishing parameters for the final application's UI. Such mockups are recommended as worthwhile artefacts in a number of different Agile methodologies [35, 36]. Therefore the approach does indeed fit the overlapping activities mandated by the Unified Process.

3.1.3 Technique #3 – Metaphors (from Xtreme Programming)

XP projects utilise a technique which Beck describes as a "single overarching metaphor" [37]. In this project, this will take the form of several paragraphs of descriptive prose with certain terms – "entity metaphors" – highlighted in bold. The intention behind such metaphors is to define a common language and overview of the application that can be understood by all project participants regardless of their technical inclinations. The key outcome of this technique is to ensure that any terminology used is consistent across the project, and that all project team member understand a given term as meaning the same thing. Technical entities of the project should take their names and relationships directly from the metaphor, so as to ensure that the eventual system architecture will reflect the metaphor and thus match the users' understanding of the ideal end result.

Ordinarily, a project would have only a *single* metaphor. In this project, given the phased approach to development and the fact that the application's functions will change considerably with each passing phase, the intent is to have a distinct metaphor for each development phase. The final version of these metaphors can be found in Appendix A.

3.1.4 Technique #4 – Use cases

Use cases as a tool for requirements analysis is commonplace in Agile methods. However, the manner in which one might compose a user case varies greatly. In *Writing Effective Use-Cases* [38] Cockburn notes the difference between "fully dressed" use cases and more casual approaches. While doing so, he warns against the dangers inherent in over-elaborating and imposing too much formality on the use case composition process. In his own Agile methodology, Crystal Clear, Cockburn takes his own advice and uses the casual, two paragraph use case format [39].

In this project, the original intent was to use Crystal Clear-style two paragraph use cases, but early on in the Inception phase it became clear that the results lacked clarity. Thus, the final template used employs indented series of bullet points to describe processes and behaviour. The intent was to avoid the over-formality of Cockburn's "fully dressed" use cases, avoid the "wooliness" of prosaic paragraph blocks, while still providing the same level of simplicity as Cockburn's "casual" template. Some aspects of fully dressed use cases will be used, where appropriate – for example, to describe alternative process flows, error conditions and the like.

3.1.5 Technique #5 – iterative development

A core principle of Agile methods [27, 40, 41] is iterative development. The terminology differs across methods, but the same basic process is entailed: developers work for brief periods of time (e.g. a "sprint" in Scrum) at the end of which a prototype is delivered. The user community then offers feedback on the prototype; this feedback in turn shapes the next period of development. This means the software is subject to an evolution-like process, where over the course of the iterations bugs and incorrectly designed functionality are "weeded out" until, at the end of the cycle of iterations, a software artefact emerges that is very much in line with the users' needs.

In order to distinguish a timebox where development will take place, the Scrum terminology *sprint* will be used. In all, six of these are anticipated. The five stages of development outlined in 2.3.1 will therefore be the subject of one sprint each. A sixth sprint is allocated for contingency, mop up, or possibly even further development beyond the core features that were prioritised M or S in the MoSCoW system.

These development timeboxes or sprints will also introduce a number of Scrum-specific techniques. As part of the Elaboration process, a product backlog will have been created from the use cases. During the kick-off session of the timebox/sprint, a selection of items from the product backlog will be used to compose a Sprint Backlog document. This will be influenced by the priorities of the outstanding product backlog items. However, it will also be influenced by the staged approach of development outlined in 2.3.1 and the need to provide a coherent feature set at each stage.

3.1.6 Technique #6 – Information Radiators (from Cockburn) / Big Visible Charts (from XP)

The idea of finding a prominent wall or walls in the area where a project team works and covering it with copious project documentation is a recurring concept in Agile approaches. Cockburn refers to this as an Information Radiator [42] and notes that they should be big, very easy to see and show the reader "information that they care about". XP also has a similar concept in the shape of the Big Visible Chart, and Ron Jeffries gives a similar recommendation in [43]: "chart what you care about, what you worry about, what you want other people to know".

In a project where a single person will conduct the majority of day-to-day activity, the inter-personal communication aspect of an information radiator is less useful. However, self-communication is important. The prominence of the information radiator which means everything of importance is constantly in one's visual field helps ensures that work and deadlines stay on track.

Figure 7 shows the state of the information radiator at the end of the development cycle of the project. Each week length timebox is represented by one A5 sheet. Post it notes – colour-coded by duration – represent the tasks for the timebox. As a task is completed, it is crossed off. Backlog documents [27] and other artefacts such as use cases are also placed on the radiator and elements crossed off by hand as they are completed:

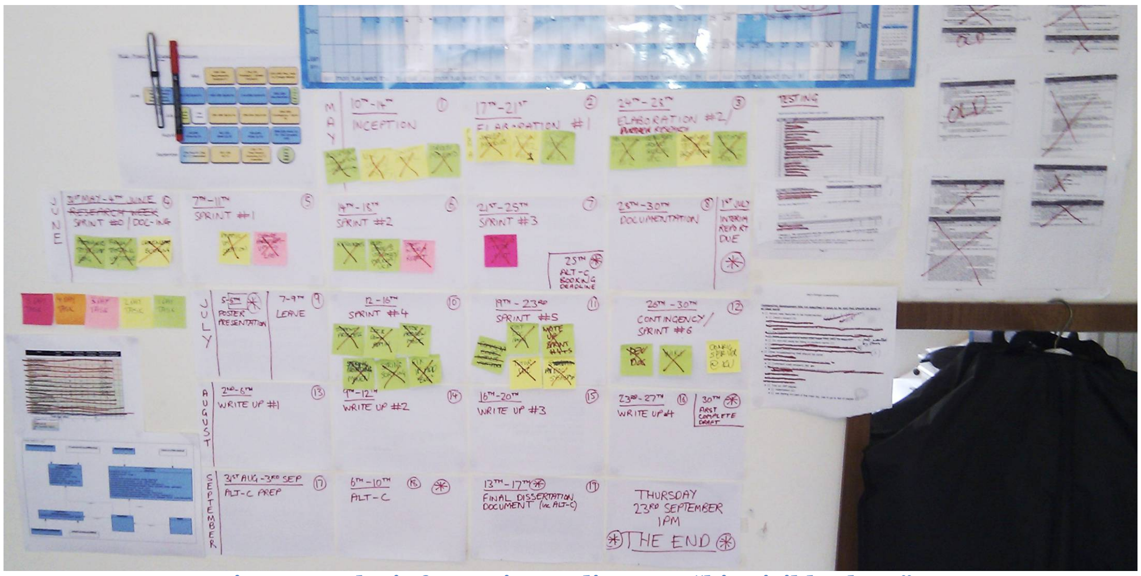


Figure 7 – the information radiator or "big visible chart"

3.1.7 Technique #7 – "Deliverables may change but timescales are fixed"

A key Agile concept is that while the target deliverables might evolve over a project lifecycle, timescales may not [44, 45, 46]. This is inherent in the phased development approach. While the objective is the completion of all five phases, one must be pragmatic and anticipate that the possibility of difficulties or obstacles that make that impossible. Consequently, it is expected that the list of project deliverables will evolve. One or more project phases may need to be re-classified as a MoSCoW "won't have" and relegated to the list for future post-project work. Equally, on the other side of the spectrum, new requirements may be introduced or existing ones altered. However, the common factor in all of this is that the overall timescale is fixed, and final delivery will take place at a set, immutable date.

3.2 Selection of technologies

For the most part, the technologies used in this project were dictated by non-functional requirements. The following non-functional requirements have already been either explicitly articulated or implied:

- The system must be accessible from both Microsoft and non-Microsoft clients
- The system must be able to present a virtual machine console within a web browser, alongside other content[4]
- System data must be stored as XML files, conforming to a clearly defined schema

The following non-functional requirements are dictated either by functionality in the existing VLab system, the manner in which it is currently used, or by the infrastructure available at Kingston University:

- Microsoft Hyper-V is the target virtualisation backend
- The system must be accessible both within the institution and remotely
- Authentication will be done against an LDAP-based directory service, which may or may not be Microsoft Active Directory
- The system must be able to deliver both Windows and Linux virtual machines

The following is seen as highly desirable by the author:

- The server-side application must not be restricted to a specific server operating system

Each of these non-functional requirements was analysed; the results dictated the suite of technologies that would be used to create WLab.

The system must not be restricted to a specific server platform

The current VLab solution is written in ASP.NET, which generally speaking demands a Microsoft platform. One might argue that the open source implementation of .NET, Mono [47] invalidates this assertion; however, one would anticipate issues were Mono used to eliminate the Microsoft specificity at the server side. Almost certainly this project will make use of third party libraries and other open source and/or public domain resources. If Mono were used there is a risk of such libraries malfunctioning, given the likely assumption by their authors' that their work will be running on "real" .NET. One of the developers of Mono cites only a 50% chance that existing .NET code will run without refactoring [48]. The Free Software Foundation has also warned that, from a legal perspective, Mono might not actually be "safe" from Microsoft [49]. The conclusion is that Mono is not an advisable option for this project.

Consequently, an early decision has been made to use Java, specifically Java Servlets and JSP in conjunction with the Spring framework [50]. Java is inherently cross platform by design. The Spring framework's Model/View/Controller module for web applications will help encourage an efficient application design. Additionally, Spring's dependency injection approach will make it easy for any future work to extend the application. (For example, one would be able to write support for an alternative virtualisation backend and then use it in the application simply by changing an XML file – no change to core code would be required).

Ensuring cross-platform client access / presenting the VM console in a web browser alongside other content

The existing VLab system uses Terminal Services Gateway (TSG) to connect clients to virtual machines. TSG uses an ActiveX control to call Microsft's Remote Desktop client, which creates the Windows-only client restrictions mentioned previously.

[4] If not, it would be impossible to provide the integrated environment where the practical part of an ICT lab exercise can be displayed side-by-side with the associated static learning content.

One solution would be to use an open source RDP client, such as ProperJavaRDP [51]. This can be delivered as a Java applet, so would address any client-side platform issues. Unfortunately, ProperJavaRDP displays the remote console as a separate window rather than as an applet embedded in the web page. Examination of the source code shows that it would not be trivial to alter this behaviour; the applet mode appears to have been an afterthought in that it simply provides an entry point to create and display the same JFrame used in the desktop mode.

The intention is therefore to use VNC, which is available on a multitude of platforms in both server and client forms [52]. Each virtual machine will require a VNC server installed. Students will connect to virtual machines' VNC servers via the TightVNC [53] Java applet; the TightVNC flavour of VNC has been chosen as a client because the "tight" encoding is common, reasonably fast and compatible with many other modern implementations of VNC, and the source code for the applet is open source and thus modifiable. The applet can also be embedded in a browser page.

The simple web aspect of the client-facing side of the application should not present any cross-platform difficulties, although inevitably there will be browser-specific quirks to overcome.

System data must be stored as XML files, conforming to a clearly defined schema

There are many methods of manipulating XML in Java. For the purposes of this project, the most relevant are those that allow for the abstraction of XML elements into Java objects, such as the Java Architecture for XML Binding (JAXB). JAXB allows a developer to manipulate Java objects in a familiar fashion using getter and setter methods, and have the work of translating this into XML done by the binding library [54].

However JiBX not only provides similar features [55] but also allows one to generate a schema from existing code [56]. This feature is particularly suited to the Agile methods used here where requirements – and thus the data model – may frequently change. In such a scenario a new version of the schema could be re-generated directly from the changed code.

Some test classes were written for familiarisation purposes, and to ensure that JiBX did indeed work as advertised.

Microsoft Hyper-V is the target virtualisation backend

There is a wealth of material on the internet discussing the manipulation of Hyper-V virtual machines using Windows PowerShell, and several libraries exist for this purpose. One library in particular is highly mature and offers a considerable amount of control over the virtualisation layer via a simple command set [57].

Unfortunately, upon experimentation, accessing PowerShell – and by extension, this library – proved to be problematic in Java. Other developers have had similar issues [58]. The use of PowerShell was thus discarded in favour of using Microsoft's Windows Management Instrumentation (WMI) directly. JWBem provides access to WMI in Java [59], and the project website includes example code for manipulating Hyper-V. JWBem also includes J-Interop, a means for sending DCOM requests from Java [60], which was required for a number of functions where WMI did not go far enough. Test code was written as a proof of concept to determine that virtual machines could be started and suspended using these tools.

Additionally, the project goals are likely to necessitate the need to *clone* virtual machines as a basic requirement. Consider the process of a student accessing a lab. Each stage will have a virtual machine, authored by the tutor; however, students cannot all work on this single VM and will each need their own instance to work in. The specific VM authored by the tutor can be considered a *template*, and when students access a lab stage, the system will need to create a clone of this template and allocate it to the student. The student will undertake the activities of this stage of the lab in their clone, not in the original tutor-authored VM.

Test code was written to demonstrate that virtual machines could be cloned through Java code. Actual *cloning* of virtual machines proved problematic because of the time required to copy the large files used for virtual hard disks. However, the use of differencing disks, which simply record changes with reference to a parent disk, resulted in a cloning process that was almost immediate. This approach provides an additional benefit in that such difference disks take up only a fraction of the disk space full clones would.

Authentication will be against an LDAP based directory service, which may or may not be Microsoft Active Directory

The Java Naming and Directory Interface (JNDI) [61] provides a built in method of accessing directories and authenticating against them, including LDAP-based ones. Some test classes were written for familiarisation purposes. During this process it was found that the UID attribute, commonly found on UNIX-based directories and equivalent to a user's login name (as opposed to their human readable "display name"), was not present on Microsoft Active Directory (AD). This presented a problem in that a method of authenticating against AD using user name was not immediately evident. One could authenticate against a display name (e.g. "Paul Neve") but this would be counter-intuitive to those used to using a specific login name (e.g. "K929923"). Additionally, such display names are not always formatted as a user might expect (e.g. "Neve, Paul R" versus "Paul Neve").

Fortunately, it transpires one can authenticate to Active Directory's LDAP implementation using the syntax *user@domain* [62] rather than the more common LDAP strings, e.g. *uid=user, cn=domain*. This syntax provides the desired behaviour where *user* is the user name rather than the display name.

The system must be accessible both within the institution and remotely

Consider VLab where a student accesses a lab from a client machine on the public internet. In order to interact with a virtual machine console, their connection must be routed to an RDP service on an internal IP address and port number. There is no direct route to this internal address and port from the public internet. Currently, terminal Services Gateway addresses this; TSG accepts inbound traffic through HTTPS, then negotiates between server and client to route traffic appropriately.

Without such routing, there exists an issue which might be referred to as "the firewall problem". Figure 8 illustrates this.

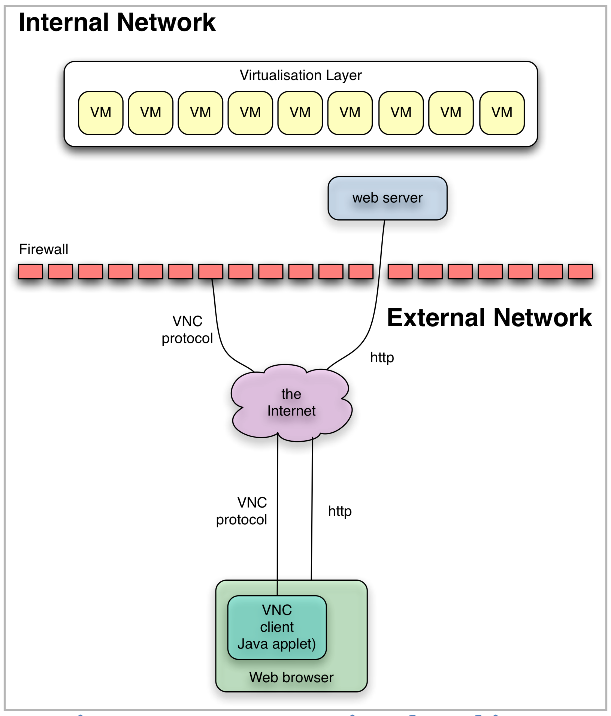


Figure 8 – no route to virtual machines

Figure 8 shows the proposed client environment with using a VNC client in a web browser. The web application aspect of the system can easily be published to the outside world by opening a single firewall port, usually port 80. However, there is no way of routing VNC traffic to the virtual machines. This could be resolved by placing virtual machines in a DMZ, but in the environment at Kingston University no such facility exists, and the only way to route traffic from the outside world into addresses within the institution is by opening ports on the firewall.

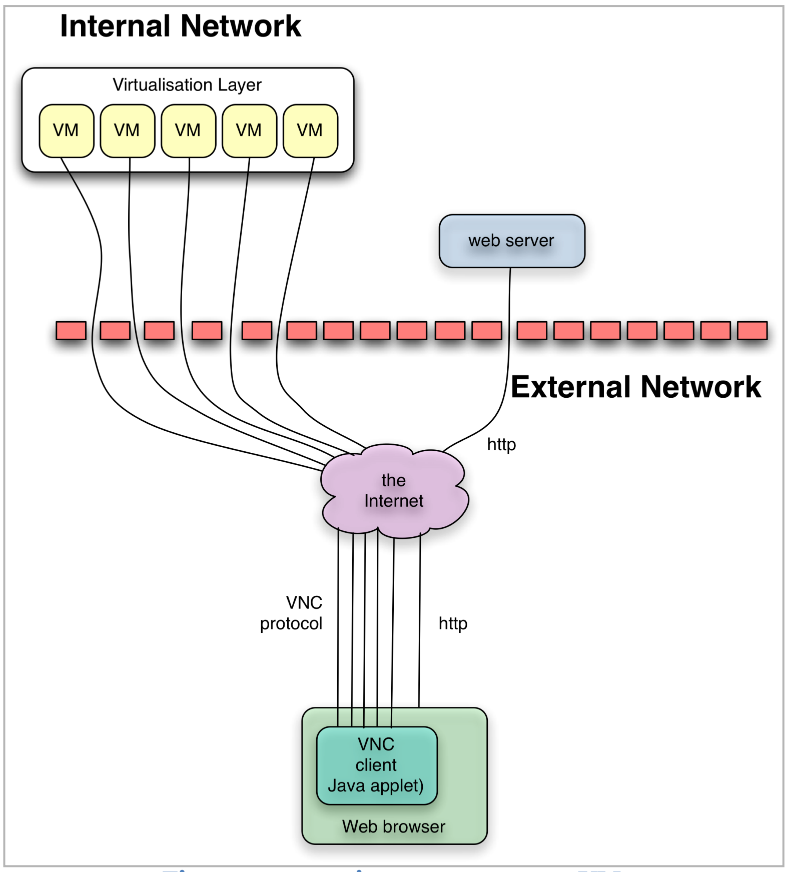


Figure 9 – opening ports to *every* VM

In Figure 9, this issue has been "resolved" by doing precisely this. However, this is not a practical solution. Aside from being unrealistic from a security standpoint, it would mean that each and every virtual machine would need a fixed IP address, which would not lend itself to dynamic VM creation. In the proposed system, during a lifetime of a course dozens of VMs will be created, one every time a student moves from one stage of a lab exercise to another. Every time one of these new VMs came on line, it would require administrative changes to the firewall. This is entirely impractical.

The ideal scenario is to extend the VNC protocol to enable the client to negotiate with an intermediate "routing" application. This routing application would be installed behind the firewall, and a single "hole" would be opened in the firewall at a defined port and IP address, as shown in Figure 10:

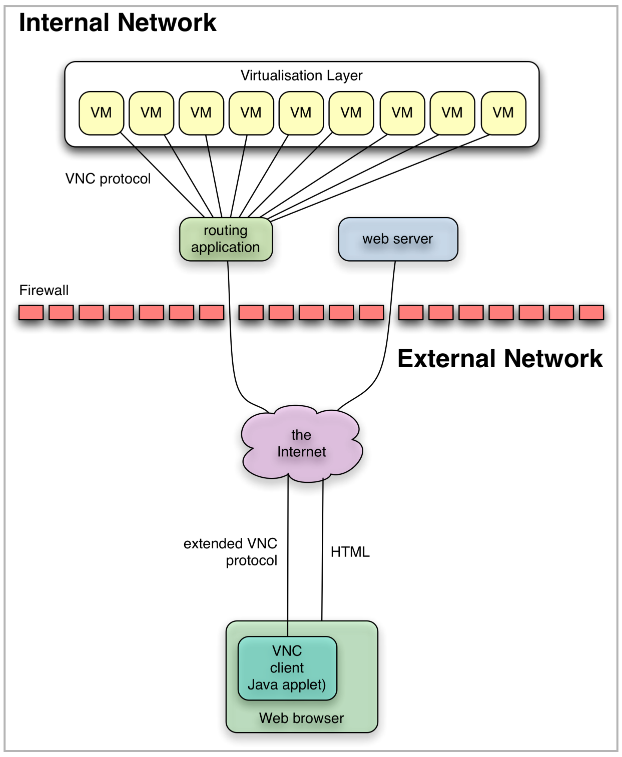


Figure 10 – using an intermediate application to route to VMs

Here, a connection is established to the routing application, the VNC client sends details of the internal address it wishes to connect to, and the routing application then forwards subsequent traffic for that session to the correct internal address and port.

The UltraVNC implementation of VNC has an extension they call a "Repeater" [63]. In "mode 1" it works in a broadly similar fashion to that outlined in Figure 10. However, UltraVNC is a Windows-only implementation, as is their Repeater application, and in any case their Java applet client does not support the Repeater. These factors make it unsuitable for use here.

An alternative is to find a generic port forwarding application and use that in a similar fashion. There are many such applications, but Jumpgate [64] is the most appropriate because it offers an "interactive mode". This works as follows:

- Client opens connection to Jumpgate
- Jumpgate opens no forwarding connection; instead it waits for the client to transmit an IP address and port number
- Once this is received, Jumpgate forwards subsequent traffic to the specified IP address and port

The key element here is that Jumpgate does not require the destination port and IP address to be determined in advance – it can be specified by the client. Therefore, it should be feasible to use Jumpgate as a routing solution. The TightVNC client applet will be extended to provide a "Jumpgate mode", which will send the requisite details to Jumpgate before initiating its own communications protocol. Preliminary modifications to the TightVNC client were made as a proof of concept, and successful VNC connections via Jumpgate to multiple internal addresses were established from remote locations with only a single port exposed at the firewall.

Jumpgate's author notes that he has only built the application on UNIX environments, although he reports that users have had success building on Windows using Cygwin [64]. In order to ascertain there were no platform-specific dependencies, this was tested, and confirmed to work.

The system must be able to deliver both Windows and Linux virtual machines

While Hyper-V can run non-Microsoft operating systems, some consideration is required as to different platforms' needs when *cloning* virtual machines. Particularly in the case of Windows-based operating systems, such clones must be made unique and each workstation must have a distinct and different computer name. Otherwise, clones will “clash” with each other and networking issues will result.

Microsoft’s recommended solution is to use their tool *Sysprep* [65, 66]. Among other things, Sysprep allows an administrator to create a base image, and then roll this image out to many workstations. The Sysprep process will replace the aspects of the base image that must be unique such the first time the clone boots.

Unfortunately, Sysprep operates by scripting the Windows Setup procedure. While this can be configured to run in a zero interaction configuration, meaning the user is not prompted to enter any data and simply has to wait for the process to conclude, it is not a quick process - a test conducted with a simple image of Windows XP took over ten minutes. It would not be acceptable to expect a student to wait for this period of time when moving from one stage of a lab to another.

In fairness, it should be noted that Sysprep does far *more* than is actually needed for this project – which is simply to ensure that the computer name of a Windows VM is always unique. For this, it is unnecessary to go through the entire Windows Setup procedure again – it would be enough to change the computer name on the new VM’s first boot.

David Clarke’s utility *wsname* provides a means of scripting a Windows computer name change that works immediately, simply requiring a reboot to take effect [67]. To test the use of wsname, the following script was created. This was placed in the `c:\wsname` folder on an XP virtual machine as `rename.bat`, and set to run on startup *before* user login, by adding it to the startup scripts in *Local Computer Policy / Windows Settings*.

```
@echo off
c:\wsname\wsname /n:WLAB-$RANDOM[10]
shutdown –r –f –t 1
del c:\wsname\rename.bat
```

The script uses *wsname* to rename the computer to WLAB plus a random string of ten characters. The VM is then instructed to restart in 1 second’s time; the slightly-in-the-future shutdown request means that there is time for the next line to execute, deleting the script and preventing it from running on subsequent reboots. The XP virtual machine was then cloned; upon boot the script changed the clone’s computer name. The clone then rebooted to make the new name take effect.

When tested, the *wsname* approach meant that the first “boot” of a dynamically created Windows XP VM (actually two boots) took approximately one minute and 30 seconds to start. By following Dennis O’Reilly’s guidelines [68] this was reduced to less than one minute. This represents a vast improvement over Sysprep. It is still not ideal, as a student will be subject to a similar waiting time when they navigate to a new lab stage. However, this wait time will only required the first time a student accesses a lab stage.

Assuming that some dynamic means of IP address allocation is being used such as DHCP, and IP address clashes are avoided, there are likely to be fewer problems with UNIX-type VMs such as Linux. However, if such VMs were using Samba – which effectively makes them an analogue of a Windows workstation – then a similar approach would be required.

4 Results

4.1 Inception phase

4.1.1 Initial use cases

Immediately after the kick-off meeting, as the first output of the inception phase, a series of use cases were created. As discussed in section 3.1.4 each distinct step of a use case has been separated onto a single bullet-pointed line; taken in order, one after another, these bullet-points roughly indicate the steps taken to achieve the use case. The aim was to compose use cases that sat at Cockburn's sea level [38] – i.e. user goals that can be accomplished by an individual in a single sitting.

Given the phased approach to development, each phase has its own distinct set of use cases that build on top of the previous. When a use case is unchanged from previous phases it is not repeated. However, when an existing use case is modified to suit a subsequent phase, its text is shown with in grey with additions and modifications in black. The use cases can be found in full in Appendix B.

As noted in *Development and Project Management Methodology*, the MoSCoW method is used to prioritise the use cases. However, prioritisation is also implied by the phased approach to development. Given the caveat that later stages may be "dropped" depending on progress, this suggests that initial phases are class M, i.e. "must haves", with later stages decreasing in priority. Initial prioritisation levels were therefore assigned where functionality in phases 1-3 is class "M", with functions in phases 4-5 set at "Should Have" and "Could Have" levels. This is in line with the DSDM Consortium's rule of thumb that M-level requirements should not make up more than 60% of effort [69].

The initial use cases and their prioritisation levels are as follows:

Use Case Number	Name	Development phase	Priority
1	Change virtual machine state	1	M
2	Connect to running virtual machine	1	M
3	Shut down idling VMs	1	M
4	Book lab session	2	M
1 v2	Change virtual machine state	2	M
3 v2	Shut down idling VMs	2	M
5	Do lab	3	S
5 v2	Do lab	4	C
6	Create/edit lab stage	5	C
7	Create/edit a tutor VM	5	C
8	Create/edit lab	5	C
9	Publish lab to students	5	C

Table 2 - initial use cases

At phase 4, the use cases begin to use the phrase "tutor VM", and this is also reflected in the metaphor for this phase (see Appendix A). A distinction between virtual machines authored by the tutor and those that simply exist on the virtualisation backend (referred to as "backend VMs) was seen as a necessity for the following reasons:

- VMs authored by the tutor will have metadata above and beyond that which can be stored on the virtualisation backend (e.g. course details).
- Some means of distinguishing between those VMs that are intended for use in a lab stage, and those that are there for reasons unrelated to WLab is necessary; otherwise, on busy virtualisation servers, tutors would be faced with a hunt for suitable VMs in amongst the "noise" every time they authored a lab stage.
- It is not practical to expect tutors to author VMs entirely from scratch. They would need to manually install and configure the operating system and their applications before they could begin work for the lab stage itself. Additionally, there would be no way that a

"fresh" VM could be accessed through the application – a newly configured VM would not have VNC installed, thus a tutor would not be able to even connect to it. Thus, tutors will author their VMs by cloning existing ones. In order to avoid a "chicken and egg" scenario, the very first VM created by a tutor will have to be cloned from an existing backend VM. Subsequent tutor VMs might use existing tutor VMs as their "parent", but the assumption is that a system administration will prepare at least one starting point VM that can server as a foundation for tutors.

The associated sub-functions derived from the use cases are as follows:

No.	Name	Project Phase	Priority
10	Log into the system	1	M
11	Start a VM	1	M
12	Suspend a VM	1	M
13	List (backend system) VMs	1	M
14	Display VM console	1	M
15	Show calendar	2	M
16	Choose date and time	2	M
17	List labs	3	M
18	Clone existing VM	3	M
19	List tutor VMs	5	C
20	Filter list of tutor VMs	5	C
21	Filter list of (backend system) VMs	5	C
22	Edit metadata for lab stage	5	C
23	Edit metadata for tutor VM	5	C
24	List lab stages	5	C
25	Filter list of lab stages	5	C
26	Edit metadata for lab	5	C
27	List students	5	C
28	Filter list of students	5	C

Table 3 – sub-functions derived from use cases

The priorities allocated were not immutable, and indeed were *expected* to change, i.e. "C"s would be reclassified as "S"s if development followed the planned schedule. However, if development stalled for any reason, the reverse would apply and "C"s would have been reclassified as "W"s. This is in line with the Agile technique articulated in 3.1.7, "deliverables may change but deadlines are fixed".

One decision that might invite debate is the placement of the tutor-facing, management functionality in phase 5. However, labs could be authored by manually editing XML files in a text editor in combination with using the Hyper-V management tools to create virtual machines. Thus the phase 5 features are important, but not mandatory – the application could be used without them.

4.1.2 Initial use case diagrams

The interactions between the use cases and sub-functions are described by use case diagrams. There is heavy use of the <<include>>notation to represent the sub-functions shown in Table 3. Pilone notes [70] that use cases should be <<included>> if they are subsidiary use cases that otherwise never occur outside of the larger function; this is the case for most sub-functions here. The one exception where <<extend>> is used can be found in stage 5, where the use case **Create/Edit VM**, itself a bona fide user goal, may be used from and interrupt the flow of **Create/Edit Lab Stage**. Such usage would seem to fit Pilone's distinction between <<include>> and <<extend>>.

4.1.2.1 Use Case Diagram – Phase 1

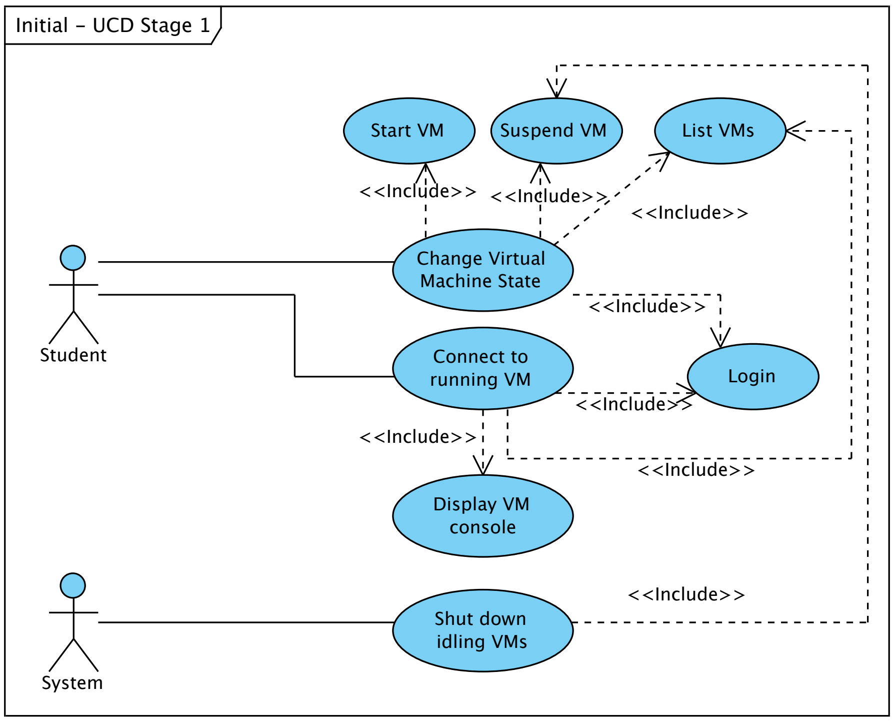


Figure 11 – initial use case diagram for phase 1

4.1.2.2 Use Case Diagram – Phase 2

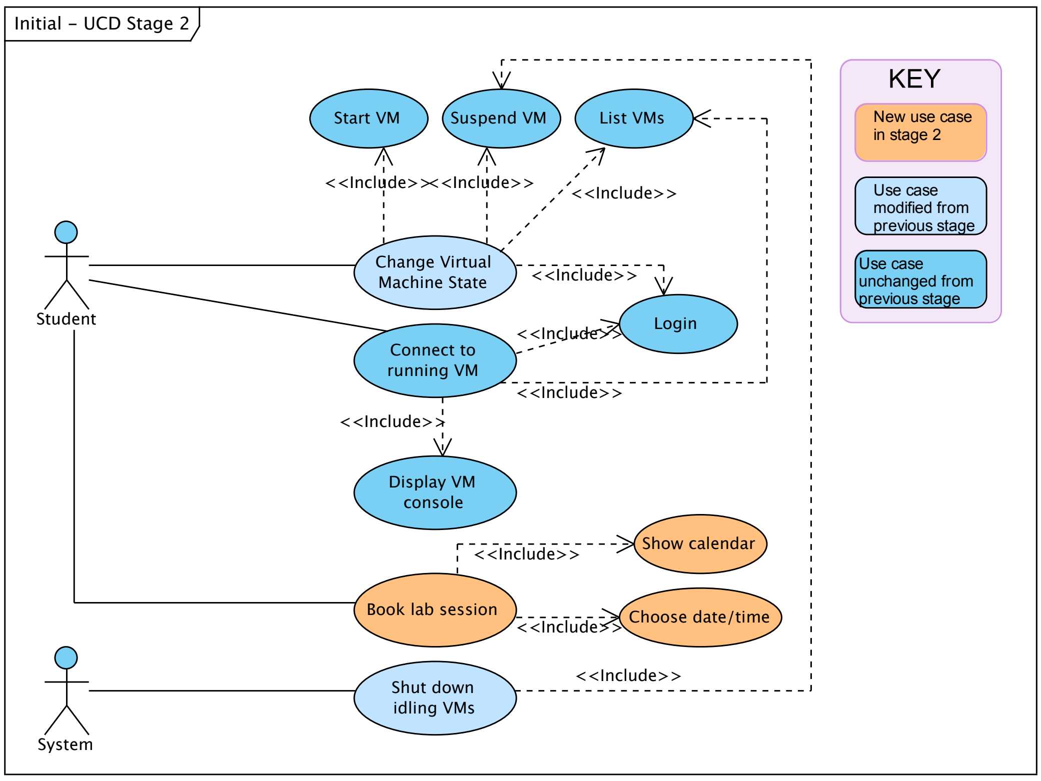


Figure 12 – initial use case diagram for phase 2

4.1.2.3 Use Case Diagram – Stage 3

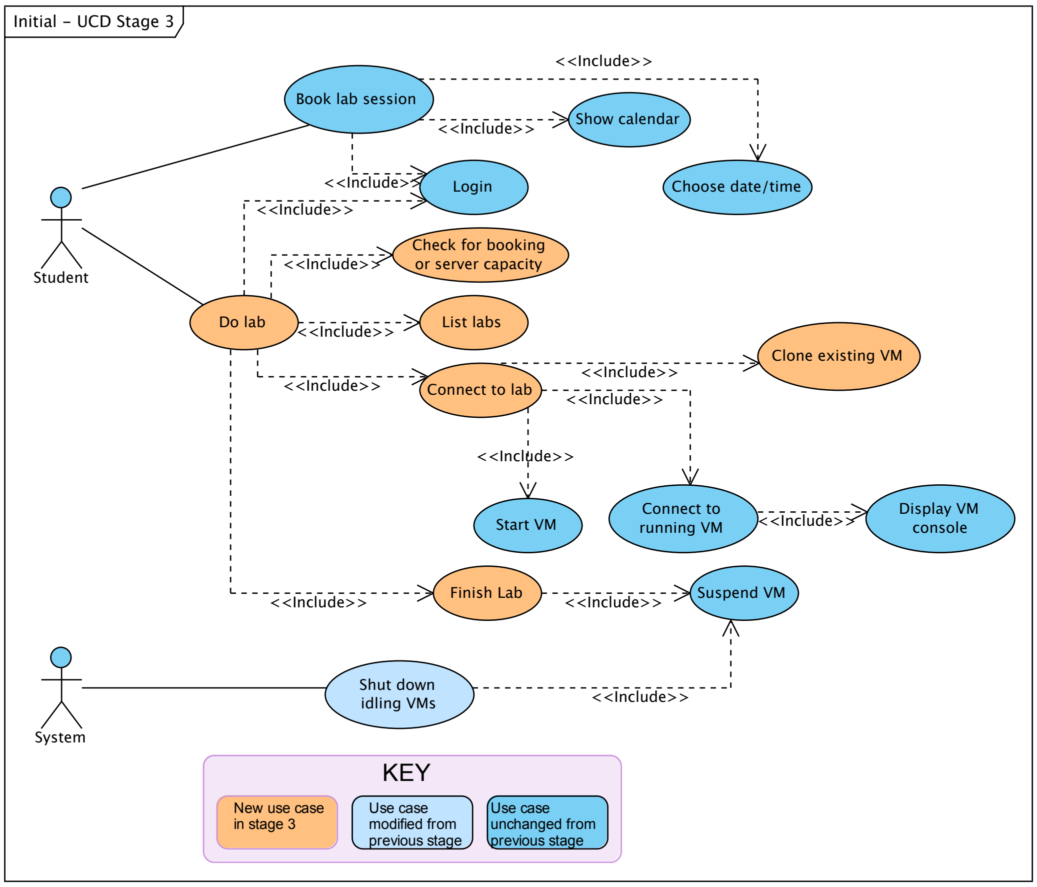


Figure 13 – initial use case diagram for phase 3

4.1.2.4 Use Case Diagram – Phase 4

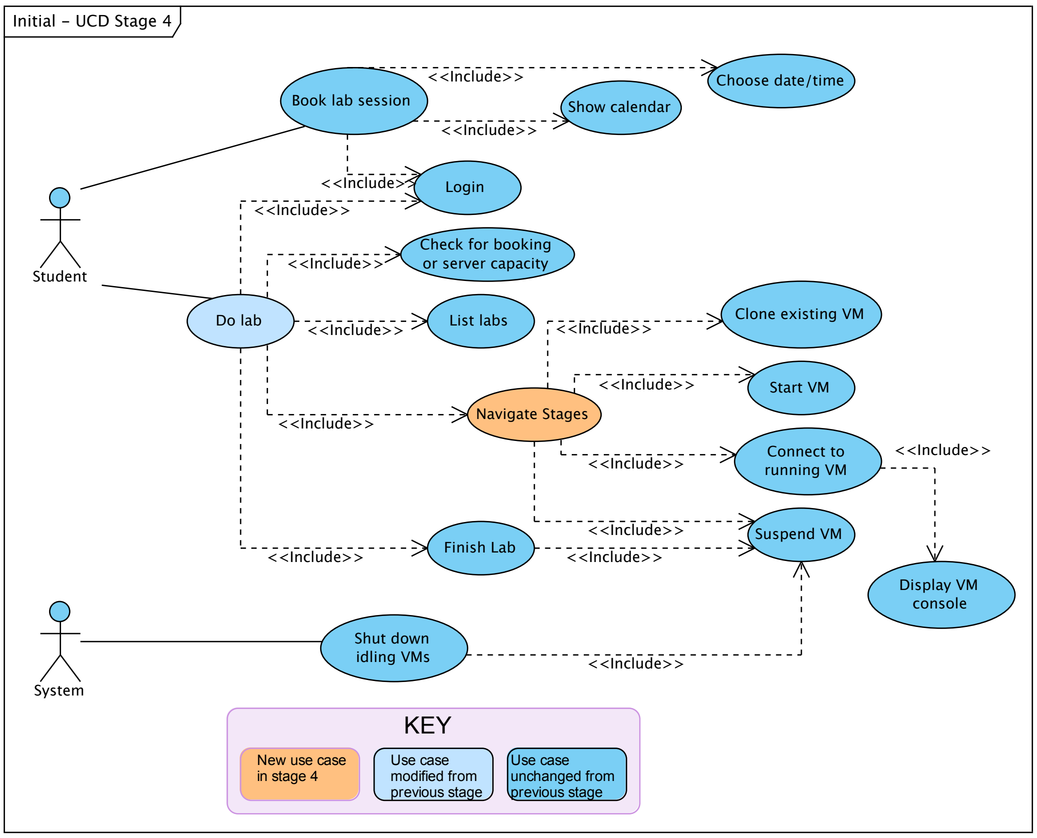


Figure 14 – initial use case diagram for phase 4

4.1.2.5 Use Case Diagram – Phase 5

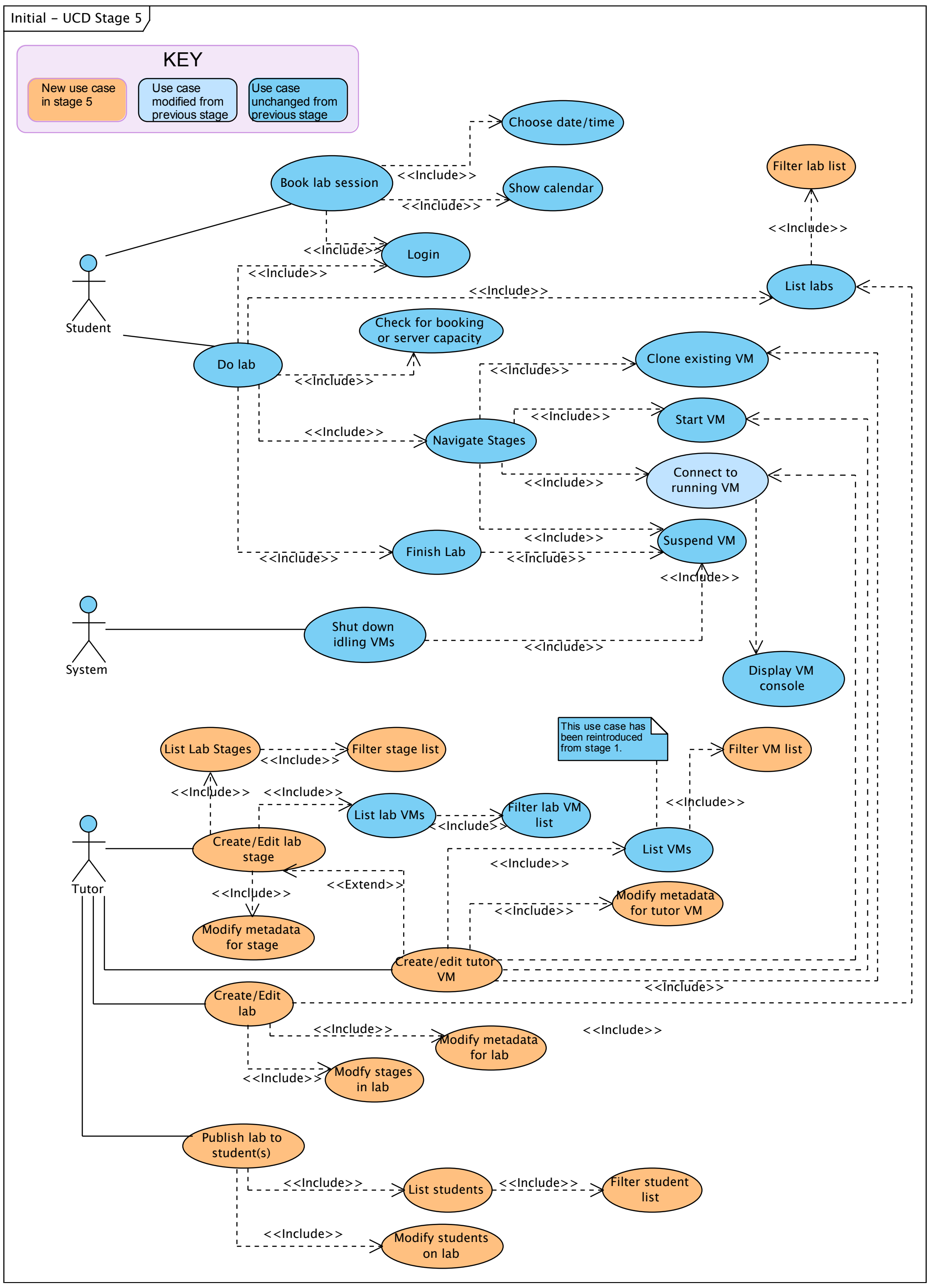


Figure 15 – initial use case diagram for phase 5

4.1.3 Initial conceptual modelling

Figure 16 shows the initial conceptual model. This shows anticipated data classes and associations for all use cases up to and including stage 5. Given the simplicity of this stage 5 model there seemed little point distinguishing between this and earlier phases. Additionally, by designing for phase 5 up front, even if development did not proceed at the anticipated pace and some phases were dropped, any future work to complete the phases would be greatly simplified because such work would already fit the available model.

Ambler notes that a key principle of Agile modelling is "model with a purpose" [71]. At this early stage, the purpose was simply to identify the various classes/objects that the use cases suggest, and the relationships between them. Identifying the attributes of the classes will take place during the Elaboration phase, making use of the outputs generated there such as the screen mockups.

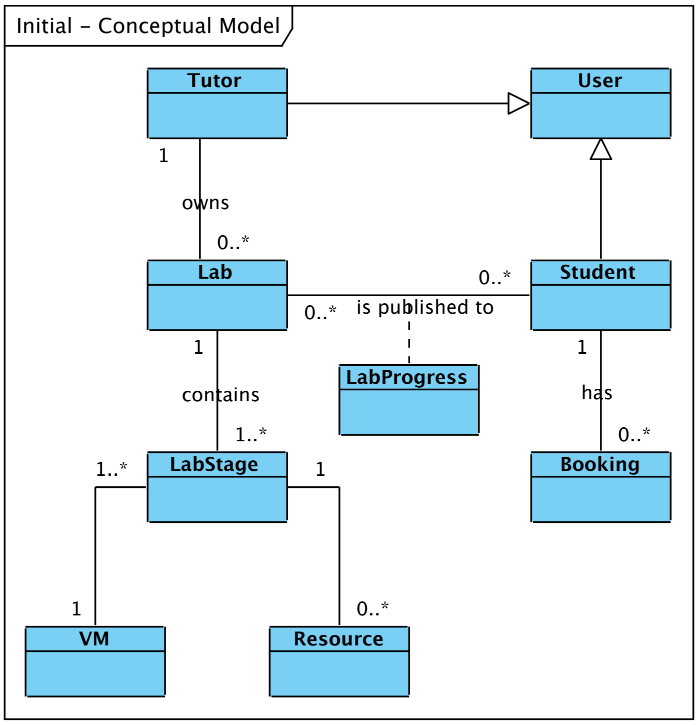


Figure 16 – initial conceptual model

4.2 Elaboration phase

4.2.1 Updates to Inception artefacts generated by the developer during Elaboration

The first "prototype" generated during the Elaboration period took the form of screen mockups designed to illustrate to the user community how the application might look and function, and also to provide a guide for subsequent development. These can be found in Appendix C. In authoring these mockups it became apparent that changes would be required to two use cases, *#5: Do Lab* and *#8: Create/edit lab*, in order to accommodate the concept of "progress" during a lab stage, and to integrate the previous distinct use case *#9: Publish lab to student* into #8. These modified use cases can be found in full in the appropriate part of Appendix B.

An additional sub-function implicit in the screen mockup in Figure 39 is the ability to modify the lab stages within a lab (i.e. the use of the $\boxed{<<}$ and $\boxed{>>}$ buttons to add and remove lab stages).

Therefore, the following new sub-functions have been introduced:

No.	Name	Project Stage	Priority
29	Set progress flag for lab stage	4	C
30	Edit student list for lab	5	C
31	Modify stages within lab	5	C

The use case diagrams for phase 4 and phase 5 must be modified accordingly:

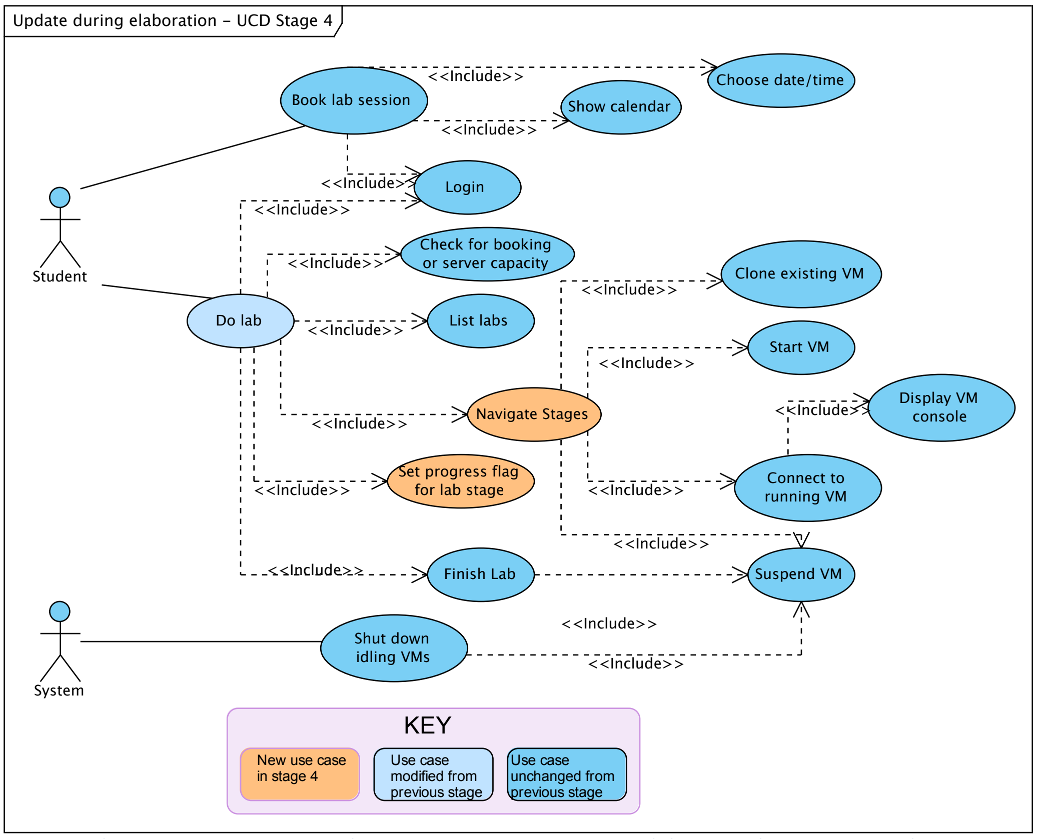


Figure 17 – use case diagram for phase 4, modified during Elaboration

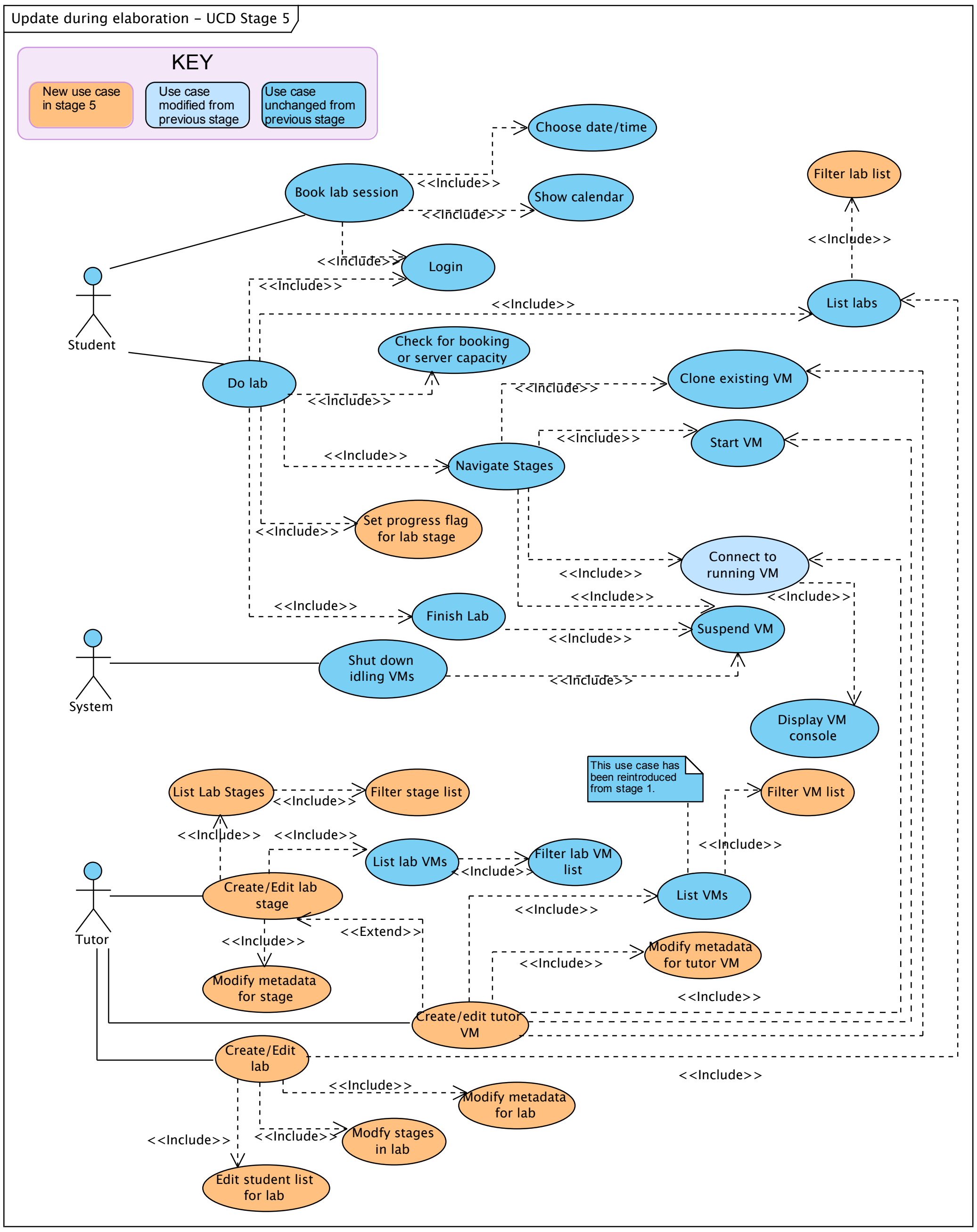


Figure 18 – use case diagram for phase 5, modified by the developer during Elaboration

4.2.2 Updates to Inception artefacts generated by user feedback during Elaboration

For the most part, the user community accepted the artefacts produced during the Inception and Elaboration phases of the project with no changes, with the following exceptions:

Change 1. The maximum runtime length should be *per student*, not *per lab*.

Change 2: Tutors (and students) would like the ability to take a snapshot of a current VM state, and then to be able to revert to the previously snapshotted state.

Change 3. No ability yet exists for non-deterministic labs where the start point of a lab stage is not the same as the end point of the previous. There is an assumption in the mockups that the end point of a lab stage will always equate to the start point of the next. Lab stages must have an additional attribute to hold the "end point" of the exercise. This can be used as a reference by tutors as to the correct "answer" for the lab stage if the start point of the next stage is not appropriate for this purpose. The student user experience is unchanged.

These changes resulted in new versions of several use cases:

Use Case Number	Name	Development phase	Priority
3	Shut down idling VMs	1	M
5	Do lab	4	C
6	Create/edit lab stage	5	C

The snapshot functions will be considered sub-functions. Due to the anticipated complexity of these functions, they have been allocated to Stage 5 of development with the priority set accordingly:

No.	Name	Project Stage	Priority
32	Take snapshot	5	C
33	Revert to snapshot	5	C

The modified use cases can be found in Appendix B. Modifications to the screen mockups were also required, and these can be found in Appendix C (except in the case of the snapshot features, where the feature will simply require the addition of two new buttons to the screens where VM consoles are displayed).

While it was not asked for explicitly, a logical extension of snapshotting is for an initial snapshot to be quietly taken when a student first accesses a lab stage. This will enable the provision of an additional option when a student is working in a lab, **Revert to lab stage start** – this will allow the student to go back to the proverbial drawing board in situations where they have travelled down a blind alley. This is also reflected in the updates to use case 5.

A final update to the use case diagram is shown in Figure 19.

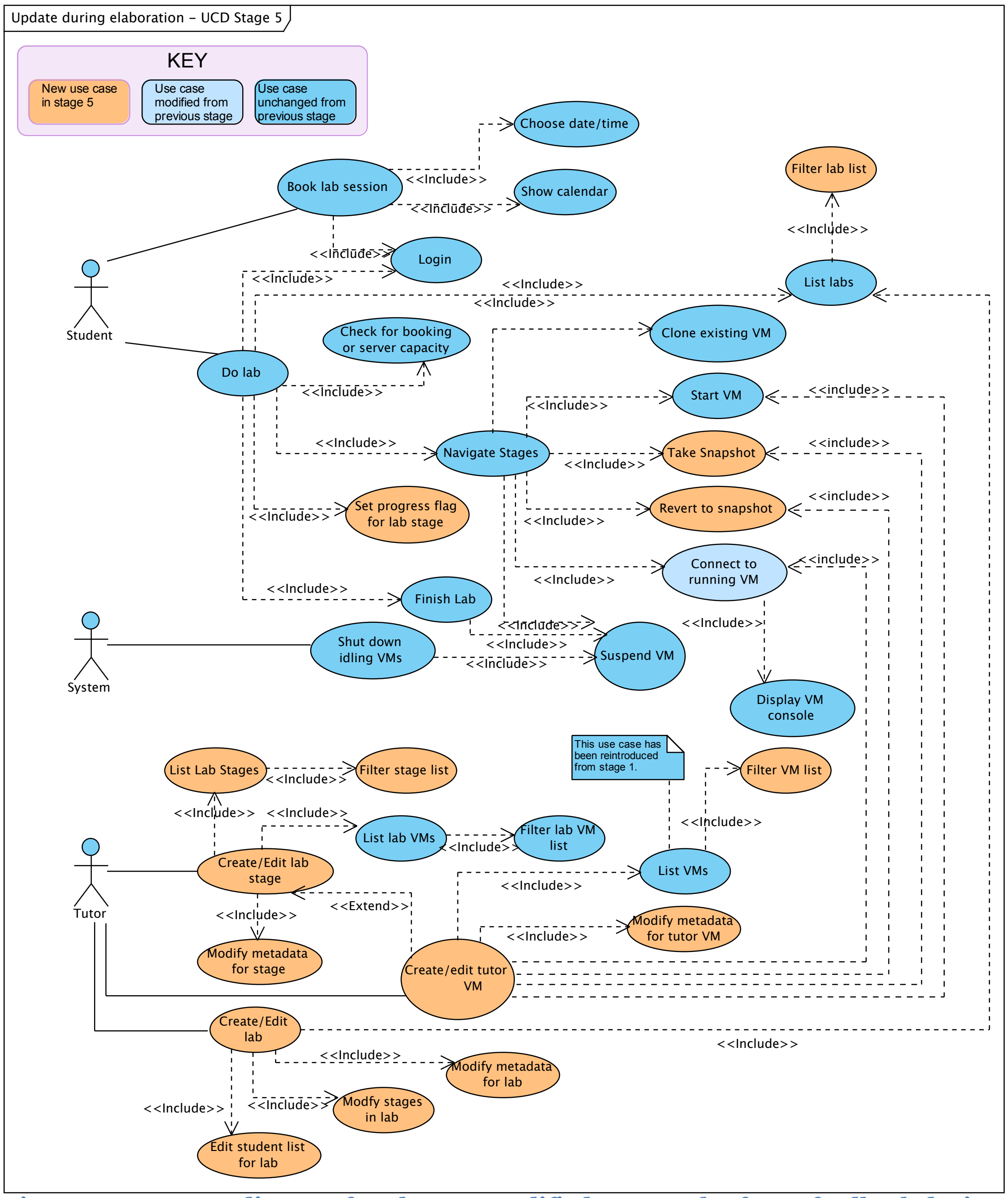


Figure 19 - Use case diagram for phase 5, modified as a result of user feedback during Elaboration

4.2.3 Design modelling

Based on the class diagram that was created in the Inception phase of the project, and using the information obtained during the requirements analysis activities of Inception and Elaboration, a first-cut data model was produced in the form of an updated class diagram:

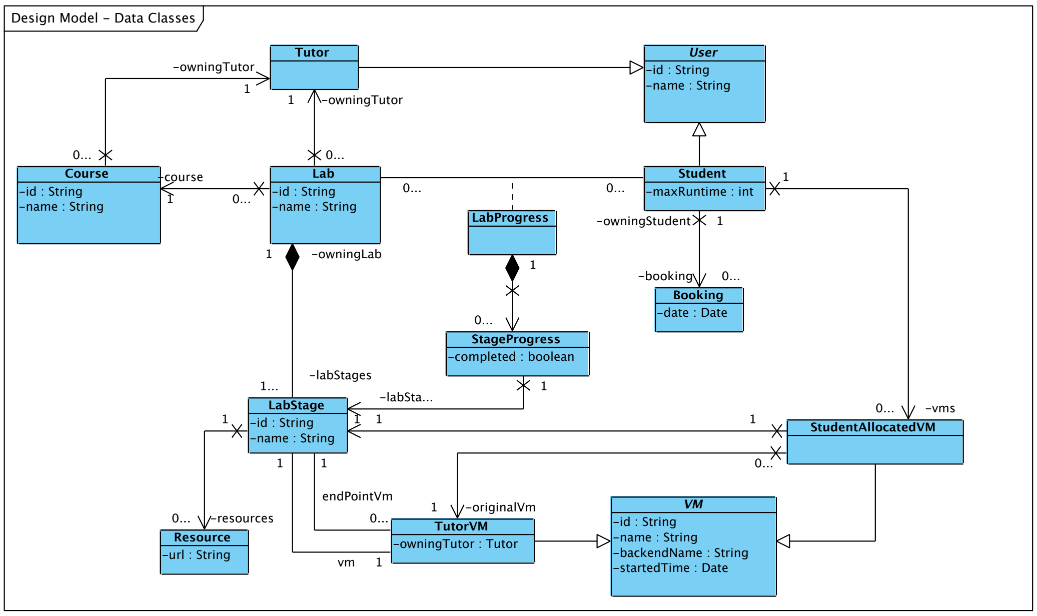


Figure 20 – first cut data model

The attributes implied by previous activities have been added, navigability has been specified along with a number of other changes. Only attributes and associations are shown – as these are data classes it is not expected that they will have any operators apart from those required to manipulate attributes. Such getters and setters are considered implicit (enumerating them would serve little communicative value).

The most noteworthy structural change from the conceptual model is the new distinction between different types of virtual machines that arose in later development stages:

- A “tutor VM” represented by the *TutorVM* class, i.e. a VM authored by a tutor to serve as the start or end point of a lab stage.
- A VM created dynamically by the system that is used by a student when they undertake a lab stage – represented by the class *StudentAllocatedVM*.
- A “backend VM” is any VM present on the virtualisation server. There is no specific class; these are derived directly from the virtualisation server. Both tutor VMs and *StudentAllocatedVMs* are, inherently, also backend VMs.

Some commentary on the changes from the conceptual model follows:

- Use case 5 describes how the system will dynamically create a new VM on demand if needed when a student navigates to a lab stage. This mandates an association between *StudentAllocatedVM* and *TutorVM*.
- Use case 6 version 2 introduces the concept of an “end point” virtual machine, which is represented by the second *endPointVm* association between *StudentAllocatedVM* and *TutorVM*.

Navigability between classes was restricted to a single direction wherever possible. These choices here were driven by the screen mockups and also by applying the “Information Expert” design pattern [72].

The design model was used to create an initial version of the XML schema provided in Appendix H.

4.2.4 Project backlog document

As noted in *Methodology*, a Scrum-style project backlog document was used to help manage development. At the conclusion of the Elaboration stage, this document was as follows:

Item	Use case	Priority	Estimate of Value	Estimate of effort	New estimate of effort as of sprint				
					1	2	3	4	5
Write interface to virtualisation backend		M	10	3 days					
Write interface to authentication backend		S	5	3 days					
Write interface to data backend		M	10	1 day					
Resolve the "firewall problem"		M	10	2 days					
Secure system		S	5	3 days					
Change virtual machine state	1	M	10	1 days					
Connect to running virtual machine	2	M	10	2 days					
Shut down idling VMs	3	M	10	1 day					
Book lab session	4	M	8	3 days					
Do lab (note: introduces dynamic creation of VMs)	5	S	10	5 days					
Extend "Do lab" to provide "stages"	5 v3	C	10	5 days					
Create/edit lab stage	6 v2	C	4	2 days					
Create/edit a tutor VM	7	C	4	2 days					
Create/edit a lab	8 v2	C	4	2 days					
Write snapshotting functions	Various	C	3	2 days					
"Prettify" user interface		C	3	3 days					
			Total days' effort:	40 days					

Table 4 – project backlog document

Key:
Infrastructural
User requested feature
UI

It should be noted that the number of days estimated to complete all the tasks exceeds the amount of time allocated to development. However, this was mitigated by the research that took place before evaluate the various technologies. The code generated during this process was often readily adaptable for use within the development phase.

4.3 Implementation details

At the end of the development stage, a complete implementation of all use cases had been delivered, and in some places additional functionality was added in order to address "holes" that were exposed during initial testing by the developer. A development log, detailing the work carried out in each sprint with a walkthrough and screenshots of the system at each increment can be found in Appendix D.

4.3.1 Modular approach via interfaces and Spring dependency injection

The design approach taken considers the possibility of future support for alternative virtualisation backends, authentication methods and/or data storage methods. Certain aspects of the application are represented via Java interfaces [73]. Implementations of these interfaces are injected via Spring at runtime. Thus, one could write a new implementation of (for example) the virtualisation interface to support VMWare, and switch between the two interfaces – i.e. between Hyper-V and VMWare – simply by modifying one of the Spring XML configuration files.

The interaction of the various packages and key interfaces in WLab is shown in Figure 21. The dotted classes indicate examples of possible future implementations of the various interfaces:

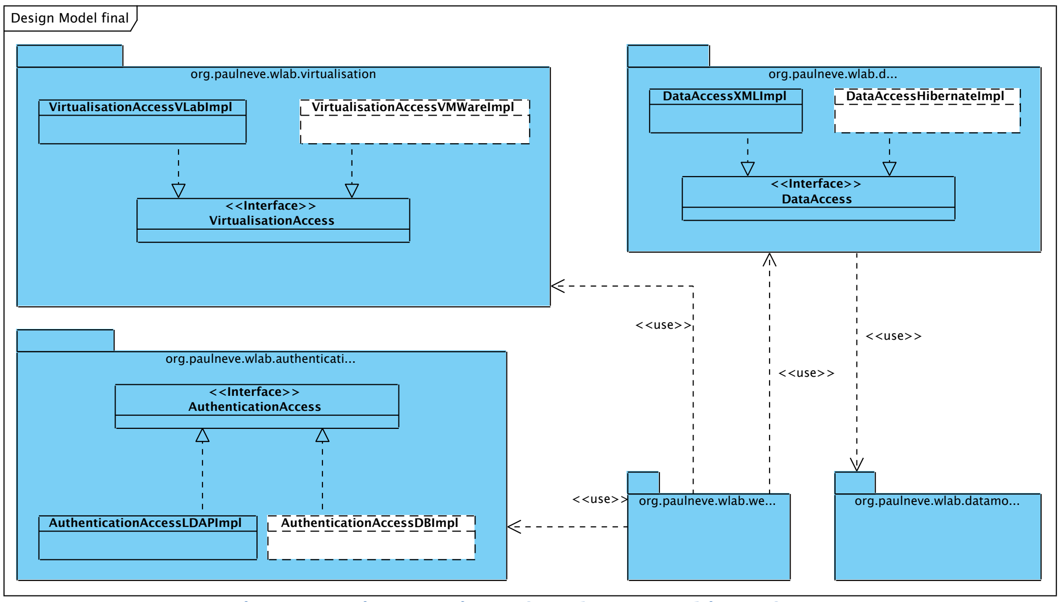

Design Model final
org.paulneve.wlab.virtualisation
VirtualisationAccessVLabImpl
VirtualisationAccessVMWareImpl
<<Interface>> VirtualisationAccess
org.paulneve.wlab.d...
DataAccessXMLImpl
DataAccessHibernateImpl
<<Interface>> DataAccess
org.paulneve.wlab.authenticati...
<<Interface>> AuthenticationAccess
AuthenticationAccessLDAPImpl
AuthenticationAccessDBImpl
<<use>>
<<use>>
<<use>>
<<use>>
org.paulneve.wlab.we...
org.paulneve.wlab.datamo...

Figure 21 - interaction of packages and interfaces

A full specification of these interfaces with details of their methods can be found in *Documentation for Developers* (see Appendix J).

4.3.2 Final data model

The WLab data model is implemented by the classes in the org.paulneve.wlab.data package, shown in Figure 22. It is based on the original conceptual model from Figure 16, but evolved further during development.

Associations between data objects are maintained by string values that contain the ID of the object being referenced. This is not usual in Java, where ordinarily one would simply use an attribute of the same type of the class being referred. Because in Java object variables are simply references [74] this means that the referring attribute would contain such a reference, not a copy of the object.

However, this causes problems when the data moves outside the Java realm. Here, JiBX is used to read and write XML files. Therefore consider a tutor with the name *David Livingstone* and the ID ku12345. This object might be serialised as follows:

```
<tutor>
  <id>ku12345</id>
  <name>David Livingstone</name>
</tutor>
```

However, now consider a corresponding lab object owned by this tutor. Once serialised, it too results in a file:

```
<lab>
  <id>CIM124-lab3</id>
  <name>ECT – mashups</name>
  <owningTutor>
    <tutor>
      <id>ku12345</id>
      <name>David Livingstone</name>
    </tutor>
  </owningTutor>
</lab>
```

Note the red text where data is duplicated across two files. This is not only inefficient, but would be very difficult to keep synchronised. Hence the use of string values containing object IDs as a reference medium.

Many of the one way references in the conceptual model are now two way; for example, the link between *Student* and *Booking* is now a two way link. This change occurred in order to avoid what otherwise would have required convoluted code in the booking functionality.

The XML schema used in the application to implement this data model can be found in Appendix H.

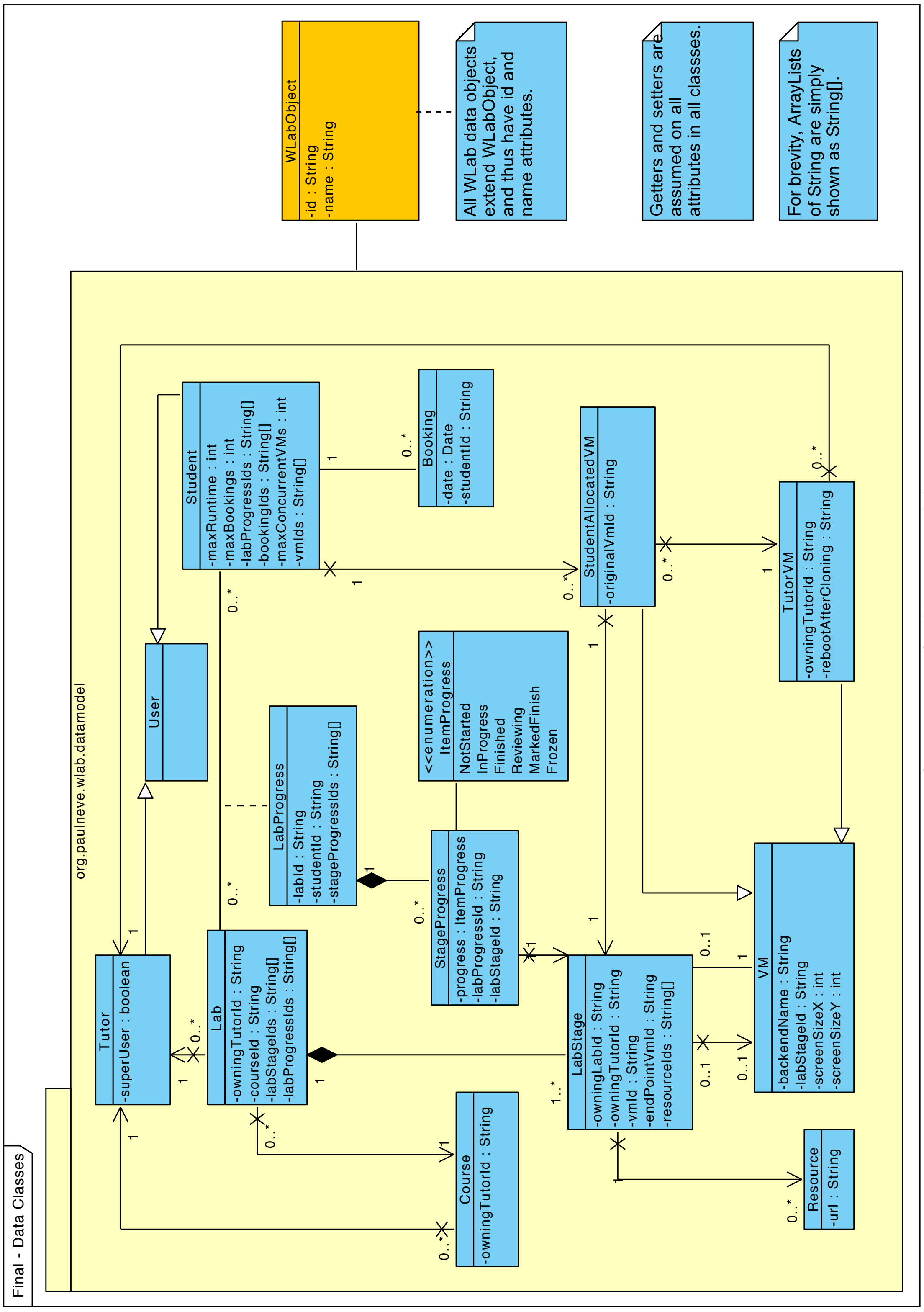


Figure 22 – Final Data Model

4.3.3 Spring MVC

The Spring framework is also used to implement a form of the model-view-controller pattern throughout the user-facing aspects of the application. Java classes at the server side process requests from the user, then build a Spring *ModelAndView* (MaV) object. The MaV includes a reference to one of the JSP views, and the associated model the view will need in order to display meaningful data. Figure 23 provides a simplified illustration of how this works in WLab:

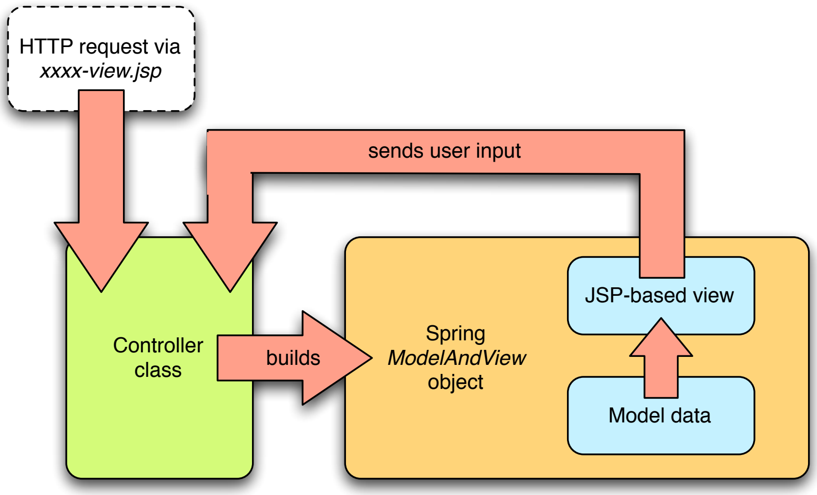


Figure 23 – WLab's use of Spring MVC

4.3.4 Interaction of components in a WLab deployment

The deployment diagram in Figure 24 shows how the various discrete components interact with each other when the application is in a deployed, running state. The element shown in green represents an original output of this project. Elements shown in pink represent existing open source applications that have been modified and/or extended to perform the functions required of them in this project.

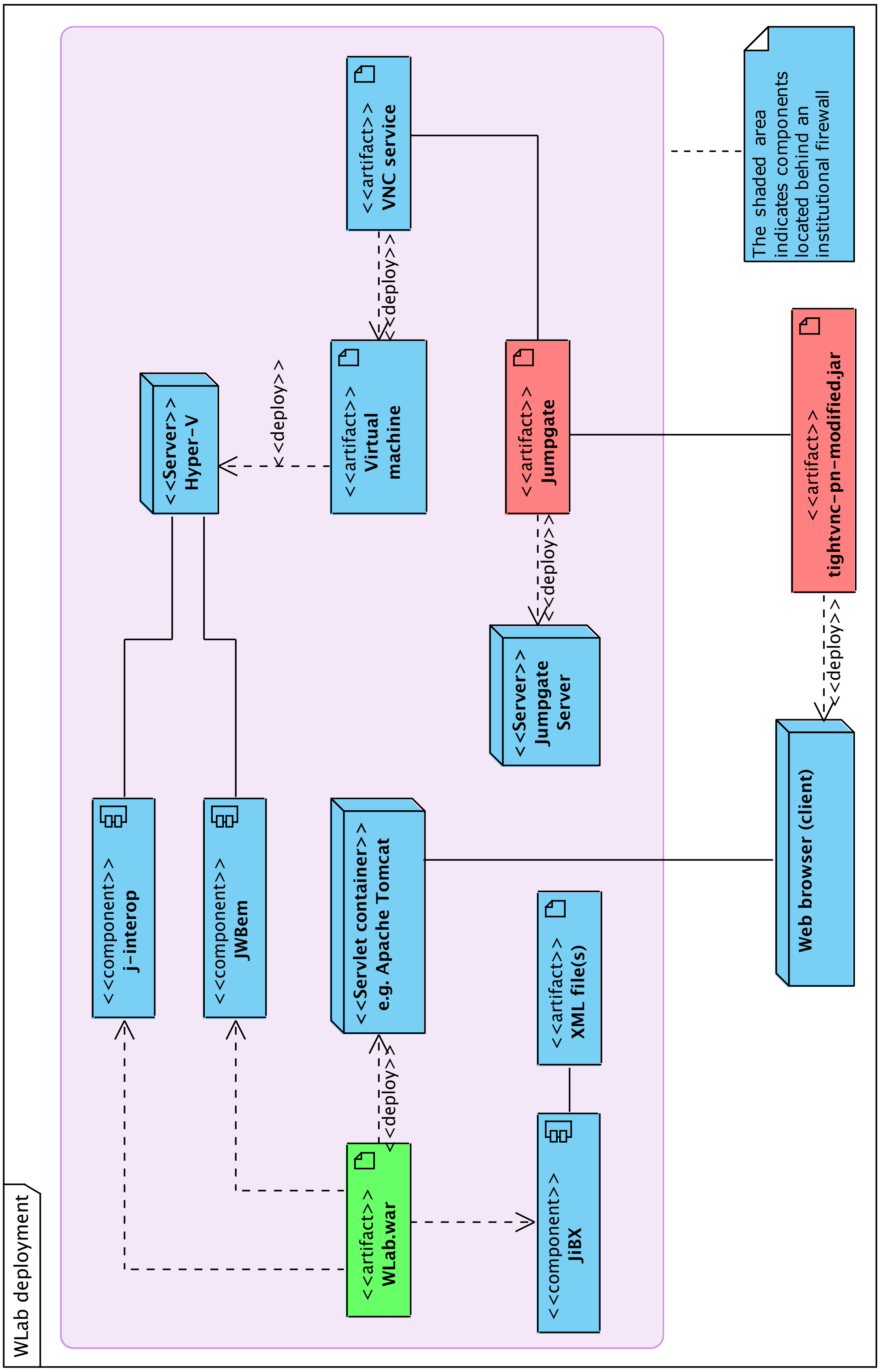


Figure 1 - WLab deployment diagram

4.4 Testing

4.4.1 Suitability for purpose

The original intention was that the majority of testing would occur “organically” as a result of the iterative development cycle; prototypes would be delivered to the user community, who would attempt to use the functions delivered and feed back to the developer bugs, areas where functions do not operate as required, the resultant change requests, etc. Any use cases or elements of use cases that were not provided for in the system delivered would become self-evident during such a process, and any functions that did not operate as intended would similarly be exposed.

Unfortunately, this did not occur as a result of limited availability of the user community during the summer period when the development phases were taking place. This was further compounded by that late stage at which necessary hardware resources were provided. This is discussed further in 5.3, Evaluation of development and project management methodology.

To address this a series of end-to-end tests was written that that work through all functions in the application, matching them up to use cases or other anticipated user needs. Developers might also use this suite of tests in order to confirm that additions or changes to the application have not impacted existing functionality.

This testing programme pre-supposes that the application is starting from a blank slate with no pre-existing data and that a basic Windows XP virtual machine exists on the virtualisation backend. The term “make the VM safe” is used to indicate a point where the VM should be prepared with a tool such as *Sysprep* or *wsname* so that subsequent clones will become unique upon boot.

These tests are intended to confirm functionality of the system in its stage 5 development phase. Functions and use cases from previous phases of development that have been obsoleted in subsequent phases are not tested.

At the end of development, these tests were used to identify outstanding bugs and other cosmetic issues. These were addressed iteratively – after fixes were applied the tests were re-run and this process repeated until all 12 steps of testing occurred with no visible issues arising.

The complete suite of tests can be found in Appendix E.

4.4.2 User testing / Sample labs

Two sample labs were written to provide a foundation for future real-world testing, and to provide exemplars for demonstration purposes:

4.4.2.1 The Paintbrush demo lab

The Paintbrush demo lab is a simple exercise with three stages. In the first stage, the student is presented with a VM with a blank instance of Microsoft Paintbrush and invited to draw a stick figure. In the second stage, the student is invited to add a hat to the stick figure, and in the third, the stick figure is given a bunch of flowers to hold. Approximations of the images after each stages are shown in Figure 25.

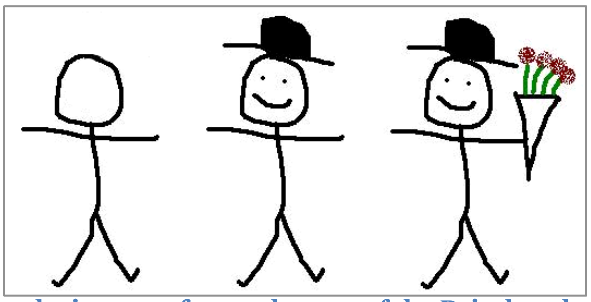


Figure 25 - the images after each stage of the Paintbrush demo lab

The accompanying resources explain how to use the tools in Paintbrush to achieve the picture required in each stage. They are in WLab webapp directory, called *resource1example.htm*, *resource2example.htm* and *resource3example.htm* respectively, and can be found in the WLab SVN.

This lab serves a number of different purposes. The first is to provide a lab exercise that can easily and quickly be replicated by novice tutors during training. If provided with a basic Windows XP instance by their system administrator, tutors can immediately start to create this lab and learn how to use WLab. The tutor documentation uses this example extensively. The second purpose is to illustrate how a well-crafted lab will introduce new techniques with each stage. Here, basic use of the paintbrush is introduced in stage 1, the fill tool in stage 2, and colour selection and the spray can in stage 3. Finally, this lab also provides an exemplar that could be used for testing from a student perspective that does not require specialist IT knowledge on the students' part.

4.4.2.2 "Beginning programming using Java"

This example lab is intended not to provide a real-world, definitive starting point for such a course, but simply to illustrate how such a course *might* be configured.

A basic VM using Lubuntu was created. Within this VM the Java JDK and the NetBeans IDE was configured. This VM was used to create three tutor VMs for a three-staged WLab lab. Each VM provided the NetBeans environment, with the code segments in Figure 26, Figure 27 and Figure 28 pre-loaded in stage 1, 2 and 3 respectively:

```
package beginningjava;

public class Main {

   public static void main(String[] args)
   {
      // TODO: Make the application say "Hello World!"
   }
}
```

Figure 26 - "beginning programming using Java" - code segment for stage 1

```
package beginningjava;

public class Main {

   public static void main(String[] args)
   {
       System.out.println("Hello world!");
       int mynumber = 7;
       System.out.println("My number is "+mynumber);
   }
}
```

Figure 27 - "beginning programming using Java" - code segment for stage 2

```
package beginningjava;

public class Main {

   public static void main(String[] args)
   {
       int mynumber = 7;
       System.out.println("My number is "+mynumber);
       if (mynumber > 10)
       {
           System.out.println("That's a big number");
       }
       if (mynumber < 10)
       {
           System.out.println("That's a small number");
       }

   }
}
```

Figure 28 - "beginning programming using Java" - code segment for stage 3

The accompanying resources for the stages can be found in Appendix I. In brief, the resource for stage 1 talks the student through the addition of the appropriate line to print “Hello world!” to the console. The second resource introduces the concept of variables, and invites the student to change the number assigned to *mynumber* and re-run the program to see the altered output. The resource for the final stage introduces the *if* keyword and conditional processing. The student is prompted to alter the code to fire both of the triggers, and then told to try a value of 10 in *mynumber* (which will produce no output). They are encouraged to think about *why*, and then to “fix” the problem.

There are several purposes behind the authoring of this lab. Firstly, it illustrates a lab more akin to the intended purpose of WLab. The Paintbrush demo lab is trivial by design for good reasons, but when used as an exemplar during demonstrations some in the audience struggled to think beyond this triviality and could not visualise the real world uses. Secondly, the lab illustrates the concept of a lab that is not directly deterministic, i.e. where lab stages do not *directly* follow on from each other. A logical progression still takes place that describes an overall learning objective, but there are “jumps” between the state at the end of one lab and the start of the next. Finally, it demonstrates WLab’s ability to deliver a Linux-based environment.

An interesting fact discovered during the showcasing of this lab was that a tutor could connect simultaneously to the same lab as a student with no adverse effects: both users see the same lab state and can interact with the session. Strictly speaking this is a bug, but was actually perceived by the user community to be desirable behavour in that it provides a mechanism for a tutor to provide real-time support and directly interact with the student’s lab environment. Additionally, it also opens up possibilities with respect to group work. Using a communal login, a group of students could simultaneously access the *same* lab exercise, interacting with it and working through the learning activity path in a collaborative fashion.

4.5 Other project deliverables and outcomes

4.5.1 Source code / Subversion repository

A Subversion repository for the project can be found at

svn://wlab.paulneve.com

This includes source for the WLab application itself, along with the modified versions of Jumpgate and the TightVNC Java applet required for proper functioning of WLab. The latter two components are distributed under the terms of their open source licence – see Appendix G for project licencing details.

Instructions for downloading source from this repository can be found in *Documentation for Developers* (see Appendix J).

4.5.2 User documentation

In 2.2, Project aims, the importance of user documentation as a means of facilitating the creation of a user community was noted. This documentation can be found in Appendix J. This was written with the intent that it could stand alone, separate from the rest of this report. Each of the four user categories specified in 2.2 (student, tutor, system administrator and developer) has a discrete section within this documentation.

4.5.3 WLab web presence

A web site for WLab has been set up at the URL

http://www.paulneve.com/wlab

At the moment, this site contains a wikified version of the user documentation, and downloads of binary distributions of the various WLab components. Functions for community discussion are provided via comments facilities on each page of the wiki, and a separate forum with areas for each of the four user categories.

4.5.4 Dissemination at ALT-C 2010

A poster presentation with accompanying handouts will occur at the ALT-C 2010 conference [75]. The poster and handout can be found in Appendix F.

5 Evaluation

5.1 Evaluation of WLab's role in ICT teaching

The WLab application successfully achieves the aims set out in 2.2, Project aims, providing a mechanism for the authoring (by tutors) and the delivery (to students) of composite learning objects of the type illustrated in Figure 2. A student is presented with a lab environment where a virtual machine console provides them with an area to undertake the practical aspect of a workshop exercise, and static learning content that establishes the context of this virtual machine can be displayed alongside it. Tutors use a web-based interface to author these lab exercises, and to follow students' progress.

WLab is not intended to provide a complete replacement for VLE products such as Blackboard or Moodle. It cannot provide a complete electronic analogue of the traditional tutor-student engagement, where a tutor supplies students with learning materials via media such as lectures, handouts, references to textbooks and so on. Most VLEs – assuming willingness on the part of the tutor, and good course design – can readily approximate this dynamic in an electronic form. This is not the purpose of WLab.

WLab's role in ICT pedagogy is to provide a complementary solution that runs alongside such VLEs. It provides an active learning solution for ICT that current VLEs do not, by supplying the student with a simulation of an on-campus workshop that ordinarily would require institutional hardware. This environment is one in which S-Learning can readily take place. WLab is not dependent on the existence or availability of a physical computer lab for workshop sessions to take place, and can be used in a distance-learning mode at any time of the day.

It should be noted then that WLab in its current form is *not* an electronic assessment application: rather, it is an environment in which activities can take place that encourage learning by hands-on experimentation. In contrast, it is not suitable for assessment-driven learning whereby a tutor directly evaluates a student's performance based on the outputs of the lab exercise, because WLab's ability for a student to navigate between lab stages means there is nothing to stop a student jumping to the end and thus seeing the "answer" to an exercise. Consequently, unwilling students will probably gain little from WLab. However, a student who *wants* to learn may acquire useful data in the act of comparing their current work with the exemplar available in the form of the next lab stage. Important learning content can be imparted via these comparisons, and in some respects this can be considered a form of formative feedback in that the student is assessing *themselves* based on the exemplars available. The quick navigation between milestone points of a workshop activity is not easily replicated outside of WLab, which makes such self-assessment difficult or even impossible in a conventional, physical workshop arrangement.

The composite Lab object in WLab also ensures that both the workshop environment itself and the accompanying materials are inexorably linked, always delivered together, with one putting the other into an appropriate context. Contrast the conventional approach, where students may be encouraged to take handouts away for additional home study after an on-campus workshop session; these handouts' usefulness are curtailed when removed from the context of the computer environment available at the institution.

Labs are ultimately stored as files on a computer file system and are easily distributable, which opens up collaborative possibilities for ICT workshops not previously possible. Tutors might share their labs with colleagues and other institutions, who may extend or tailor the labs to suit their purposes. The logical conclusion of such collaboration would be a library of open educational resources (OERs) potentially stored as IMS content packages [76]. While existing OER repositories such as Jorum contain a wealth of static learning content [77], their ability to supply content suitable for workshop based ICT teaching is limited. Widespread adoption of WLab would enable such repositories to deliver fully self-contained workshop sessions for ICT.

One criticism of the application that has been levelled by academics is its terminology. There may be tutors who would otherwise enjoy the pedagogic benefits of the application and the composite learning object it implements, who would be discouraged by technical discussion such as "backend virtual machine versus tutor virtual machine".

One argument is that terms more in keeping with a pedagogic lexicon (e.g. "workshop template") would make the application more accessible and increase the likelihood of widespread uptake of the application. However, a counter argument is that new terminology still requires explanation to tutors, with the added disadvantage that you also have to explain the new terminology to those who might actually have understood "virtual machine". Additionally, there is a school of thought that making WLab *too* accessible is not necessarily a desirable outcome due to considerations of competency. Novice users who do not fully understand the implications of creating virtual machines that are connected to an institutional network could actually do serious harm to IT services via WLab (e.g. if a VM were created with an IP address or machine name that clashed with a server). Therefore, while WLab provides an environment where tutors do not need to deal *directly* with the virtualisation backend, it is probably fair to state that they do need to know what a virtual machine *is*, and what the implications are for a system that dynamically creates them on demand. The documentation for tutors makes this plain, and thus sets out to dissuade those who may not understand such considerations for using WLab.

A compromise exists in that an institution can choose to override the default terminology of WLab by modifying the *strings.xml* configuration file as they see fit. Should an institution feel they could manage the technical risks (e.g. by placing the WLab environment in an entirely isolated network) it is quite possible for the technically oriented terminology to be replaced with one designed to be more accessible by less computer literate tutors.

5.2 Evaluation of technology outputs

The project has delivered the following software outputs:

5.2.1 The WLab application

The WLab application delivered represents a complete implementation of all functionality specified by the use cases, screen mockups, and other Inception and Elaboration phase artefacts. Indeed, the application that emerged at the end of the development process actually goes *beyond* the requirements explicitly specified. Nevertheless, the following limitations have either been identified, or are anticipated as a possibility when deployed in a real-world environment:

5.2.1.1 Issue #1: Resolution of virtual machines

One limitation of WLab, compared to the existing VLab application, arises *because* of its integrated presentation of both virtual machine console and accompanying static learning resources within a web page. VLab's use of Windows Terminal Services and Remote Desktop means that a student connecting to a lab session can utilise the entire screen space of their client computer. A tutor does not therefore need to consider screen resolution settings. In contrast, WLab uses the resolution of the tutor VM. If this is set higher than that of the client PC, the student will have to scroll around the virtual desktop via scrollbars. Setting the resolution to a low value eliminates this issue, but often this will hinder the student's ability to work efficiently on the workshop task. For example, using an integrated development environment such as Eclipse or NetBeans with a resolution of 800 x 600 would be an incredibly frustrating experience. The ability to scale the virtual machine console mitigates this to some degree, but the loss of detail that is entailed may make text hard to read which, again, will hinder the student's efficiency.

5.2.1.2 Issue #2: Scalability and concurrency

Thus far, all use of the application has been in a testing or demo environment with only a handful of concurrent users. The scalability of the system is totally untested and, ultimately, it will be difficult to establish this without subjecting it to a notable amount of concurrent users in a production environment.

WLab uses only a single server with no direct ability to balance load across several servers[5]. The amount of concurrent users that a WLab deployment can accommodate will depend on the capacity of the virtualisation backend, rising with the RAM size and processor power of the server.

Regardless of the capabilities of the hardware, one unknown quantity is the Hyper-V software's ability to cope with many simultaneous requests to create a virtual machine. Consider the scenario of a scheduled workshop with 20 students. When the tutor tells his students to "click the *Create lab*" button for the first time, WLab will send 20 simultaneous requests to Hyper-V to clone a VM. WLab's scalability will be directly linked with how well Hyper-V can handle such an event.

An attempt to simulate such a scenario was conducted with via the code fragment below, which uses WLab's *VirtualisationAccessHyperVImpl* class:

```
for (int i=0; i<20; i++)
{
      Thread thread = new Thread()
      {
            public void run()
            {
                  hv.cloneVM("Base XP", "Test-"+UUID.randomUUID());
                  this.stop();
            }
      };
      thread.start();
}
```

This ran with no problems and 20 VMs were created within a second or two. This would seem to imply that there would be no problem. However, there may still be factors that remain unconsidered that would impact a real-world many-user scenario.

Finally, the application itself is written with concurrency in mind, using Java Servlet sessions to maintain user-specific states. No issues here are anticipated – there is no theoretical reason why 3 users should work and 30 or even 300 should not – but it should be noted that this, also, is untested.

5.2.1.3 Authentication

Kingston University uses both Active Directory and Novell eDirectory, and the intention was to develop and test against both, which would demonstrate compatibility of authentication across different vendors. When tested against Kingston University's eDirectory server, authentication was unsuccessful. Given that similar problems were experienced when trying to connect the popular Java-based LDAP browser JXplorer one can conclude that an issue exists between KU's particular instance of eDirectory and Java's JNDI. Unfortunately development had to cease before a resolution was found. Authentication against non Active Directory LDAP servers is thus untested.

5.2.2 Connectivity and security

As detailed in Figure 10, Jumpgate is used as an intermediate routing application in order to route network connections to the correct virtual machine through an institutional firewall.

In its interactive mode, the default version of Jumpgate accepts any and all connections to the port specified in its runtime command line. It also supplies a prompt upon the establishment of a TCP/IP connection:

```
<hostname or IP address> <port> (e.g.: bsd.gr 80):
```

[5] Although if a clustering solution was used that presents itself to other services as a single server, there is no reason why such a solution could not be used with WLab.

This represents a considerable security risk in that someone could manually connect to this open port and from there, establish a connection to any internal machine accessible from the Jumpgate server. The helpful reminder of syntax is rather *too* helpful in this scenario!

In order to address this, Jumpgate was extended to accept a password parameter as part of its command line. Any incoming connection that does not supply the password is disregarded. This password must be specified in the Jumpgate command line when run, and thus would only be known by the system administrator responsible for starting the Jumpgate service. Additionally, the "helpful" prompt has been removed. This would prevent any casual port-scanning activity from using Jumpgate. It is, however, by no means an entirely secure solution. When establishing a connection to Jumpgate, the password is sent in a plain-text format and thus a hacker with packet sniffing software would be able to read this password. The intent behind these changes was simply to discourage the "casual" kind of hacking commonly seen in university environments.

A superior approach from a security point of view would have been to make use of the "experimental" version of the TightVNC applet which has support for SSH tunnelling. This would provide routing facilities – making Jumpgate unnecessary – and also have the benefit of encrypting all traffic [78]. However, this would have make the operation of WLab dependent on the existence of such an SSH server at an institution.

As well as the other Jumpgate-related modifications made to the TightVNC applet, a facility to encrypt applet parameters was also added. This means that users cannot derive information about the internal network by selecting *View Source* while viewing one of the web pages that comprise the client-facing aspects of WLab. However, this encryption uses a hard-coded string within the VNC applet source code, so one could easily obtain this string by reading the (freely available!) source. It should be noted that, as with the measures taken in Jumpgate, this is intended simply to obfuscate such parameters from casual mischief-makers among the student body. For extra security, a system administrator could change the string and then recompile the applet.

5.3 Evaluation of development and project management methodology

On the surface, the hybrid Agile approach outlined in 3.1 was a resounding success. All 5 development phases were completed and, as noted previously, the final application goes beyond the specified user requirements. The original development schedule, if anything, proved to be slightly pessimistic.

However, this is only part of the picture. A key principle of the approach was iterative development. The expectation was that the regular delivery of working prototypes to users, and a regular feedback cycle would both ensure the end product fitted user needs in the real world and also eliminate any bugs en route. Unfortunately, opportunities for feedback between developer and the user community were less than hoped for at the start of the project. The development period – scheduled during June and July – coincided with the summer holiday period. As a result, the user community were often unavailable at the end of an iteration, and no feedback was available. Development still adopted a weekly cycle of iterations, and at the end of each iteration the features from one of the five project phases (see 2.3.1) was indeed implemented, but the user community's sight of these increments was irregular.

The issue was compounded by the lack of on-site hardware resources available to the project during development. Even when the required individuals were available, the post-iteration dialogue was restricted to a session of usually an hour or two's duration, with new features demonstrated by the developer. No real capacity for "hands-on" time by the users was possible as the only place the application existed in a deployed and usable state was on the developer's own hardware. Hardware resources were eventually made available in early August, and the application has now been rolled out and is accessible by the users, but by then the development period had concluded and the opportunity for an iterative, evolutionary approach had elapsed, at least with respect to the schedule outlined in Figure 4 and the project detailed here.

The most deleterious outcome of this is that the application remains largely untested from a real-world tutor's perspective. The end-to-end test plan in 4.4.1 was designed to mitigate this to some extent, but ultimately represents the developer's understanding of the tutor role. Some assumptions may be faulty. While better than nothing, the test plan is markedly inferior to the evolutionary, ongoing testing that would have occurred in a true iterative, Agile development cycle.

In hindsight, the DSDM Project Approach Questionnaire [79] might have been a useful tool. As was the case with the other techniques borrowed from the various Agile methods, it would have required adaptation for the small project team involved, but point 13 in particular (which ascertains whether "it will be possible for the Solution Developers to have easy access to Business Ambassadors and Business Advisors throughout the project") would have exposed the issue at an early stage.

Given the reduced availability of the user community, it is perhaps worth re-evaluating whether more traditional, "big design up front" methodologies might have been appropriate for this work. With a BDUF approach, the first month or so of effort could have been devoted to producing an exhaustive and complete design[6] for the application before a single line of code was committed. Once complete, this design could then have been discussed with and signed off with the users. Only a single session with them would have been required.

One criticism of BDUF is that it can yield software that, once delivered, proves not to fit the needs of its user community. This usually occurs when requirements change during the development period, or when requirements are not correctly encapsulated in the first place – both issues that Agile approaches set out to prevent. However, it can be argued that this project would be no worse off – the software delivered at the end of the design and development cycle would still need its suitability for purpose ascertained by actual use in the real world. However, a detrimental aspect of BDUF would be the inevitable reduction of development time, which would result in a less comprehensive feature set.

Proponents of Agile go to great pains to dissuade people of the misconception that Agile means *no* design up front [80, 81]; rather, "Little Design Up Front" or "Just Enough Design Initially" (or JEDI) is the correct approach. In this project, JEDI was used to good effect. The design artefacts from the early stages were not as comprehensive as would have been the case from a BDUF approach[7], but they provided enough of a skeleton to ensure that, during development, blind alleys were avoided. These artefacts were in no way final, and evolved during the development process, but without these foundations a trial-and-error approach would have been inevitable. This would have greatly complicated development.

Ultimately, while it was not intended at the outset, the methodology evolved into something that has commonalities with Feature Driven Development:

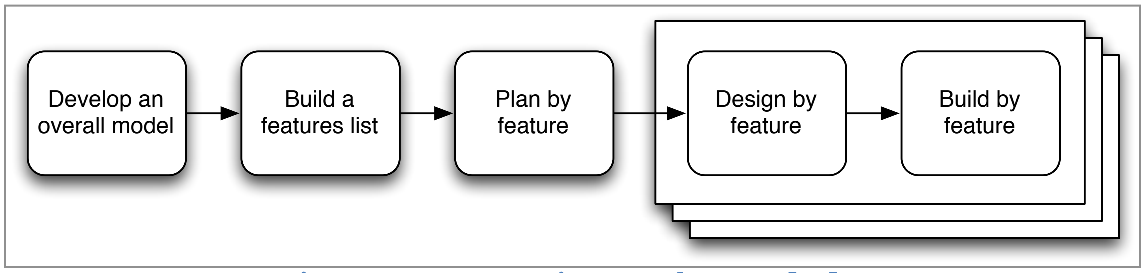


Figure 29 - Feature Driven Development [82]

The stacked layers indicate iterations. If one substitutes the word "phases" for "feature", this fits the development approach that ultimately emerged fairly closely (albeit slightly re-ordered). The grouping of functions within each phase proved to be well organised. In most cases, subsequent phases entailed the creation of *new* code rather than the modification of existing code. In only a few instances did existing code require any fundamental changes to accommodate a subsequent phase, and for the most part these had been anticipated by the versioning of certain use cases (see Appendix B), and in the use case diagrams (see 4.1.2, Initial use case diagrams).

[6] In contrast to the skeletal artefacts actually produced, e.g. Figure 15.

[7] The Agile Manifesto [1] states "working software over comprehensive documentation", so this is by no means a bad thing!

Overall then, difficulties of the iterative/feedback aspect aside, the methodology worked well. Ultimately, as stated in the *Principles of the Agile Manifesto* [83], “working software is the primary measure of progress”. In this respect, progress can certainly be said to have occurred.

6 Conclusion and future work

For ICT subjects, virtual learning environments provide an electronic analogue of a traditional FE or HE classroom environment – but only to a point. VLEs fail to provide an environment in which students can undertake practical lab work, which is a crucial component of ICT teaching.

Past projects that leveraged virtualisation to deliver this component electronically have addressed only the infrastructural issues presented by ICT workshops – their delivery to distance learning students, and overcoming the limitations of institutionally available equipment which is usually ill-suited for advanced ICT teaching.

WLab goes further by using the characteristics of virtual machines for pedagogic advantage. Virtual machines are only one element of WLab's composite learning object, supplying the data, environment and work area for the student to undertake a **stage** of a **lab** exercise. They are delivered in an integrated fashion alongside static learning content (called **resources**) which set an appropriate context for the practical activity, linking it to other learning content such as lectures, textbooks and other sources. The concept of stages (with each stage described by its own virtual machine) permits a tutor to define an optimum path through a lab activity with working exemplars represented by milestone points. A student can navigate between these stages and immediately see their work area update. They can compare and contrast their current work with a previous or future stage of the activity, and then revert back to their own efforts. Such comparisons provide important learning content.

WLab labs are portable, thus the application provides a foundation for the growth of a community where tutors and institutions can collaborate on and share complete, self-contained ICT workshops. Previously, such collaboration could only involve the static content aspect of a workshop exercise, and tutors/institutions would have to replicate the original laboratory environment or support the students in doing so themselves.

To promote the creation of such a community, an open approach has been taken throughout. Where third party components were required, open source solutions were chosen. From a user perspective, the application has been written to run across all operating systems and web browsers. From an administrative perspective, while time constraints and institutional needs have dictated the requirement of Microsoft Hyper-V as a virtualisation backend, the application has been written in such a fashion and using components that would enable future work to simply "slot in" support for alternative virtualisation solutions. A similar arrangement exists for authentication and data storage.

The creation of a WLab user community was the driving factor behind the establishment of a WLab web site, available at http://www.paulneve.com/wlab. A wiki and forums are provided for community discussion and collaboration.

While a stand-alone project in its own right, WLab was always intended to be an iterative step within a wider programme of e-learning development at Kingston University. Figure 30 (itself a progression of Figure 3) shows how this programme will guide and shape future work:

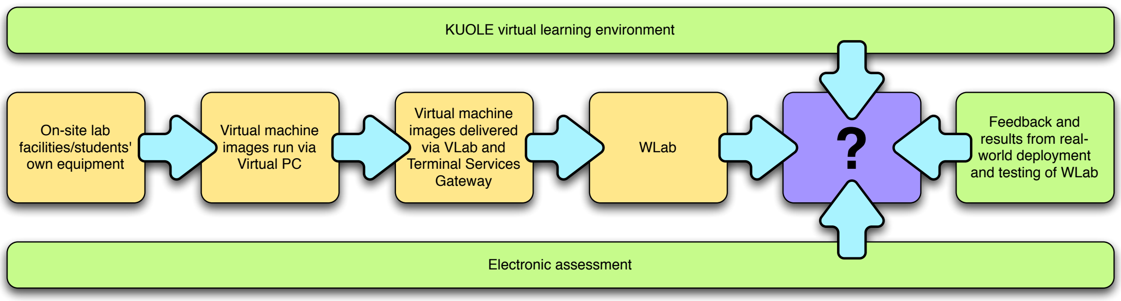

KUOLE virtual learning environment
On-site lab facilities/students' own equipment
Virtual machine images run via Virtual PC
Virtual machine images delivered via VLab and Terminal Services Gateway
WLab
?
Feedback and results from real-world deployment and testing of WLab
Electronic assessment

Figure 30 - a roadmap for future work

WLab is intended to complement existing VLEs; it does not replace or even yet integrate with them. Integration with Kingston University's own KUOLE VLE was de-scoped at a relatively early point of this project, but re-introducing such integration is anticipated to be an early endeavour. Certain elements of the XML schema and several of the non-functional requirements have KUOLE in mind which makes it a logical starting point for VLE integration; later, integration with industry standard VLEs such as Moodle and/or Blackboard will be highly desirable.

The preceding has primarily been concerned with documenting the *creation* of WLab. There has been little opportunity for real-world testing; testing has concentrated on basic application functionality and eliminating bugs. However, in the immediate future it is expected that several labs will be authored and delivered to students in a real-world environment. This will almost certainly expose gaps in functionality and flawed assumptions that were not readily revealed in the theoretical evaluation and analysis in the preceding. There will also be opportunities to gather analytics that will form the basis of an evaluation of WLab's pedagogic effectiveness in a real world environment.

The Learning Technologies Research Group at KU has produced a number of outputs designed to facilitate electronic assessment. WLab, as discussed in 5.1 Evaluation of WLab's role in ICT teaching, only provides a very basic form of assessment in that a student might assess *themselves* by means of comparison against next or previous lab stages. A potentially important stream for future work might be to examine the use of electronic assessment alongside WLab. Initially, this would probably occur simply by including appropriate tools within WLab tutor VMs. Later, the WLab environment itself might integrate these tools more directly.

Other future development work, not directly implied by Figure 30, may seek to modify the composite learning object for different contexts. Unlike many virtual lab environments, WLab does not seek to provide a solution for a specific domain (e.g. security or networking). If a workshop exercise can be performed at a computer, then conceivably, a WLab lab can be authored for it. However, this genericism does mean that, for certain domains within ICT teaching such as security, more specialised solutions such as V-NetLab might be a better fit [12, 23].

Consider a workshop exercise in IT security where the student has two machines, a "hacker" machine and a "server" machine. This does not really fit WLab's staged paradigm, where a learning object has several virtual machines, but they are intended to provide a description of the same entity over time and do not run concurrently. To deliver this workshop in WLab, one would need to author two separate labs each with a single lab stage/VM. This is just one example of an ICT workshop scenario where a single virtual machine does not provide a rich enough environment for the student's activities during the exercise.

One might postulate an extended WLab composite learning object, where the virtual *machine* in a lab stage is replaced with a virtual *network* – i.e. a *group* of virtual machines. The lab console UI would be refactored to accommodate several VMs presented to the student simultaneously. The staged approach – along with its inherent pedagogic benefits – could still be utilised (although the effort involved in authoring the lab would be multiplied by the size of the virtual network).

Further extensions to the application can be anticipated in the administrative functionality. Basic functions exist for a tutor to manage students using WLab, but these functions would probably become unwieldy when dealing with large amounts of students and/or lab exercises. Additional functions to permit a tutor to handle a cohort rather than individual students would be desirable, as would extensions to the usage metrics currently available, which simply logs a student's progress through the lab stages. Extensions might attempt to correlate students' usage of WLab with other performance indicators, such the gradebooks available within VLEs.

The limitations imposed by the scalability of Hyper-V could be eliminated by the addition of functionality that would allow the use of several virtualisation servers. Assuming that the virtual machines were stored in a location accessible to all of the virtualisation servers (e.g. via a SAN) it would be trivial to arrange for the server currently under the least load to load and start a given VM.

Finally, another stream for future work would be to introduce "intelligence" into the application so that it "knows" when a student has reached the end of a stage, and automatically moves them onto the next. This could be done through the introduction of a rule set, created by the tutor when they author a lab, that looks for certain text strings within the VM console. When such text strings appear, a trigger could be sent that navigates to the next stage. Potentially, other triggers might be set up to recognise common mistakes being made, and update the static resources accordingly. This would lend itself to a phenomenographic approach, described in *Learning and Awareness* as the examination of "the variation in ways people experience phenomena in their world" [84]. Students will take different paths through a learning objective. Analysis of these different paths will reveal commonalities between them that *all* students must inevitably take, which can already be used as the basis of milestone points i.e. WLab's lab stages. The context-setting narrative supplied by the accompanying static resources, if it anticipates and discusses these potential paths, can also impart crucial learning material to students. Well-designed WLab learning objects should thus already be crafted with a phenomenographic approach at their core.

The intelligent WLab postulated would go further by allowing a tutor to design a lab that adapts itself based on the student's path. Currently, the path a student must take is linear, and if they deviate too much from the expected path the assumption is that they will eventually reach a point of diminishing returns, "give up", and simply navigate to the next lab stage whereby they are instantly provided with a "correct" environment with which to continue work. In contrast, the "intelligent" WLab described would allow for tutors to design labs that "branch", where learning content is only delivered if certain conditions were met. Consequently, new components within the WLab composite learning object are anticipated. A "blind alley" would represent an "incorrect" path. An "alternative route" would represent a non-optimal but ultimately functionally identical path. A tutor could design for these non-desirable paths and provide resource objects that discuss the differences between them and the optimal path, and guide the student in the right direction.

7 References

1. Beck K et al. *The Agile Manifesto* [online]. Available from http://www.agilemanifesto.org [Accessed 19th August 2010].
2. Wiley, DA. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. *The Instructional Use of Learning Objects* [online]. Association for Instructional Technology; 2001. Available from http://www.reusability.org/read/chapters/wiley.doc. [Accessed 4th May 2010].
3. Boisot M. *A Framework for Learning in Organisations, Institutions and Cultures*. London, UK: Routledge; 2001. p. 90-108.
4. Teo CB, Gay RKL. A Knowledge Driven Approach to Personalize E-Learning. *ACM Journal of Educations Resources in Computing* 2006; 6(1): p1-15.
5. Microsoft. *Common Language Runtime* [online]. Available from http://msdn.microsoft.com/en-us/library/8bs2ecf4.aspx. [Accessed 19th August 2010].
6. Linholm T, Yellin F. *The Java Virtual Machine Specification* [online]. Available from http://java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html. [Accessed 5th August 2010].
7. Williams DD. Evaluation of learning objects and instruction using learning objects. Wiley DA (Ed), *The Instructional Use of Learning Objects* [online]. Available from http://www.reusability.org/read/chapters/williams.doc. [Accessed 19th August 2010].
8. Neve P. *A General Approach to E-Learning for ICT Students*. Unpublished submission for MSc Informatics module *Research Methods* at Kingston University. 2009. Available from http://www.paulneve.com/cgi-bin/wiki.pl/A_General_Approach_to_E-Learning_for_ICT_Students.
9. Wilson K, Korn JH. Attention During Lectures: Beyond Ten Minutes. *Teaching of Psychology* 2007; 34(2): p85-89.
10. Bonwell CC, Eison JA. Active Learning: Creating Excitement in the Classroom. *ASHE ERIC Higher Education Report No. 1*. Available from http://www.ntlf.com/html/lib/bib/91-9dig.htm.
11. Mellor R, Mellor N. *Applied E-Learning*. Copenhagen, Denmark: Forlaget Globe; 2004. p30-31.
12. Gaspar A, Langevin S, Armitage WD. Virtualization Technologies in the Undergraduate IT Curriculum. *IEEE IT Pro*. July/August 2007: p. 10-17.
13. Popek GJ, Goldberg RP. Formal Requirements for Virtualizable Third Generation Architectures. *Communications of the ACM* 1974; 17(7): p. 412-421.
14. Varian M. VM and the VM Community: Past, Present, and Future. *SHARE 89 Sessions 9059-9061*. Princeton University, New Jersey, USA. August 1997. p3-25.
15. VMWare. *VMWare Milestones* [online]. Available from http://www.vmware.com/company/mediaresource/milestones.html [Accessed 16th June 2010].
16. IDC. *Virtualization Continues to See Strong Growth in Second Quarter* [online]. Available from http://www.idc.com/getdoc.jsp?containerId=prUS21473108 [Accessed 4th February 2010].
17. IBM Global Education. *Virtualization in Education* [online]. Available from http://www.ibm.com/solutions/in/education/download/Virtualization%20in%20Education.pdf [Accessed 4th February 2010].
18. McEwan W. Virtual machine technologies and their application in the delivery of ICT. Man S (Ed.), *Proceedings of the 15th annual conference of the New Zealand National Advisory Committee on Computing Qualifications (NACCQ)*. Hamilton, New Zealand. 2002. p55-62.
19. User Mode Linux Web Site [online] Available from http://user-mode-linux.sourceforge.net. [Accessed 16th June 2010].
20. Bullers WI, Burd S, Seazzu AF. Virtual Machines - An Idea Whose Time Has Returned: Application to Network, Security and Database Courses. *ACM SIGCSE Bulletin* 2006; 38(1). p102-106.
21. Anderson BR, Joines AK, Daniels TE. Xen Worlds: Leveraging Virtualization in Distance Education. *ACM SIGCSE Bulletin* 2009; 41(3): p293-297.
22. Krishna K, Sun W, Rana P, Li T, Sekar R. V-NetLab: A cost effective platform to support course projects in computer security. *9th Annual Colloquium for Information Systems Security Education*. Atlanta, USA. 2005.

23. Gaspar A, Langevin S, Armitage W, Rideout M. Enabling new pedagogies in operating systems and networking courses with state of the art open source kernel and virtualization technologies. *Journal of Computing Sciences in Colleges* 2008; 23: p189-198.

24. Gaspar A. *SOFTICE Project Wiki: Pedagogical Resources* [online]. Available from http://softice.lakeland.usf.edu/wiki/index.php/Category:Pedagogical_Resources. [Accessed 4th May 2010].

25. Microsoft. System Center Virtual Machine Manager web site [online]. Available from http://www.microsoft.com/systemcenter/en/us/virtual-machine-manager.aspx. [Accessed 19th August 2010].

26. Cockburn A. *Crystal Clear: A Human Powered Methodology for Small Teams*. Addison Wesley; 2004. p140.

27. Deemer P, Benefield G, Larman C. The Scrum Primer [online]. Available from http://www.scrumalliance.org/resource_download/339. [Accessed 10th May 2010].

28. DSDM Consortium. *Atern Full Product Set* [online]. Available from http://www.dsdm.org/knowledgebase/download/156/atern_full_product_set.pdf [Accessed 10th May 2010].

29. Aternity Solutions. *Atern on a page* [online]. Available from http://www.dsdm.org/knowledgebase/download/205/atern_on_a_page.jpg [Accessed 10th May 2010].

30. DSDM Consortium. *DSDM Atern Handbook*. Whitstable, Kent, UK: Whitehorse Press; 2008. p83-87.

31. Ambler SW. *The Agile Unified Process* [online]. Available from http://www.ambysoft.com/downloads/agileUP.zip [Accessed 11th May 2010].

32. IBM. *IBM Rational Unified Process* [online]. Available from ftp://ftp.software.ibm.com/software/rational/web/datasheets/RUP_DS.pdf [Accessed 11th May 2010].

33. Ash S. *MoSCoW Prioritisiation Briefing Paper* [online]. Available from http://www.dsdm.org/knowledgebase/download/165/moscow_prioritisation_briefing_paper.doc [Accessed 12th May 2010].

34. DSDM Consortium. *DSDM Atern Handbook*. Whitstable, Kent, UK: Whitehorse Press; 2008. p63-68.

35. Cockburn A. *Crystal Clear: A Human Powered Methodology for Small Teams*. Addison Wesley; 2004. p181-182.

36. DSDM Consortium. *DSDM Atern Handbook*. Whitstable, Kent, UK: Whitehorse Press; 2008. p78.

37. Beck K. *Extreme Programming Explained*. Addison-Wesley; 2000. p31.

38. Cockburn A. *Writing Effective Use-Cases*. Addison-Wesley; 2001. p7-9.

39. Cockburn A. *Crystal Clear: A Human Powered Methodology for Small Teams*. Addison Wesley; 2004. p174-175

40. DSDM Consortium. *DSDM Atern Handbook*. Whitstable, Kent, UK: Whitehorse Press; 2008. p69-75.

41. Beck K. *Extreme Programming Explained*. Addison-Wesley; 2000. p20.

42. Cockburn A. *Information Radiator* [online]. Available from http://alistair.cockburn.us/Information+radiator [Accessed 14th May 2010]

43. Jeffries R. Big Visible Charts [online]. *XP Magazine*. Available from http://xprogramming.com/xpmag/BigVisibleCharts [Accessed 14th May 2010]

44. Abrahamsson P, Warsta J, Siponen MT, Rokainen J. New Directions on Agile Methods: A Comparative Analysis. *Proceedings of the 25th International Conference on Software Engineering* Portland, Oregon, USA. 2003. p244-254.

45. Waters K. *10 Key Principles of Agile Software Development* [online]. Available from http://www.agile-software-development.com/2007/02/10-things-you-need-to-know-about-agile.html [Accessed 14th May 2010].

46. Aftab T. *Requirement Modeling in Agile Framework* [online]. Available from http://www.ephlux.com/Whitepapers/Requirement_Modeling.pdf [Accessed 14th May 2010].

47. Novell. *Mono Project web site* [online]. Available from http://www.mono-project.com. [Accessed 17th June 2010].

48. Rosen A, de Icaza M. Forum discussion*: Is Mono ready for prime time?* [online] Available from http://stackoverflow.com/questions/18450/is-mono-ready-for-prime-time/93952#93952. [Accessed 17th June 2010].

49. Smith B, Free Software Foundation. *Microsoft's Empty Promise* [online]. Available from http://www.fsf.org/news/2009-07-mscp-mono. [Accessed 17th June 2010].

50. Walls C. *Spring in Focus*. Greenwich, CT, USA: Manning; 2008.

51. Carmalt P, Elliot T. *properJavaRDP web site* [online]. Available from http://properjavardp.sourceforge.net/ [Accessed 18th May 2010].

52. AT&T Research. *Archive copy of the original VNC website* [online]. Available from http:// www.hep.phy.cam.ac.uk/vnc_docs/index.html [Accessed 18th May 2010].

53. TightVNC web site [online]. Available from http://www.tightvnc.com [Accessed 18th May 2010].

54. Ort E, Mehta B. *Java Architecture for XML Binding (JAXB)* [online]. Available from http://java.sun.com/developer/technicalArticles/WebServices/jaxb/ [Accessed 10th June 2010].

55. JiBX project web site [online]. Available from http://jibx.sourceforge.net/. [Accessed 10th June 2010]

56. Sosnoski D. *JiBX 1.2, Part 1: Java code to XML schema* [online]. Available from http://www.ibm.com/developerworks/java/tutorials/j-jibx1/. [Accessed 11th June 2010].

57. Microsoft Corporation. *PowerShell Management Library for Hyper-V* [online]. Available from http://pshyperv.codeplex.com/team/view [Accessed 15th May 2010].

58. Vaibhav/Sun Microsystems. *Not as easy as we thought - PowerShell from Java Runtime* [online]. Available from http://blogs.sun.com/vaibhav/entry/not_as_easy_as_we. [Accessed 16th May 2010].

59. JWBem Project web site [online]. Available from http://jwbem.sourceforge.net/. [Accessed 17th May 2010].

60. J-Interop Project web site [online]. Available from http://www.j-interop.org/. [Accessed 17th May 2010].

61. Sun Microsystems. *Java Naming and Directory Interface (JNDI)* [online]. Available from http://java.sun.com/products/jndi/. [Accessed 20th May 2010].

62. stackoverflow.com. Forum thread: *Authenticating against Active Directory with Java on Linux*. [online] Available from http://stackoverflow.com/questions/390150/authenticating-against-active-directory-with-java-on-linux. [Accessed 17th May 2010]

63. UltraVNC project team. *UltraVNC Repeater* [online]. Available from http://www.uvnc.com/addons/repeater.html [Accessed 18th May 2010].

64. Argyroudis P. Jumpgate web site [online]. Available from http://jumpgate.sourceforge.net [Accessed 20th May 2010].

65. Microsoft Corporation. *How to use the Sysprep tool to automat*e successful deployment of Windows XP [online]. Available from http://support.microsoft.com/default.aspx?scid=kb;en-us;302577 [Accessed 20th May 2010]

66. Microsoft Technet. *The Soul of a Virtual Machine Blog: Sysprepping a virtual machine* [online]. Available from http://blogs.technet.com/b/megand/archive/2005/01/20/357570.aspx [Accessed 23rd May 2010].

67. Clarke D. *Workstation Name Changer* [online]. Available from http://mystuff.clarke.co.nz/MyStuff/wsname.asp [Accessed 23rd May 2010].

68. O'Reilly D. *Workers' Edge: Shut down Windows in an instant* [online]. Available from http://www.cnet.com/8301-13880_1-9900788-68.html?part=rss [Accessed 24th May 2010]

69. DSDM Consortium. *DSDM Atern Handbook*. Whitstable, Kent, UK: Whitehorse Press; 2008. p105.

70. Pilone D. *UML 2.0 Pocket Reference*. O'Reilly; 2006. p72-73.

71. Ambler S. *The Principles of Agile Modelling* [online]. Available from http://www.agilemodeling.com/principles.htm#ModelWithAPurpose [Accessed 19th August 2010].

72. Larman S. *Applying UML and Patterns,* Second Edition. New Jersey, USA: Prentice-Hall; 2002. p221-226

73. Oracle. *What is an interface?* [online]. Available from http://download.oracle.com/javase/tutorial/java/concepts/interface.html [Accessed 19th August 2010].

74. Oracle. *Creating Objects* [online]. Available from http://download.oracle.com/javase/tutorial/java/javaOO/objectcreation.html [Accessed 19th August 2010].

75. ALT-C 2010 web site [online]. Available from http://www.alt.ac.uk/altc2010/ [Accessed 19th August 2010].

76. IMS Global Learning Consortium. *Content Packaging Specification* [online]. Available from http://www.imsglobal.org/content/packaging/ [Accessed 19th August 2010].

77. Jorum website [online]. Available from http://www.jorum.ac.uk/ [Accessed 19th August 2010].

78. TightVNC project team. *TightVNC Java Viewer with SSH Tunnelling* [online]. Available from http://www.tightvnc.com/ssh-java-vnc-viewer.php [Accessed 19th August 2010].

79. DSDM Consortium. *DSDM Atern Project Approach Questionnaire* [online]. Available from http://www.dsdm.org/public/File/pdf/Atern%20Project%20Approach%20Questionnaire.xls [Accessed 19th August 2010].

80. "Uncle Bob". The Scatology of Agile Architecture. *Object Mentor* [online]. Available from http://blog.objectmentor.com/articles/2009/04/25/the-scatology-of-agile-architecture [Accessed 19th August 2010].

81. Appleton B. JEDI Programming – Just Enough Design Initially. *Agile Journal* [online]. Available from http://www.agilejournal.com/blogs/blogs/brad-appletons-acme-blog-mainmenu-77/2003-jedi-programming-just-enough-design-initially [Accessed 19th August 2010].

82. Palmer S. *Feature Driven Development: An Introduction* [online]. Available from http://www.step-10.com/SoftwareProcess/FeatureDrivenDevelopment/introduction.html [Accessed 19th August 2010].

83. Beck K et al. *Principles Behind the Agile Manifesto* [online]. Available from http://www.agilemanifesto.org/principles.html [Accessed 19th August 2010].

84. Marton F, Booth S. *Learning and Awareness*. Marhwah, New Jersey, USA: Lawrence Erlbaum Associates; 1997. p121.

Appendix A – Metaphors

The following metaphors were established at the start of the project. The intention was for these to provide a common lexicon through the project, and also to supply an initial "mission" statement for each project phase. The latter aspect is fleshed out and in some respect, superseded by the artefacts of the Inception and Elaboration process.

Metaphor: Stage 1

We will build a system for the delivery of virtual machines to students to assist in the teaching of advanced ICT subjects, reverse engineering the current VLab system using open solutions and technologies so as to remove the current dependencies on Microsoft technology at both server and client sides.

A ***student*** *is allocated one or more* ***virtual machines****, pre-configured by the* ***tutor****. The student is provided with a web-based front end from which they may access the system. This front end will be accessible either from university equipment or remotely, accessible from as many platforms and web browsers as is possible, and require zero configuration or software installation on the student's part.*

Upon logging in, the student will be presented with a list of their virtual machines, and given options to start or suspend any of them. If a machine is running, they may also connect to it, which will produce a page in their web browser with the virtual machine's console embedded.

The system will manage capacity by limiting concurrent connections. Each student may only start a pre-defined, per student allocated number of virtual machines at a time. A student will also be allocated a maximum amount of time that they can run lab sessions for; after this period their sessions will be suspended and they will not be permitted to restart them until a specified time has elapsed. A student may also not start a virtual machine if there is already more than a maximum, per server amount running.

Metaphor: Stage 2

We will build a system that incorporates the features listed in Metaphor: Stage 1. *Additionally, there will be the ability for a student to create a* ***booking****. This is a period of runtime at a specified, fixed time and date where they are guaranteed the ability to run a virtual machine.*

The capacity management arrangements change slightly from that previously stated: The system will manage capacity by limiting concurrent connections. Each student may only start a pre-defined, per student allocated number of virtual machines at a time. Unless they have a booking, a student will also be allocated a maximum amount of time that they can run lab sessions for. After this period their sessions will be suspended and they will not be permitted to restart them until a specified time has elapsed. A student may also not start a virtual machine if the number of machines already running, plus the number of bookings at the current time exceeds a maximum, per server amount.

Metaphor: Stage 3

We will build a system for the delivery of lab exercises to assist in the teaching of advanced ICT subjects, building on the lessons learnt from the existing VLab system. The system will follow the workshop paradigm often used in teaching these subjects, where conventional lecturing is supplemented by workshop sessions; during such sessions students are set practical tasks to complete that exercise the knowledge delivered.

A ***lab*** *will contain a* ***virtual machine****. In this context a virtual machine means an exemplar image, pre-configured by the tutor, containing both the software and data required by a student to perform the lab exercise.*

The system will provide an interface for a ***student*** *to access the labs published to them by a* ***tutor****. This interface will be web-based, accessible either from university equipment or remotely, accessible from as many platforms and web browsers as is possible, and require* *zero configuration or software installation on the student's part. When a student attempts a lab for the first time, the lab's own virtual machine is used as the basis to clone a new, student specific VM. This new VM belongs specifically to the student and is where they will perform the tasks expected of them during the lab stage. As a student works through a lab, their* ***progress*** *is recorded.*

The student will be able to book a guaranteed lab session, as per Metaphor: Stage 2.

The system will manage capacity by limiting concurrent connections. Each student may only start a pre-defined, per student allocated number of virtual machines at a time. Unless they have a ***booking****, a student will also be allocated a maximum amount of time that they can run lab sessions for. After this period their sessions will be suspended and they will not be permitted to restart them until a specified time has elapsed. A student may also not start a virtual machine if the number of machines already running, plus the number of bookings at the current time exceeds a maximum, per server amount.*

Metaphor: Stage 4

We will build a system for the delivery of lab exercises to assist in the teaching of advanced ICT subjects, building on the lessons learnt from the existing VLab system. The system will follow the workshop paradigm often used in teaching such subjects, where conventional lecturing is supplemented by workshop sessions; during such sessions students are set practical tasks to complete that exercise the knowledge delivered.

A ***lab*** *will consist of a series of* ***lab stages****. Each individual lab stage will include a* ***tutor virtual machine (VM)****. This is an exemplar VM image, pre-configured by the tutor, containing both the software and data required by a student to perform this stage of the lab exercise, representing the starting point of this lab stage. Lab stages also contain* ***resources****, which take the form of static learning content relevant to the lab stage at hand. Resources are displayed to the student alongside the virtual environment for the lab stage.*
Lab stages will be ordered in a logical sequence that replicates the order that a subject expert would follow if they were to attempt to complete the task themselves.

The system will provide an interface for a ***student*** *to access the labs published to them by a* ***tutor****. This interface will be web-based, accessible either from university equipment or remotely, accessible from as many platforms and web browsers as is possible, and require zero configuration or software installation on the student's part. When a student accesses a lab stage for the first time, the tutor VM of the lab stage will be used as the basis to create a new, student specific VM. This new VM is where the student will perform the tasks expected of them during the lab stage. Additionally, for each lab stage, the tutor may specify one or more URLs to serve as the resources to be displayed alongside the VM console. As a student works through a lab, their* ***progress*** *is recorded.*

The student will be able to book a guaranteed lab session, as per Metaphor: Stage 2.

The ***system*** *will manage capacity by limiting concurrent connections. A student will be allocated a maximum amount of time that they can run lab sessions for; after this period their sessions will be suspended and they will not be permitted to restart them for the time specified. The student may create a* ***booking****, i.e. a specified period where they are guaranteed access. The system will check such bookings to ensure that capacity is not exceeded; a student will not be permitted to make a booking if other students have already done so up to this capacity.*

Metaphor: Stage 5

From the ***student's*** *perspective, the system will provide the features outlined in* Metaphor: Stage 4.

From the ***tutor's*** *perspective, we will build an additional sub-system to allow for the management of students,* ***labs****,* ***lab stages, tutor virtual machines (VMs)*** *and* ***resources****. The definitions of these terms are, again, as per* Metaphor: Stage 4.

A tutor may create or edit a lab. A screen will prompt them to choose from a number of existing lab stages that will comprise the lab. They can then list the students they wish to publish the lab to.

A tutor may create or edit a lab stage. A screen will prompt them to choose from a number of existing tutor VMs. For each lab stage one VM must be chosen to serve as its starting point.

A tutor may create or edit a tutor VM. A screen will prompt them to choose from the virtual machines that currently exist on the virtualisation backend. They may select an existing tutor VM or alternatively, a ***backend VM****. A backend VM is a virtual machine that exists solely on the virtualisation backend. They are created by a System Administrator, using the tools available via the virtualisation backend. They may only be edited using these tools and are not editable by tutors – they are intended to provide a foundation for a tutor to create tutor VMs.*

When a tutor chooses a virtual machine it is cloned. The tutor is then connected to the new clone, and they may tailor it to suit their needs. These needs may be dictated by the requirements of a specific lab stage; alternatively, a tutor may wish to author a VM solely to provide a pre-configured starting point for their future VMs (and thus avoid repeating the same configuration tasks every time they author a lab).

Appendix B – Use cases

Subfunctions are shown in **bold**, whereas links to other use cases are shown underlined.

Initial use cases

See section 4.1.1, Initial use cases, for further detail and discussion of the following.

Initial Use Cases – Stage 1

Use Case No	Use Case Name	Actor	Project Stage	Priority
1	**Change Virtual Machine State**	**Student**	**1**	**M**

- The student **logs into the system**.
- The **virtual machines are listed**, together with their current state – either started, suspended or stopped.
- The student selects to change the VM's state:
    - If the VM is currently stopped or suspended, they may **start the VM**.
    - If the VM is currently started, they may **suspend the VM**.
        - The student may only completely shut down the VM from the VM's console.

Alternative flows/exceptions:

- A virtual machine was previously suspended by the system as per use case #3, Shut down idling VMs, and the permitted time to restart has not yet elapsed:
    - the option to start the machine is suppressed and a prompt explains why

- There are already more than the defined maximum amount of VMs in the started state
    - The option to start the machine is suppressed and a prompt explains why

Use Case No	Use Case Name	Actor	Project Stage	Priority
2	**Connect to running virtual machine**	**Student**	**1**	**M**

- The student **logs into the system**.
- The **virtual machines are listed**, together with their current state – either started, suspended or stopped.
- The student selects a virtual machine from the list to connect to.
- The student's browser **displays the VM console**.

Alternative flows/exceptions:

- The connection to the VM is unsuccessful:
    - an appropriate error message is displayed

Use Case No	Use Case Name	Actor	Project Stage	Priority
3	**Shut down idling VMs**	**System**	**1**	**M**

- Each minute, the system polls all running VMs
- If a VM has been in the "started" state for a period of time greater than that permitted to its owning student, it is suspended.

Initial Use Cases – Stage 2

Use Case No	Use Case Name	Actor	Project Stage	Priority
4	**Book Lab Session**	**Student**	**2**	**M**

- The student **logs into the system**.
- The student selects "book lab session".
- The student is **shown a calendar**
- The student may **choose a date and time** from the calendar.
    - periods where there are already bookings that will equal or exceed defined server capacity are marked as unavailable.
- This date and time is allocated to the student as a guaranteed period where they will be able to start and access a VM, regardless of any other considerations.

Use Case No	Use Case Name	Actor	Project Stage	Priority
1 v2	**Change Virtual Machine State**	**Student**	**2**	**M**

- The student **logs into the system**.
- The **virtual machines are listed**, together with their current state – either started, suspended or stopped.
- The student selects to change the VM's state:
    - If the VM is currently stopped or suspended, they may **start the VM**.
    - If the VM is currently started, they may **suspend the VM**.
        - The student may only completely shut down the VM from the VM's console.

Alternative flows/exceptions:

- The student attempts to change the machine state but is unsuccessful:
    - an appropriate error message is displayed

- A virtual machine was previously suspended by the system as per use case #3, Shut down idling VMs, and the permitted time to restart has not yet elapsed:
    - the option to start the machine is suppressed and a prompt explains why, unless the current time/date corresponds with a previously booked lab session for this student.

- If the current number of started VMs plus the number of bookings for the current time/date is greater than the defined amount:
    - The option to start the machine is suppressed and a prompt explains why, unless the current time/date corresponds with a previously booked lab session for this student.

Use Case No	Use Case Name	Actor	Project Stage	Priority
3 v2	**Shut down idling VMs**	**System**	**2**	**M**

- Each minute, the system polls all running VMs
- If a VM has been in the "started" state for a period of time greater than that permitted to its owning student, it is suspended, unless its owning student has a previously booked lab session.

Initial Use Cases – Stage 3

Use Case No	Use Case Name	Actor	Project Stage	Priority
5	**Do lab**	**Student**	**3**	**S**

- The student **logs into the system**.
- The server checks for a booking and/or whether current server capacity can accommodate them.
- The student is presented with the **list of lab exercises** available to them and invited to select one.
- At this point, a "lab exercise" is simply a base virtual machine image.
- When one is selected, the student is connected to it:
    - The system checks to see if a virtual machine already exists for this student/lab combination. If not:
        - A **VM is cloned** from the lab's base image.
        - The new VM is given a unique name and allocated to the student.
    - The student's VM is **started**.
    - The student may then connect to the lab. The lab's **VM console is displayed** in the student's web browser.
- When the student has finished the lab
    - The VM is suspended.

Alternative flows/exceptions:

- Server is already at maximum capacity on login (i.e. the student has not booked a session, and concurrent usage plus current bookings is equal to or greater than the set max):
    - an appropriate error message is displayed
    - the student is invited to book a lab session instead.

Initial Use Cases – Stage 4

Use Case No	Use Case Name	Actor	Project Stage	Priority
5 v2	Do lab	Student	4	C

- The student **logs into the system**.
- The server checks for a booking and/or whether current server capacity can accommodate them.
- The student is presented with the **list of lab exercises** available to them and invited to select one.
    - A lab consists of a series of stages. Each stage consists of:
        - A single virtual machine designed by the tutor (a "tutor VM")
        - One or more "resources" in the form of a web page URL
- When one is selected:
    - The system checks to see if the student has already undertaken this lab. If so, the initial stage is set to be wherever they left off. If not, the initial stage is set to be the first of the lab.
    - The system checks to see if a virtual machine already exists for this student/lab stage combination. If not:
        - A **VM is cloned** for them from the tutor VM.
        - The new VM is given a unique name and assigned to the student.
    - The student's VM for this lab stage is **started**.
    - The system **displays the VM console** for the newly created VM, alongside the contents of the resource web site(s).
    - The student may navigate between the web resources.
    - The student may navigate to a different lab stage. This will prompt the same check as previously to see whether an appropriate VM already exists, and a new one will be created if not.

Alternative flows/exceptions

- Server is already at maximum capacity on login (i.e. the student has not booked a session, and concurrent usage plus current bookings is equal to or greater than the set max):
    - an appropriate error message is displayed
    - the student is invited to book a lab session instead.

Initial Use Cases – Stage 5

Use Case No	Use Case Name	Actor	Project Stage	Priority
6	Create/edit lab stage	Tutor	5	C

- The tutor is presented with a **list of lab stages**. They may select an existing one to edit, or create a new one.
- The tutor is presented with a **list of tutor VMs.**
    - The list can be **filtered**.
    - If a new VM is required, a route to create a new tutor VM is available
- The tutor selects a VM to be the lab stage's base
- The tutor enters one or more web URLs as resources for the stage
- The tutor **modifies metadata for the stage**
- The tutor may save or discard the stage.

Use Case No	Use Case Name	Actor	Project Stage	Priority
7	**Create/edit a tutor VM**	**Tutor**	**5**	**C**

- The tutor is presented with the current **list of tutor VMs**. They can choose one to edit, or create a new one.
    - The list can be **filtered**.
- If they create a new one:
    - The tutor is presented with the current **list of backend system VMs**.
    - The tutor selects an existing VM to be the base for their new one
    - A new VM is **dynamically created** using the existing one as a base.
- The chosen VM, or the newly created clone, is **started** and a console connection established (via the existing **Display VM console** functionality).
- The tutor tailors the new VM to suit their needs. During this stage:
    - The tutor may take a snapshot of the VM's current state
    - The VM may revert the VM's state back to the snapshot, if one is present.
- The tutor edits metadata for the VM.
- The tutor shuts down the new VM, which is now ready to be used as a start point for a lab stage.

Use Case No	Use Case Name	Actor	Project Stage	Priority
8	**Create/edit lab**	**Tutor**	**5**	**C**

- The tutor is presented with a **list of existing labs**, from which they are invited to select one. Or they may instead choose to create a new blank lab.
    - The list can be **filtered.**
- The tutor is presented with a **list of lab stages:**
    - The list can be **filtered**.
- The tutor may add, remove or re-order **(modify) stages** in the lab.
- The tutor **adds metadata for the lab.**
- The tutor may save or discard the lab.

Use Case No	Use Case Name	Actor	Project Stage	Priority
9	**Publish lab to students**	**Tutor**	**5**	**C**

- The tutor is presented with a **list of existing labs**, from which they are invited to select one. Or they may instead choose to create a new blank lab.
    - The list can be **filtered**.
- The tutor is presented with a **list of students**.
    - The list can be **filtered**
- The tutor may add, remove or re-order students who may access the lab.

NB: This use case is obsoleted by changes that were generated during the Elaboration phase.

Updates to use cases generated during the Elaboration stage

Use Case No	Use Case Name	Actor	Project Stage	Priority
3 v3	**Shut down idling VMs**	**System**	**2**	**M**
• Each minute, the system polls all running VMs • If a VM has been in the "started" state for a period of time greater than that permitted to its owning student, all of the student's currently running VMs are suspended, unless its owning student has a previously booked lab session.				

Use Case No	Use Case Name	Actor	Project Stage	Priority
5 v3	**Do lab**	**Student**	**4**	**C**
• The student **logs into the system**. • The server checks for a booking and/or whether current server capacity can accommodate them. • The student is presented with the **list of lab exercises** available to them and invited to select one. o A lab consists of a series of stages. Each stage consists of: o A single virtual machine designed by the tutor (a "tutor VM") o One or more "resources" in the form of a web page URL • When one is selected: o The system checks to see if the student has already undertaken this lab. If so, the initial stage is set to be wherever they left off. If not, the initial stage is set to be the first of the lab, and the progress status flag for each lab stage is set to "Not started yet." o The system checks to see if a virtual machine already exists for this student/lab stage combination. If not: o A **VM is cloned** for them from the tutor VM. o The new VM is given a unique name and assigned to the student. o The student's VM for this lab stage is **started**. o A snapshot of the VM's starting point is taken. o The system **displays the VM console** for the newly created VM, alongside the contents of the resource web site(s). o The student is shown their progress status flag for this lab stage, and may **Set progress flag for lab stage.** o The student may navigate between the web resources. o The student may **take a snapshot** of the VM's current state o The student may **revert (to snapshot)** the VM's state back to the snapshot, if one is present. o The student may **revert (to snapshot)** the VM's state back to the start point snapshot. o The student may navigate to a different lab stage. This will prompt the same check as previously to see whether an appropriate VM already exists, and a new one will be created if not. **Alternative flows/exceptions** • Server is already at maximum capacity on login (i.e. the student has not booked a session, and concurrent usage plus current bookings is equal to or greater than the set max): o an appropriate error message is displayed o the student is invited to book a lab session instead.				

Use Case No	Use Case Name	Actor	Project Stage	Priority
8 v2	**Create/edit lab**	**Tutor**	**5**	**C**

- The tutor is presented with a **list of existing labs**, from which they are invited to select one. Or they may instead choose to create a new blank lab.
  - The list can be **filtered.**
- The tutor is presented with a **list of lab stages:**
  - The list can be **filtered**.
- The tutor may add, remove or re-order (**modify) stages** in the lab.
- The tutor **adds metadata for the lab.**
- The tutor **edits the student list** for the lab.
- The tutor may save or discard the lab.

Comments:
The previously distinct use case 9 has been integrated into this revision of use case 8.

Use Case No	Use Case Name	Actor	Project Stage	Priority
6 v2	**Create/edit lab stage**	**Tutor**	**5**	**C**

- The tutor is presented with a l**ist of lab stages**. They may select an existing one to edit, or create a new one.
- The tutor is presented with a **list of tutor VMs.**
  - The list can be **filtered**.
  - If a new VM is required, a route to create a new tutor VM is available
- The tutor selects a VM to be the lab stage's start point
- The tutor may select a VM to be the lab stage's end point
- The tutor enters one or more web URLs as resources for the stage
- The tutor **modifies metadata for the stage**
- The tutor may save or discard the stage.

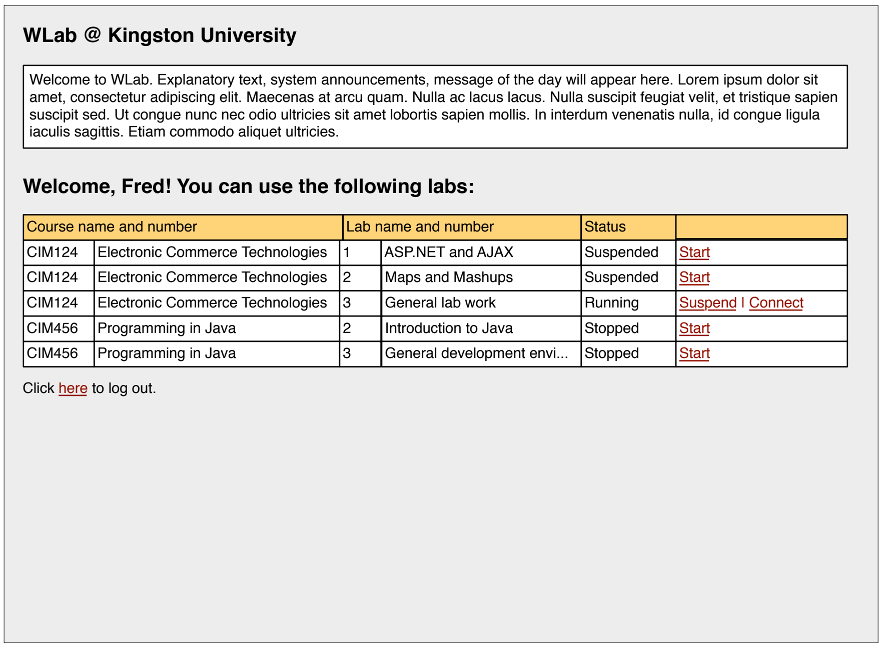
Appendix C – Screen Mockups

Screen mockups

The first "prototype" delivered to the user community during the elaboration phase was the following series of screen mockups. These were used to describe the expected functionality that would be delivered during each development stage, and to illustrate to the user community how the final application might work.

Phase 1

It is assumed that login has already taken place. Thus, this is the first screen seen by the student:



WLab @ Kingston University

Welcome to WLab. Explanatory text, system announcements, message of the day will appear here. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas at arcu quam. Nulla ac lacus lacus. Nulla suscipit feugiat velit, et tristique sapien suscipit sed. Ut congue nunc nec odio ultricies sit amet lobortis sapien mollis. In interdum venenatis nulla, id congue ligula iaculis sagittis. Etiam commodo aliquet ultricies.

Welcome, Fred! You can use the following labs:

Course name and number		Lab name and number		Status	
CIM124	Electronic Commerce Technologies	1	ASP.NET and AJAX	Suspended	Start
CIM124	Electronic Commerce Technologies	2	Maps and Mashups	Suspended	Start
CIM124	Electronic Commerce Technologies	3	General lab work	Running	Suspend I Connect
CIM456	Programming in Java	2	Introduction to Java	Stopped	Start
CIM456	Programming in Java	3	General development envi...	Stopped	Start

Click here to log out.

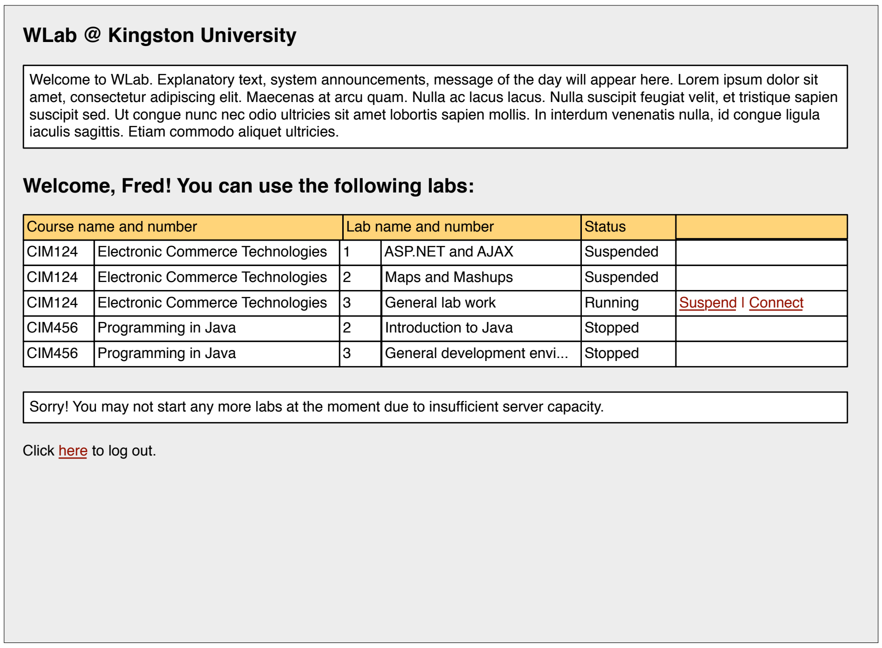
Figure 31 – initial post-login screen mockup

This is similar in some respects to the current VLab system. However, in contrast to the current VLab system which displays *all* VMs regardless, the expectation is that WLab will only display VMs relevant to (i.e. owned by) the student who is logged in.

There is also a paradigm shift versus VLab in that each student now has *labs*, rather than a single VM object that they can stop and start. This is a foreshadowing of the composite learning object that is the ultimate objective of WLab. However, at this stage a "lab" will just be a virtual machine. However, a student may have many labs (in contrast to VLab where each student has only a single VM).

The mockup also extends the information displayed to the student by providing appropriate metadata for each lab. Without such metadata, a student would find it very hard to know which lab was which.

As is the case on the current system, the student may **start** a stopped or suspended lab, and **suspend** a started lab. Or, a final option at the bottom of the page invites them to log out. Any running labs will continue to run if not suspended or shut down before logout.



WLab @ Kingston University

Welcome to WLab. Explanatory text, system announcements, message of the day will appear here. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas at arcu quam. Nulla ac lacus lacus. Nulla suscipit feugiat velit, et tristique sapien suscipit sed. Ut congue nunc nec odio ultricies sit amet lobortis sapien mollis. In interdum venenatis nulla, id congue ligula iaculis sagittis. Etiam commodo aliquet ultricies.

Welcome, Fred! You can use the following labs:

Course name and number		Lab name and number		Status	
CIM124	Electronic Commerce Technologies	1	ASP.NET and AJAX	Suspended	
CIM124	Electronic Commerce Technologies	2	Maps and Mashups	Suspended	
CIM124	Electronic Commerce Technologies	3	General lab work	Running	Suspend I Connect
CIM456	Programming in Java	2	Introduction to Java	Stopped	
CIM456	Programming in Java	3	General development envi...	Stopped	

Sorry! You may not start any more labs at the moment due to insufficient server capacity.

Click here to log out.

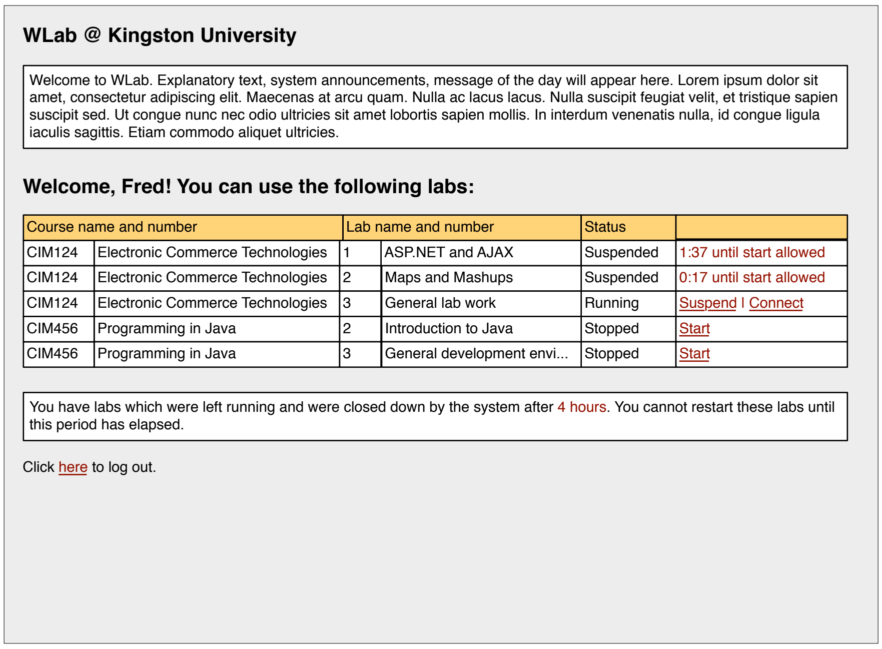
Figure 32 – server at maximum capacity mockup

The screen above shows the same user interface, but in the scenario where the server is already at maximum capacity. In this instance, any options to start labs are suppressed. Any currently running labs would still present the option to suspend or to connect to their console.

A similar screen would appear with the alternative text

Sorry! You may not start any more labs at the moment because you are already running your maximum number of concurrent sessions.

in the appropriate circumstance.



WLab @ Kingston University

Welcome to WLab. Explanatory text, system announcements, message of the day will appear here. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas at arcu quam. Nulla ac lacus lacus. Nulla suscipit feugiat velit, et tristique sapien suscipit sed. Ut congue nunc nec odio ultricies sit amet lobortis sapien mollis. In interdum venenatis nulla, id congue ligula iaculis sagittis. Etiam commodo aliquet ultricies.

Welcome, Fred! You can use the following labs:

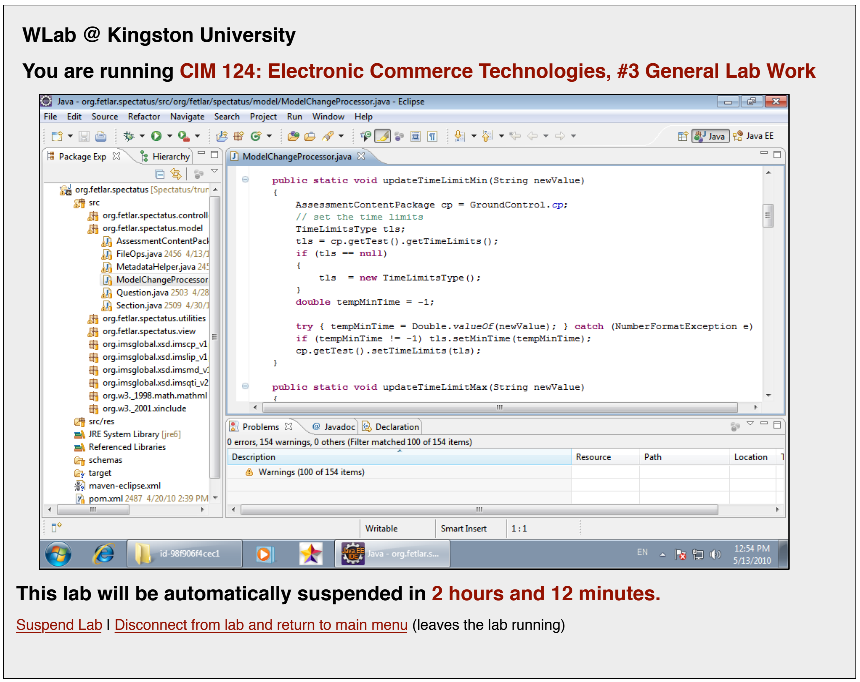
Course name and number		Lab name and number		Status	
CIM124	Electronic Commerce Technologies	1	ASP.NET and AJAX	Suspended	1:37 until start allowed
CIM124	Electronic Commerce Technologies	2	Maps and Mashups	Suspended	0:17 until start allowed
CIM124	Electronic Commerce Technologies	3	General lab work	Running	Suspend I Connect
CIM456	Programming in Java	2	Introduction to Java	Stopped	Start
CIM456	Programming in Java	3	General development envi...	Stopped	Start

You have labs which were left running and were closed down by the system after 4 hours. You cannot restart these labs until this period has elapsed.

Click here to log out.

Figure 33 – auto-suspended labs mockup

The screen above indicates what the student sees if labs are left running for more than the allotted runtime. The labs will be automatically suspended by the system, and the option to restart the labs suppressed until an appropriate time has elapsed.



WLab @ Kingston University

You are running CIM 124: Electronic Commerce Technologies, #3 General Lab Work

Java - org.fetlar.spectatus/src/org/fetlar/spectatus/model/ModelChangeProcessor.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Java EE
Package Exp
Hierarchy
org.fetlar.spectatus [Spectatus/trur
src
org.fetlar.spectatus.controll
org.fetlar.spectatus.model
AssessmentContentPack
FileOps.java 2456 4/13/1
MetadataHelper.java 245
ModelChangeProcessor
Question.java 2503 4/28
Section.java 2509 4/30/1
org.fetlar.spectatus.utilities
org.fetlar.spectatus.view
org.imsglobal.xsd.imscp_v1
org.imsglobal.xsd.imslip_v1
org.imsglobal.xsd.imsmd_v
org.imsglobal.xsd.imsqti_v2
org.w3._1998.math.mathml
org.w3._2001.xinclude
src/res
JRE System Library [jre6]
Referenced Libraries
schemas
target
maven-eclipse.xml
pom.xml 2487 4/20/10 2:39 PM
ModelChangeProcessor.java
Problems
Javadoc
Declaration
Writable
Smart Insert
1 : 1
id-98f906f4cec1
Java - org.fetlar.s...
EN
12:54 PM
5/13/2010

```java
public static void updateTimeLimitMin(String newValue)
{
    AssessmentContentPackage cp = GroundControl.cp;
    // set the time limits
    TimeLimitsType tls;
    tls = cp.getTest().getTimeLimits();
    if (tls == null)
    {
        tls  = new TimeLimitsType();
    }
    double tempMinTime = -1;

    try { tempMinTime = Double.valueOf(newValue); } catch (NumberFormatException e)
    if (tempMinTime != -1) tls.setMinTime(tempMinTime);
    cp.getTest().setTimeLimits(tls);
}

public static void updateTimeLimitMax(String newValue)
{
```

0 errors, 154 warnings, 0 others (Filter matched 100 of 154 items)

Description	Resource	Path	Location
Warnings (100 of 154 items)			

This lab will be automatically suspended in 2 hours and 12 minutes.

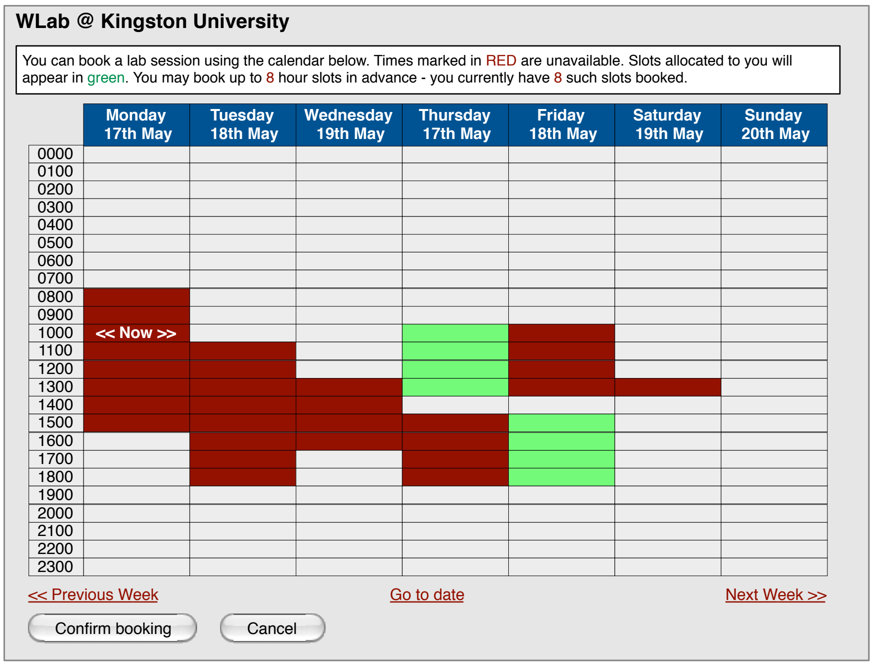
Suspend Lab | Disconnect from lab and return to main menu (leaves the lab running)

Figure 34 – the "in-lab" screen mockup

Finally, the screen above shows the proposed UI for when a student connects to a running lab session. The console of the virtual machine will be embedded within a web page that also shows the lab course details and title, and the remaining runtime available. The student is given the option to suspend the lab or to disconnect from the lab and return to the previous menu screen.

Phase 2

Phase 2 simply adds the ability for the student to book a guaranteed slot of lab time. All other features are unchanged.



WLab @ Kingston University

You can book a lab session using the calendar below. Times marked in RED are unavailable. Slots allocated to you will appear in green. You may book up to 8 hour slots in advance - you currently have 8 such slots booked.

	Monday 17th May	Tuesday 18th May	Wednesday 19th May	Thursday 17th May	Friday 18th May	Saturday 19th May	Sunday 20th May
0000							
0100							
0200							
0300							
0400							
0500							
0600							
0700							
0800	🟥						
0900	🟥						
1000	🟥 << Now >>			🟩	🟥		
1100	🟥	🟥		🟩	🟥		
1200	🟥	🟥		🟩	🟥		
1300	🟥	🟥	🟥	🟩	🟥	🟥	
1400	🟥	🟥	🟥				
1500	🟥	🟥	🟥	🟥	🟩		
1600		🟥	🟥	🟥	🟩		
1700		🟥		🟥	🟩		
1800		🟥		🟥	🟩		
1900							
2000							
2100							
2200							
2300							

Figure 35 – "booking a session" mockup

A student may book up to a defined number of hour-long sessions. The student may wish to allocate these in a sequential block or blocks to give a long guaranteed period of work, or they may choose to use them individually and have more regular albeit shorter work periods.

Times that are not available – e.g. if other students have already make bookings to capacity – are marked in red. The student's own bookings are marked in green. Appropriate navigation options are provided to move to a specific data, and if the currently displayed week intersects the current time/date, this is also displayed.

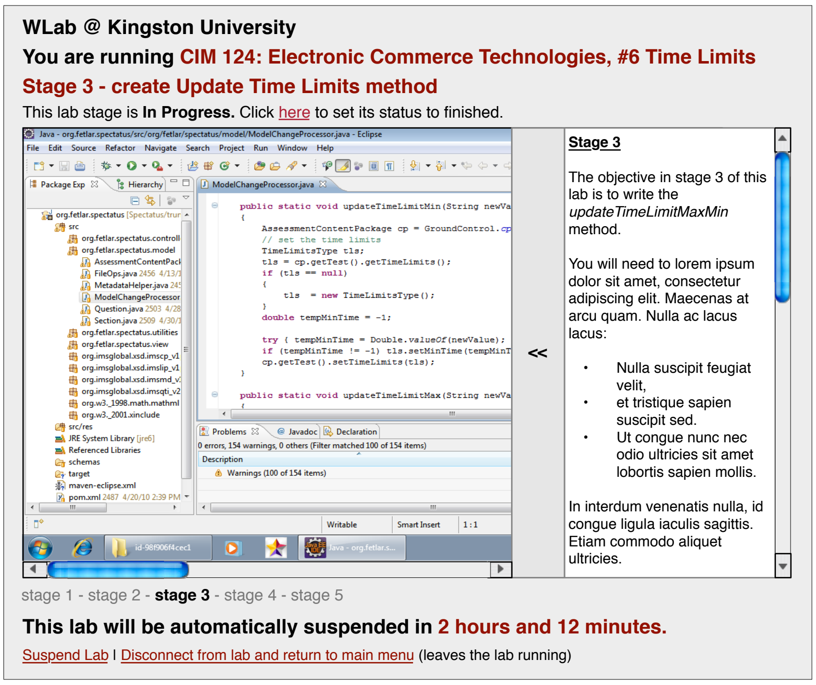
Apart from the provision of this calendar-based booking screen, for the most part this is expected to be the only addition to the UI in stage 2, apart from a link to the booking form on the "main" student screen.

Phase 3

Stage 3 does not require changes to the UI – the introduction of dynamically created virtual machines is a backend feature and will not change any user-facing aspects of the application.

Phase 4

Phase 4 introduces the concept of labs comprised of stages, with a stage consisting of a tutor VM and a number of accompanying web resources. Considerable changes will be required in the lab connection screen to accommodate this new concept:



WLab @ Kingston University

You are running CIM 124: Electronic Commerce Technologies, #6 Time Limits

Stage 3 - create Update Time Limits method

This lab stage is **In Progress.** Click here to set its status to finished.

```
public static void updateTimeLimitMin(String newVa
{
    AssessmentContentPackage cp = GroundControl.cp
    // set the time limits
    TimeLimitsType tls;
    tls = cp.getTest().getTimeLimits();
    if (tls == null)
    {
        tls  = new TimeLimitsType();
    }
    double tempMinTime = -1;

    try { tempMinTime = Double.valueOf(newValue);
    if (tempMinTime != -1) tls.setMinTime(tempMinT
    cp.getTest().setTimeLimits(tls);
}

public static void updateTimeLimitMax(String newVa
{
```

```
0 errors, 154 warnings, 0 others (Filter matched 100 of 154 items)
Description
Warnings (100 of 154 items)
```

Java - org.fetlar.spectatus/src/org/fetlar/spectatus/model/ModelChangeProcessor.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Exp
Hierarchy
ModelChangeProcessor.java
Problems
Javadoc
Declaration
Writable
Smart Insert
1 : 1
<<

Stage 3

The objective in stage 3 of this lab is to write the *updateTimeLimitMaxMin* method.

You will need to lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas at arcu quam. Nulla ac lacus lacus:

- Nulla suscipit feugiat velit,
- et tristique sapien suscipit sed.
- Ut congue nunc nec odio ultricies sit amet lobortis sapien mollis.

In interdum venenatis nulla, id congue ligula iaculis sagittis. Etiam commodo aliquet ultricies.

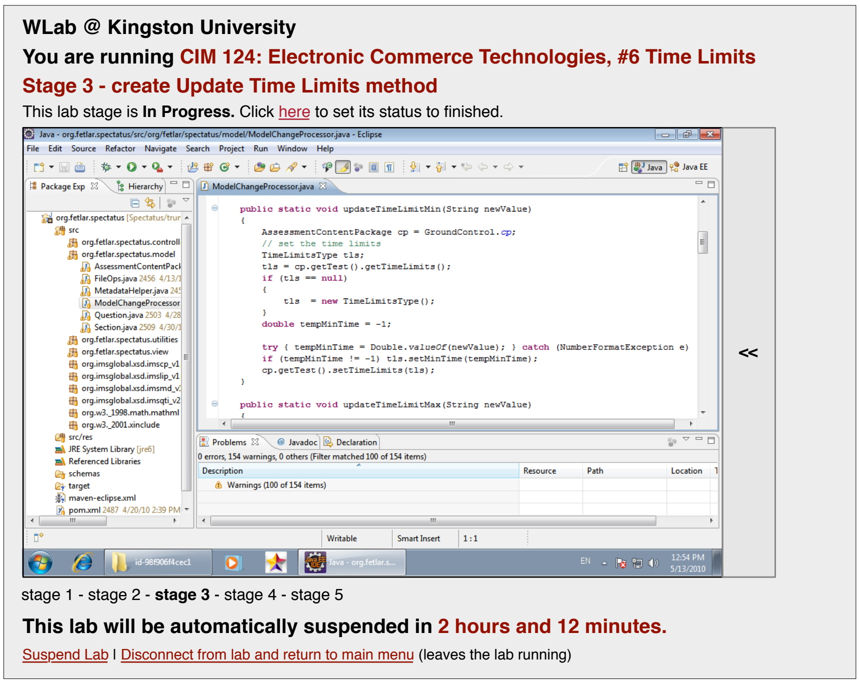
stage 1 - stage 2 - **stage 3** - stage 4 - stage 5

This lab will be automatically suspended in 2 hours and 12 minutes.

Suspend Lab | Disconnect from lab and return to main menu (leaves the lab running)

Figure 36 – the "in-lab" screen mockup including lab stages and resources

Use case 5 v2 indicates that the virtual machine and the web resource(s) should be displayed side-by-side. The mockup above shows how this would look. One issue that can be anticipated is the difficulty in fitting a VM console and other resources onto the same screen and provide adequate room to work in either. For this reason, the intention is that the resource pane will be collapsible, thus:



WLab @ Kingston University

You are running CIM 124: Electronic Commerce Technologies, #6 Time Limits

Stage 3 - create Update Time Limits method

This lab stage is **In Progress.** Click here to set its status to finished.

```
public static void updateTimeLimitMin(String newValue)
{
    AssessmentContentPackage cp = GroundControl.cp;
    // set the time limits
    TimeLimitsType tls;
    tls = cp.getTest().getTimeLimits();
    if (tls == null)
    {
        tls  = new TimeLimitsType();
    }
    double tempMinTime = -1;

    try { tempMinTime = Double.valueOf(newValue); } catch (NumberFormatException e)
    if (tempMinTime != -1) tls.setMinTime(tempMinTime);
    cp.getTest().setTimeLimits(tls);
}

public static void updateTimeLimitMax(String newValue)
{
```

0 errors, 154 warnings, 0 others (Filter matched 100 of 154 items)

Description	Resource	Path	Location
Warnings (100 of 154 items)			

stage 1 - stage 2 - **stage 3** - stage 4 - stage 5

This lab will be automatically suspended in 2 hours and 12 minutes.

Suspend Lab I Disconnect from lab and return to main menu (leaves the lab running)

Figure 37 - the "in-lab" screen mockup with resource pane collapsed

Clicking the divider will either collapse or display the resource pane, and thus give room to work in the console.

One key aspect here is the ability for the student to move from stage to stage. This is achieved by clicking on the links underneath the VM console. The currently running stage is highlighted in **bold black** text. Stage 1 in the example is shown in green, which indicates that it has been completed by the student.

The concept of stage "completion" arises from the two different reasons why a student may navigate to another stage:

1. They have completed all the tasks required of them, and simply want to move on to the next part of the lab exercise.
2. They are "stuck", and will need the solution to the current problem in order to be able to move on within a useful timescale.

To accommodate this, WLab will introduce the concept of stage "progress". Three such statuses are proposed:

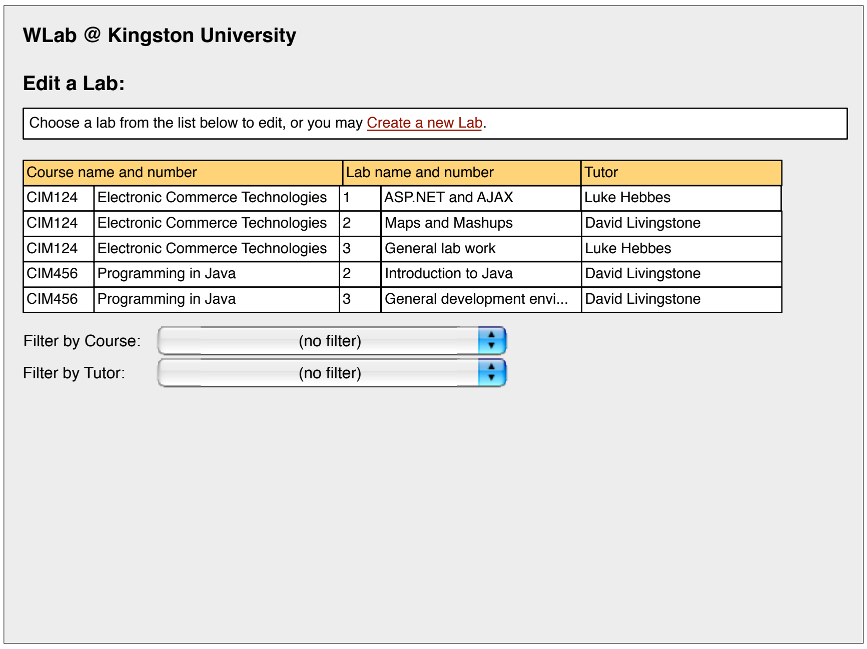
1. Not started yet
2. In progress
3. Finished

The first of these indicates that a student has yet to commence this part of the exercise; the other two are self-explanatory. When the student connects to a lab stage for the first time, it is automatically set to "In progress". When they feel they have finished the stage, they can set its status to "Finished".

Phase 5

Phase 5 of development is concerned with providing a management interface that allows tutors to manipulate lab, lab stage and tutor VM objects:

Create/edit lab



WLab @ Kingston University

Edit a Lab:

Choose a lab from the list below to edit, or you may Create a new Lab.

Course name and number		Lab name and number		Tutor
CIM124	Electronic Commerce Technologies	1	ASP.NET and AJAX	Luke Hebbes
CIM124	Electronic Commerce Technologies	2	Maps and Mashups	David Livingstone
CIM124	Electronic Commerce Technologies	3	General lab work	Luke Hebbes
CIM456	Programming in Java	2	Introduction to Java	David Livingstone
CIM456	Programming in Java	3	General development envi...	David Livingstone

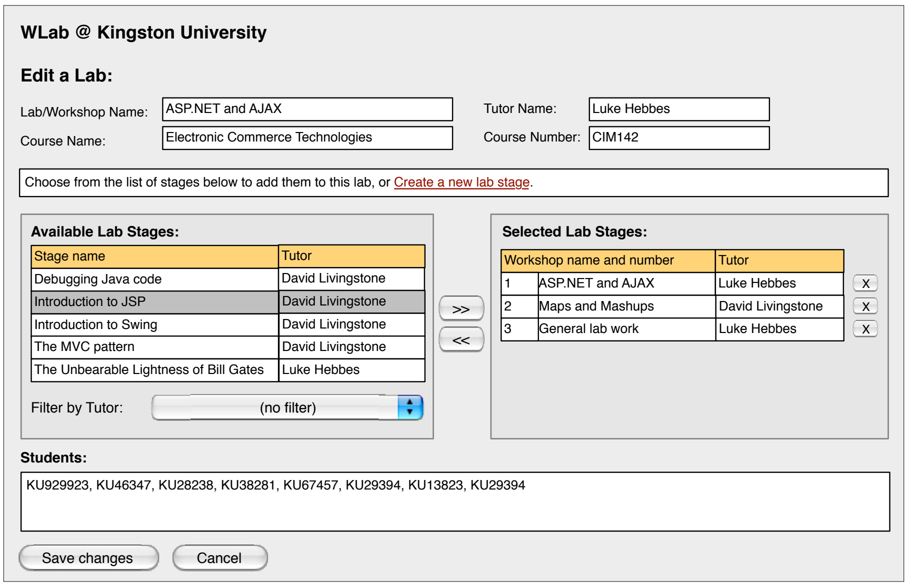
Filter by Course: (no filter)

Filter by Tutor: (no filter)

Figure 38 – create/edit lab screen mockup #1

The tutor is prompted to select an existing lab to edit, or they may create a new one. The list of labs can be filtered using pull down menus.

Upon selecting a particular lab, the tutor will see the following screen. Creating a lab would result in the same screen, except it would have no existing values in the metadata fields, and no lab stages would be selected:



WLab @ Kingston University

Edit a Lab:

Lab/Workshop Name: ASP.NET and AJAX

Tutor Name: Luke Hebbes

Course Name: Electronic Commerce Technologies

Course Number: CIM142

Choose from the list of stages below to add them to this lab, or Create a new lab stage.

Available Lab Stages:

Stage name	Tutor
Debugging Java code	David Livingstone
Introduction to JSP	David Livingstone
Introduction to Swing	David Livingstone
The MVC pattern	David Livingstone
The Unbearable Lightness of Bill Gates	Luke Hebbes

Filter by Tutor: (no filter)

Selected Lab Stages:

Workshop name and number		Tutor	
1	ASP.NET and AJAX	Luke Hebbes	X
2	Maps and Mashups	David Livingstone	X
3	General lab work	Luke Hebbes	X

Students:

KU929923, KU46347, KU28238, KU38281, KU67457, KU29394, KU13823, KU29394

Figure 39 – create/edit lab screen mockup #2

A number of metadata fields are provided so that this lab can be distinguished from others.

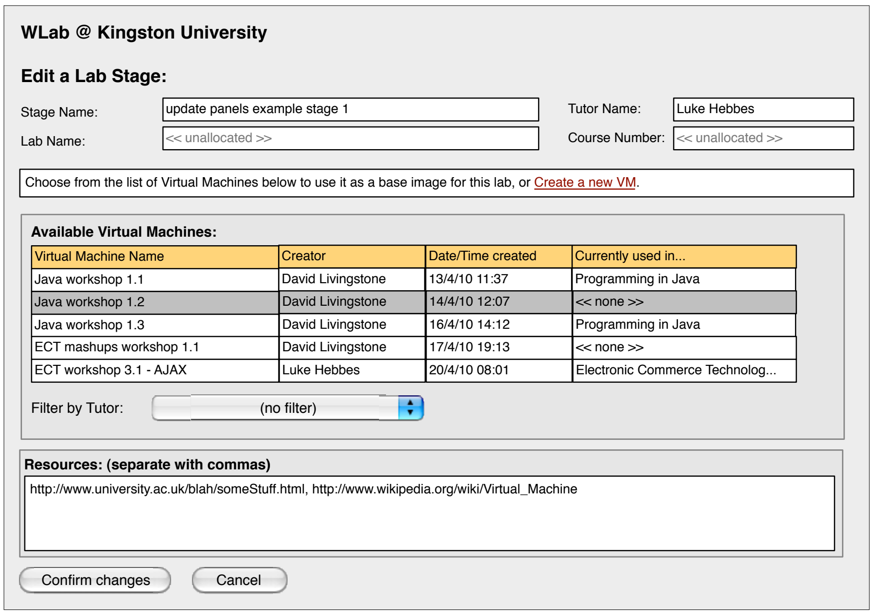
The tutor adds a stage from the left hand pane to the lab by highlighting it in the “Available lab stages” pane and clicking the $\boxed{>>}$ button. Similarly, a stage can be removed by highlighting it in the "Selected lab stages" pane and clicking the << button. Stages can be re-ordered by modifying their number; the "Selected lab stages" pane will update to reflect any new order. When the lab is delivered to the student, the stages will appear in this order.

The decision has been made to integrate use case 9, **Publish lab to students** with this screen. Upon further reflection it seems of little value to make this a distinct function. Therefore, this use case will take the form of a simple field added to the above screen into which student IDs can be entered. The act of adding a student ID to this field will immediately publish the lab to them.

When the tutor has finished making changes to the lab, they can either *Confirm Changes* or *Cancel*.

Create/Edit Lab Stage

The screen that follows shows the UI for use case 6, **Create/Edit Lab Stage**. If editing an existing stage, the tutor will select from a listing in the same way as shown for labs in Figure 38. When editing an existing lab stage, the fields in the screen shown will be populated; when creating a new lab stage the fields will appear blank in the first instance.


WLab @ Kingston University

Edit a Lab Stage:

Stage Name:	update panels example stage 1	Tutor Name:	Luke Hebbes
Lab Name:	<< unallocated >>	Course Number:	<< unallocated >>

Choose from the list of Virtual Machines below to use it as a base image for this lab, or Create a new VM.

Available Virtual Machines:

Virtual Machine Name	Creator	Date/Time created	Currently used in...
Java workshop 1.1	David Livingstone	13/4/10 11:37	Programming in Java
Java workshop 1.2	David Livingstone	14/4/10 12:07	<< none >>
Java workshop 1.3	David Livingstone	16/4/10 14:12	Programming in Java
ECT mashups workshop 1.1	David Livingstone	17/4/10 19:13	<< none >>
ECT workshop 3.1 - AJAX	Luke Hebbes	20/4/10 08:01	Electronic Commerce Technolog...

Filter by Tutor: (no filter)

Resources: (separate with commas)

http://www.university.ac.uk/blah/someStuff.html, http://www.wikipedia.org/wiki/Virtual_Machine

Figure 40 – create/edit lab stage screen mockup

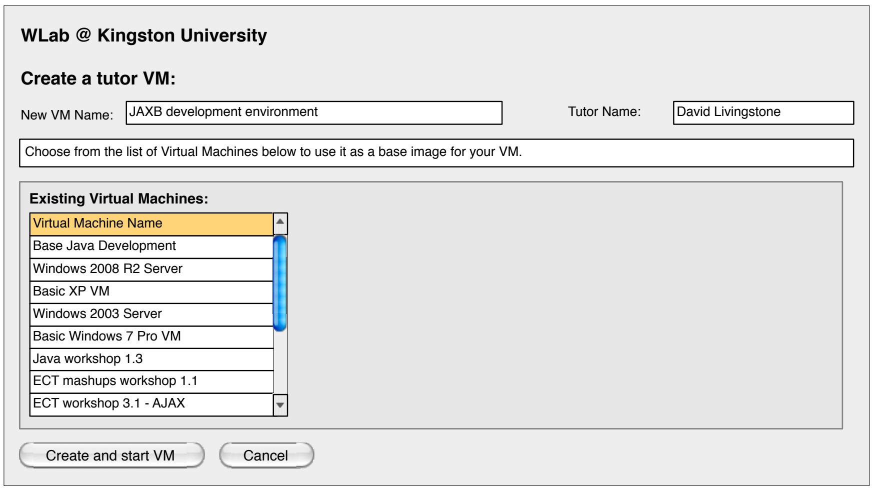
The tutor supplies a name for the stage. The course name and course number are not editable fields – they will be populated if the stage has previously been allocated to a lab.

The tutor is provided with a list of virtual machines, and they may select one that will form the basis of the lab stage. **Note that this list consists only of tutor VMs. This is *not* the full list of VMs present on the virtualisation backend.**

Finally, the tutor can provide a list of web URLs to be used as the accompanying resources for the stage. These are simply entered into the appropriate field and separated with commas.

When the tutor is finished, they select *Confirm changes*, or *Cancel*.

Create/edit tutor VM


WLab @ Kingston University

Create a tutor VM:

New VM Name: JAXB development environment

Tutor Name: David Livingstone

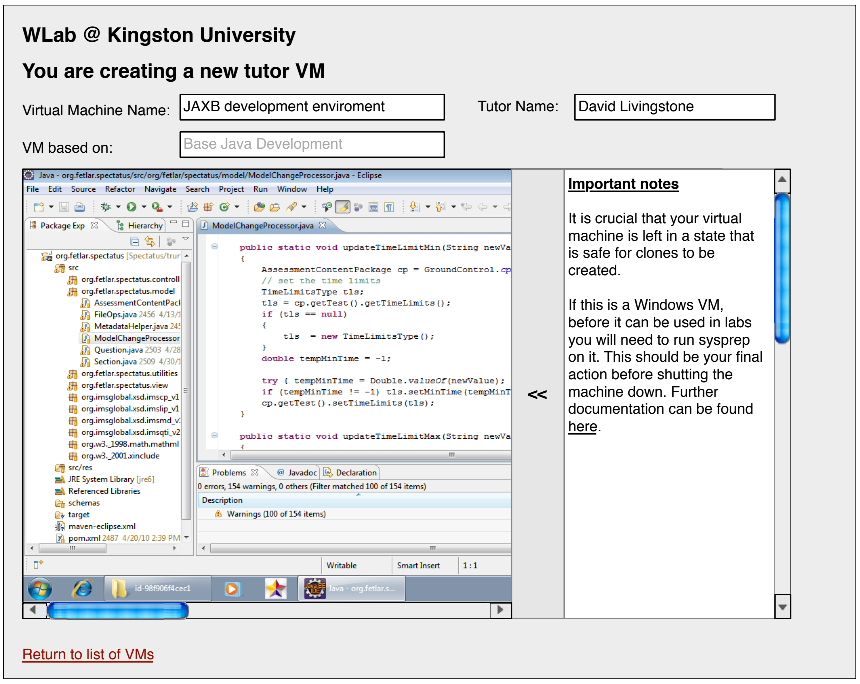
Choose from the list of Virtual Machines below to use it as a base image for your VM.

Existing Virtual Machines:

Virtual Machine Name
Base Java Development
Windows 2008 R2 Server
Basic XP VM
Windows 2003 Server
Basic Windows 7 Pro VM
Java workshop 1.3
ECT mashups workshop 1.1
ECT workshop 3.1 - AJAX

Figure 41 - Create/edit tutor VM screen mockup

The screen above shows the UI presented to a tutor when they invoke use case 7, **Create a tutor VM**. The tutor is prompted for a name for the VM. **In contrast to previous lists, this list of VMs will be a complete list straight from the virtualisation backend** (but will include existing tutor VMs). Clicking a VM and selecting *Create and start VM* will clone the VM and start up the new version. The tutor will then be given access to the console of the new VM:


WLab @ Kingston University

You are creating a new tutor VM

Virtual Machine Name: JAXB development enviroment

Tutor Name: David Livingstone

VM based on: Base Java Development

```
public static void updateTimeLimitMin(String newVa
{
    AssessmentContentPackage cp = GroundControl.cp
    // set the time limits
    TimeLimitsType tls;
    tls = cp.getTest().getTimeLimits();
    if (tls == null)
    {
        tls  = new TimeLimitsType();
    }
    double tempMinTime = -1;

    try { tempMinTime = Double.valueOf(newValue);
    if (tempMinTime != -1) tls.setMinTime(tempMinT
    cp.getTest().setTimeLimits(tls);
}

public static void updateTimeLimitMax(String newVa
{
```

```
0 errors, 154 warnings, 0 others (Filter matched 100 of 154 items)
Description
Warnings (100 of 154 items)
```

Java - org.fetlar.spectatus/src/org/fetlar/spectatus/model/ModelChangeProcessor.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Exp
Hierarchy
ModelChangeProcessor.java
Problems
Javadoc
Declaration
Writable
Smart Insert
1 : 1
<<

Important notes

It is crucial that your virtual machine is left in a state that is safe for clones to be created.

If this is a Windows VM, before it can be used in labs you will need to run sysprep on it. This should be your final action before shutting the machine down. Further documentation can be found here.

Return to list of VMs

Figure 42 - working in the tutor VM console

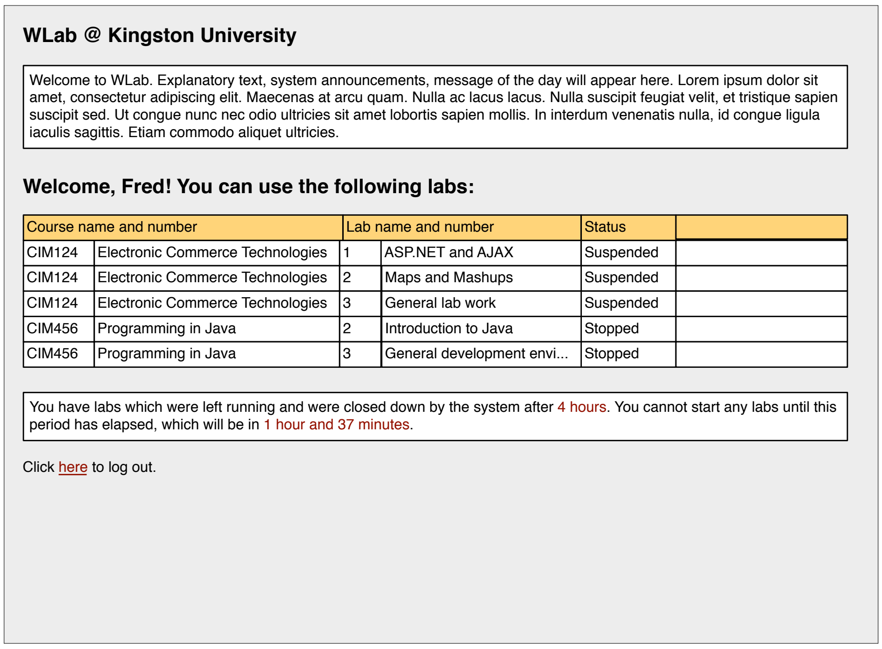
At this point, the tutor would tailor the VM as required, e.g. install software, data, etc. The tutor might be authoring with a specific stage of a lab in mind, or they may just be authoring a VM that might be used as a "template" for future VMs.

Some functionality from the student "in-lab" console is re-used here; the area previously used to display a stage's static resources can be used to display helpful information to aid the VM authoring process.

When the VM has been correctly set up, the tutor may click *Return to list of VMs*.

Modifications during the Elaboration period

In Figure 33 the two suspended labs have different periods after which they can be restarted. User feedback during Elaboration noted that this was not the requirement – the maximum runtime period is per *student*, not per lab. This screen mockup will need to be altered accordingly:



WLab @ Kingston University

Welcome to WLab. Explanatory text, system announcements, message of the day will appear here. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas at arcu quam. Nulla ac lacus lacus. Nulla suscipit feugiat velit, et tristique sapien suscipit sed. Ut congue nunc nec odio ultricies sit amet lobortis sapien mollis. In interdum venenatis nulla, id congue ligula iaculis sagittis. Etiam commodo aliquet ultricies.

Welcome, Fred! You can use the following labs:

Course name and number		Lab name and number		Status	
CIM124	Electronic Commerce Technologies	1	ASP.NET and AJAX	Suspended	
CIM124	Electronic Commerce Technologies	2	Maps and Mashups	Suspended	
CIM124	Electronic Commerce Technologies	3	General lab work	Suspended	
CIM456	Programming in Java	2	Introduction to Java	Stopped	
CIM456	Programming in Java	3	General development envi...	Stopped	

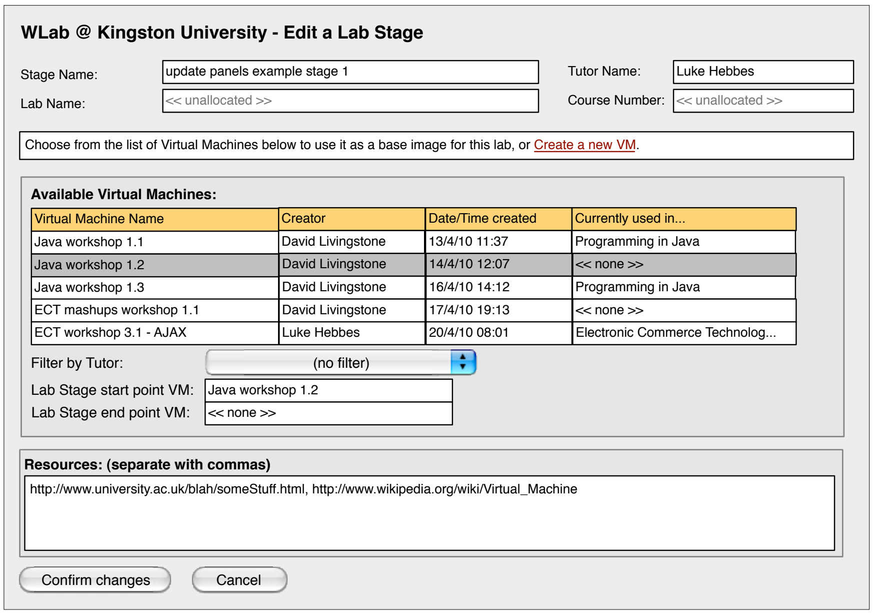
You have labs which were left running and were closed down by the system after 4 hours. You cannot start any labs until this period has elapsed, which will be in 1 hour and 37 minutes.

Click here to log out.

Figure 43 - auto-suspended labs mockup (second iteration)

User feedback also noted the importance of an "end point VM" attribute as well as a start point – see 4.2.2, Updates to Inception artefacts generated by user feedback during Elaboration.

Figure 40 changes as follows:



WLab @ Kingston University - Edit a Lab Stage

Stage Name:	update panels example stage 1	Tutor Name:	Luke Hebbes
Lab Name:	<< unallocated >>	Course Number:	<< unallocated >>

Choose from the list of Virtual Machines below to use it as a base image for this lab, or Create a new VM.

Available Virtual Machines:

Virtual Machine Name	Creator	Date/Time created	Currently used in...
Java workshop 1.1	David Livingstone	13/4/10 11:37	Programming in Java
Java workshop 1.2	David Livingstone	14/4/10 12:07	<< none >>
Java workshop 1.3	David Livingstone	16/4/10 14:12	Programming in Java
ECT mashups workshop 1.1	David Livingstone	17/4/10 19:13	<< none >>
ECT workshop 3.1 - AJAX	Luke Hebbes	20/4/10 08:01	Electronic Commerce Technolog...

Filter by Tutor:	(no filter)
Lab Stage start point VM:	Java workshop 1.2
Lab Stage end point VM:	<< none >>

Resources: (separate with commas)

http://www.university.ac.uk/blah/someStuff.html, http://www.wikipedia.org/wiki/Virtual_Machine

Figure 44 - create/edit lab stage screen mockup (second iteration)

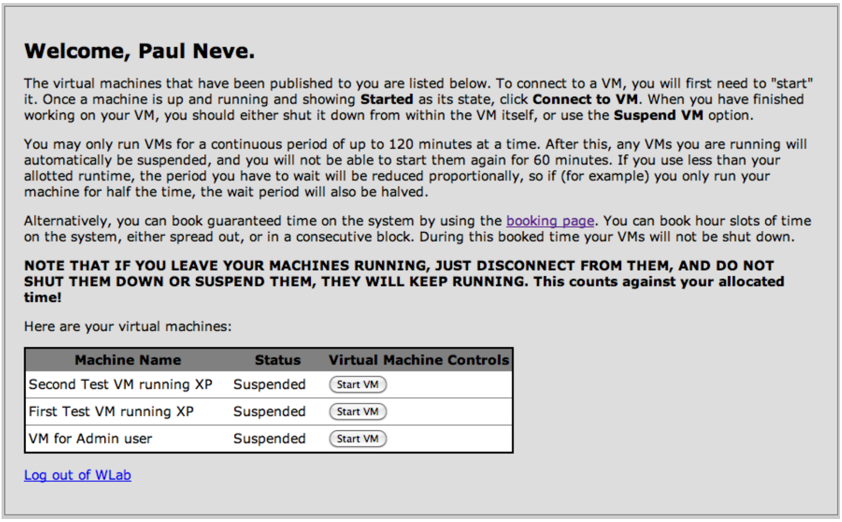
The assumption is that the tutor will highlight a target virtual machine, then click on the field next to the "start point VM" or "end point VM" in order to set these fields. If no end point VM is specified, the lab stage's end point is assumed to be the next stage's start point.

Appendix D – Development logs and analysis of work done during sprints

Sprint 1

Excellent progress was made during the first development sprint, and the prototype delivered included the features and use cases specified for both phases 1 and 2 – a week ahead of the expected schedule.

Upon logging into the system, a student sees the screen in Figure 45:



Welcome, Paul Neve.

The virtual machines that have been published to you are listed below. To connect to a VM, you will first need to "start" it. Once a machine is up and running and showing **Started** as its state, click **Connect to VM**. When you have finished working on your VM, you should either shut it down from within the VM itself, or use the **Suspend VM** option.

You may only run VMs for a continuous period of up to 120 minutes at a time. After this, any VMs you are running will automatically be suspended, and you will not be able to start them again for 60 minutes. If you use less than your allotted runtime, the period you have to wait will be reduced proportionally, so if (for example) you only run your machine for half the time, the wait period will also be halved.

Alternatively, you can book guaranteed time on the system by using the booking page. You can book hour slots of time on the system, either spread out, or in a consecutive block. During this booked time your VMs will not be shut down.

NOTE THAT IF YOU LEAVE YOUR MACHINES RUNNING, JUST DISCONNECT FROM THEM, AND DO NOT SHUT THEM DOWN OR SUSPEND THEM, THEY WILL KEEP RUNNING. This counts against your allocated time!

Here are your virtual machines:

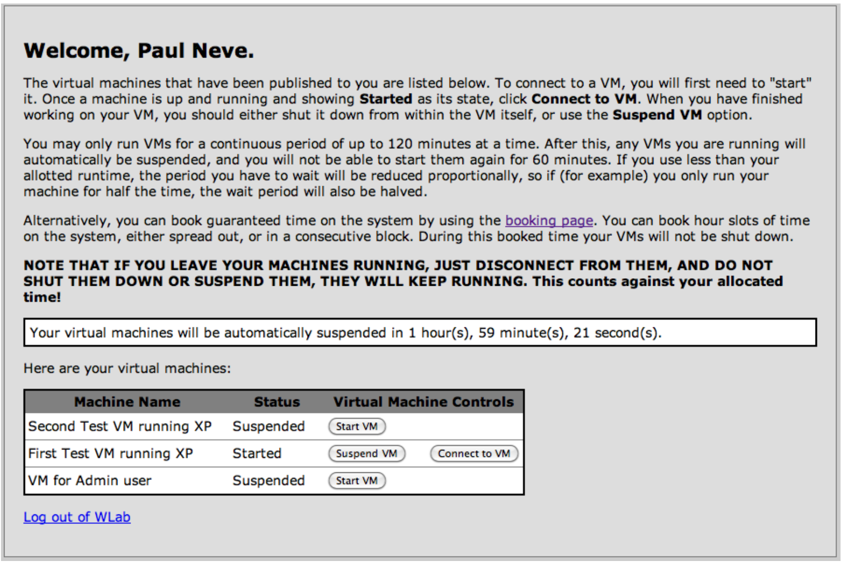
Machine Name	Status	Virtual Machine Controls
Second Test VM running XP	Suspended	Start VM
First Test VM running XP	Suspended	Start VM
VM for Admin user	Suspended	Start VM

Log out of WLab

Figure 45 – initial post-login screen for phase 1

There are a number of small differences between this and the proposed screen mockup for this point, as seen back in Figure 31. The original mockup indicated that the student would be presented with their list of *labs*, not a list of virtual machines. However, at this stage there is no difference between them and given the lab paradigm which will be introduced later, i.e. a “lab” consisting of a number of virtual machine states plus associated static resources, it was felt best to use the “virtual machine” terminology to avoid later confusion, and to approximate the current VLab system as closely as possible.

Starting a virtual machine results in the screen changing as seen in Figure 46:



Welcome, Paul Neve.

The virtual machines that have been published to you are listed below. To connect to a VM, you will first need to "start" it. Once a machine is up and running and showing **Started** as its state, click **Connect to VM**. When you have finished working on your VM, you should either shut it down from within the VM itself, or use the **Suspend VM** option.

You may only run VMs for a continuous period of up to 120 minutes at a time. After this, any VMs you are running will automatically be suspended, and you will not be able to start them again for 60 minutes. If you use less than your allotted runtime, the period you have to wait will be reduced proportionally, so if (for example) you only run your machine for half the time, the wait period will also be halved.

Alternatively, you can book guaranteed time on the system by using the booking page. You can book hour slots of time on the system, either spread out, or in a consecutive block. During this booked time your VMs will not be shut down.

NOTE THAT IF YOU LEAVE YOUR MACHINES RUNNING, JUST DISCONNECT FROM THEM, AND DO NOT SHUT THEM DOWN OR SUSPEND THEM, THEY WILL KEEP RUNNING. This counts against your allocated time!

Your virtual machines will be automatically suspended in 1 hour(s), 59 minute(s), 21 second(s).

Here are your virtual machines:

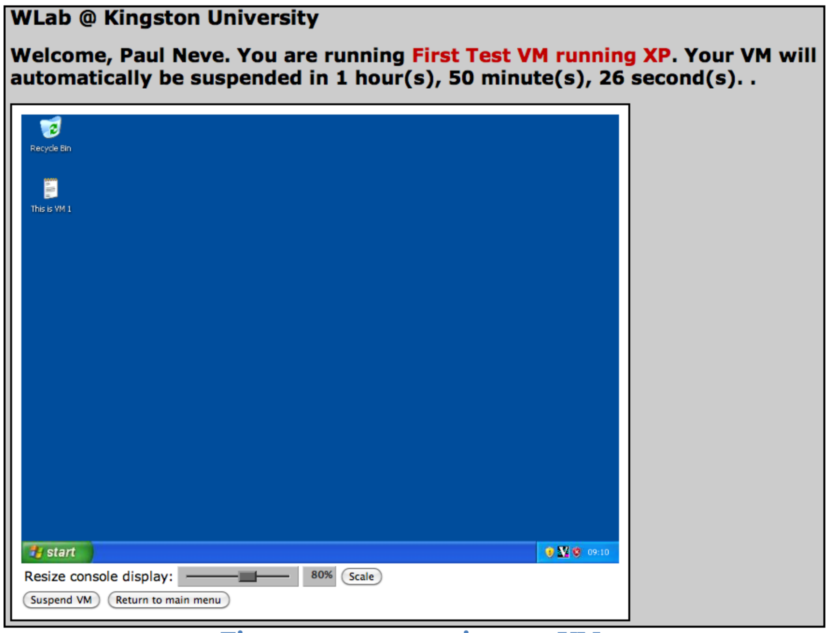
Machine Name	Status	Virtual Machine Controls
Second Test VM running XP	Suspended	Start VM
First Test VM running XP	Started	Suspend VM Connect to VM
VM for Admin user	Suspended	Start VM

Log out of WLab

Figure 46 – post-login screen with VM started (phase 1)

As outlined in the elaboration stage, each student has a period of allocated runtime, after which their running virtual machines will be suspended, and they will be prevented from restarting them for a period specified by the administrator.

Connecting to a virtual machine results in Figure 47:



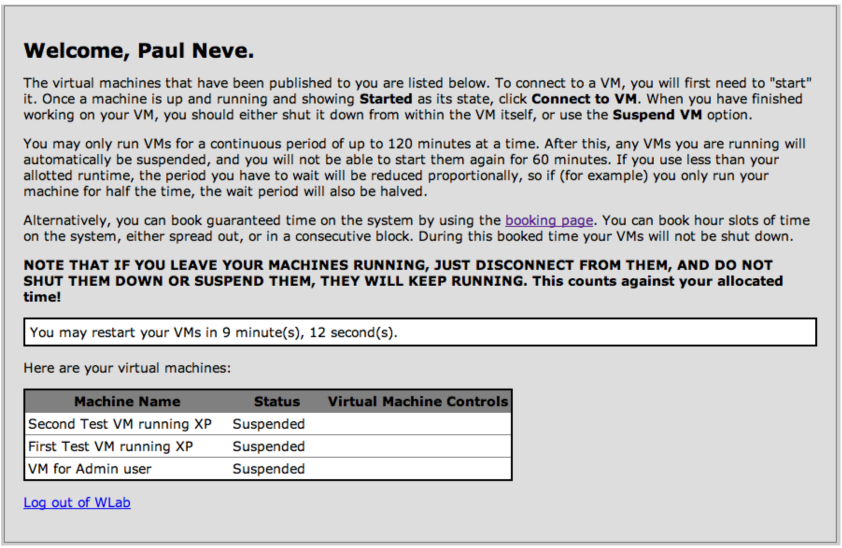
WLab @ Kingston University

Welcome, Paul Neve. You are running First Test VM running XP. Your VM will automatically be suspended in 1 hour(s), 50 minute(s), 26 second(s). .

Figure 47 – connecting to a VM

This is almost identical to that indicated by the screen mockup in Figure 34. One addition is the ability to *Resize console display*, which allows a student to reduce the size of the virtual machine console (at the price of reducing detail). This was an addition made to reduce the problems introduced by embedding the virtual machine console within a web page at a fixed size. In contrast to the current VLab system – where the destination VM is displayed to occupy the entire client screen – this means that on smaller client screens the student will have to scroll the browser page in order to see the entire VM desktop. Even on larger screens, some screen real estate is used up by the time remaining countdown, and in later phases where the lab paradigm is implemented this screen will get even busier. Therefore, the ability to scale the VM console display was seen as an important and highly needed additional feature.

When the student suspends all their VMs, their "main" screen will look like Figure 48. This is analogous to Figure 33 and its modified cousin, Figure 43:



Welcome, Paul Neve.

The virtual machines that have been published to you are listed below. To connect to a VM, you will first need to "start" it. Once a machine is up and running and showing **Started** as its state, click **Connect to VM**. When you have finished working on your VM, you should either shut it down from within the VM itself, or use the **Suspend VM** option.

You may only run VMs for a continuous period of up to 120 minutes at a time. After this, any VMs you are running will automatically be suspended, and you will not be able to start them again for 60 minutes. If you use less than your allotted runtime, the period you have to wait will be reduced proportionally, so if (for example) you only run your machine for half the time, the wait period will also be halved.

Alternatively, you can book guaranteed time on the system by using the booking page. You can book hour slots of time on the system, either spread out, or in a consecutive block. During this booked time your VMs will not be shut down.

NOTE THAT IF YOU LEAVE YOUR MACHINES RUNNING, JUST DISCONNECT FROM THEM, AND DO NOT SHUT THEM DOWN OR SUSPEND THEM, THEY WILL KEEP RUNNING. This counts against your allocated time!

You may restart your VMs in 9 minute(s), 12 second(s).

Here are your virtual machines:

Machine Name	Status	Virtual Machine Controls
Second Test VM running XP	Suspended	
First Test VM running XP	Suspended	
VM for Admin user	Suspended	

Log out of WLab

Figure 48 – waiting for the interregnum period to elapse

Here, the options to start the virtual machines is suppressed, and instead a countdown given indicating the period after which VMs may be started. Such an interregnum period occurs if the student leaves their VMs running too long and they are automatically suspended by the system; the exact duration can be specified by the system administrator and is system wide.

However, this behaviour was expanded after a discussion with the user community. The discussion examined the scenario whereby a student might avoid the automatic suspension by running their VMs until just before the automatic shut off point, shutting them down briefly, then restarting them – thus clearing their "clock". In the early stages of development, the first approach was to impose the same interregnum period on all students regardless of their run time. Thus, if a student had a two hour runtime period, the moment they started a VM, this clock would start running. Once it elapsed, any running VMs would be suspended and they would be subject to the interregnum *regardless of whether their VMs ran for the entire runtime period*. However, this was perceived to both draconian and potentially confusing for students. A student who briefly logged in for a minute or two would be subjected to the interregnum period potentially several hours later – such a student would not necessarily immediately comprehend why this was the case.

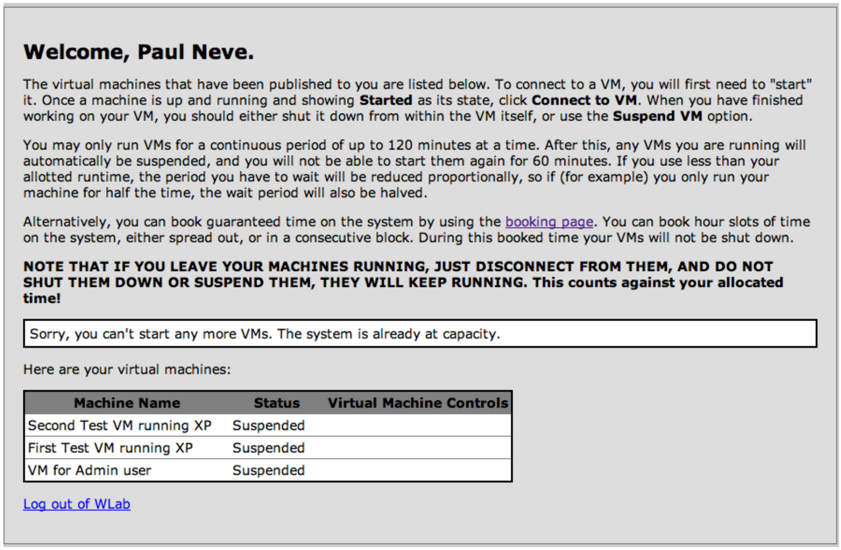
The user community suggested using a pro-rata approach. If a student uses up their entire runtime period, then they are subjected to the *entire* interregnum period. Otherwise, immediately after shutting all their VMs down they are subjected to an adjusted interregnum period proportional to the amount of runtime used. Table 5 illustrates this:

Student's maximum runtime	Maximum interregnum period	Student's actual runtime	Actual interregnum period
2 hours	30 minutes	2 hours	30 minutes
2 hours	30 minutes	1 hour	15 minutes
2 hours	30 minutes	15 minutes	3 minutes, 45 seconds

Table 5 – interregnum period examples

Thus, a student who "pops on" briefly will still be subjected to an interregnum period, but it will occur immediately after they conclude their activities (as opposed to what may seem like a "random" point in the future), and it will be of a very short duration. Students who run right up to the end of their runtime and who log off just before its end will be subject to almost the full interregnum period.

Finally, the screen in Figure 49 shows the system in a state where the number of currently running VMs is equal to or greater than the maximum defined amount. This is analogous to the screen mockup in Figure 32:



Welcome, Paul Neve.

The virtual machines that have been published to you are listed below. To connect to a VM, you will first need to "start" it. Once a machine is up and running and showing **Started** as its state, click **Connect to VM**. When you have finished working on your VM, you should either shut it down from within the VM itself, or use the **Suspend VM** option.

You may only run VMs for a continuous period of up to 120 minutes at a time. After this, any VMs you are running will automatically be suspended, and you will not be able to start them again for 60 minutes. If you use less than your allotted runtime, the period you have to wait will be reduced proportionally, so if (for example) you only run your machine for half the time, the wait period will also be halved.

Alternatively, you can book guaranteed time on the system by using the booking page. You can book hour slots of time on the system, either spread out, or in a consecutive block. During this booked time your VMs will not be shut down.

NOTE THAT IF YOU LEAVE YOUR MACHINES RUNNING, JUST DISCONNECT FROM THEM, AND DO NOT SHUT THEM DOWN OR SUSPEND THEM, THEY WILL KEEP RUNNING. This counts against your allocated time!

Sorry, you can't start any more VMs. The system is already at capacity.

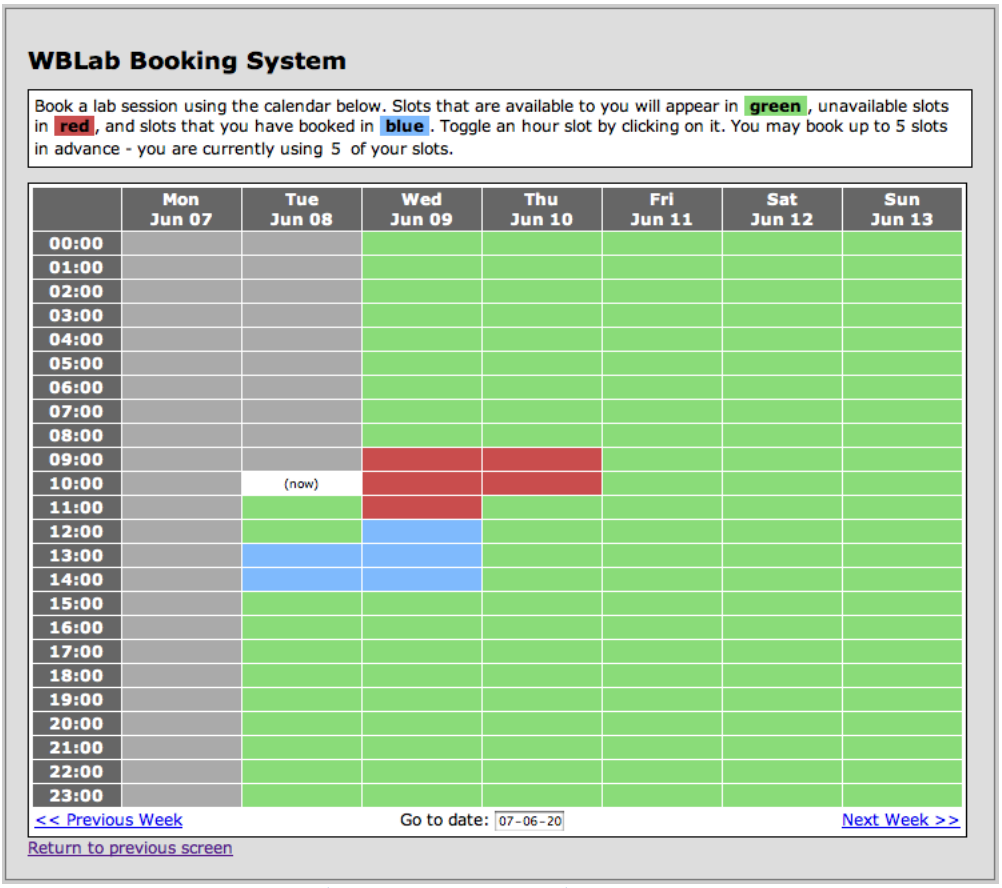
Here are your virtual machines:

Machine Name	Status	Virtual Machine Controls
Second Test VM running XP	Suspended	
First Test VM running XP	Suspended	
VM for Admin user	Suspended	

Log out of WLab

Figure 49 – system at capacity

The booking functions from phase 2 have also been implemented. The student accesses these by clicking on the link to the booking page:



WBLab Booking System

Book a lab session using the calendar below. Slots that are available to you will appear in **green**, unavailable slots in **red**, and slots that you have booked in **blue**. Toggle an hour slot by clicking on it. You may book up to 5 slots in advance - you are currently using 5 of your slots.

	Mon Jun 07	Tue Jun 08	Wed Jun 09	Thu Jun 10	Fri Jun 11	Sat Jun 12	Sun Jun 13
00:00	▨	▨	🟩	🟩	🟩	🟩	🟩
01:00	▨	▨	🟩	🟩	🟩	🟩	🟩
02:00	▨	▨	🟩	🟩	🟩	🟩	🟩
03:00	▨	▨	🟩	🟩	🟩	🟩	🟩
04:00	▨	▨	🟩	🟩	🟩	🟩	🟩
05:00	▨	▨	🟩	🟩	🟩	🟩	🟩
06:00	▨	▨	🟩	🟩	🟩	🟩	🟩
07:00	▨	▨	🟩	🟩	🟩	🟩	🟩
08:00	▨	▨	🟩	🟩	🟩	🟩	🟩
09:00	▨	▨	🟥	🟥	🟩	🟩	🟩
10:00	▨	(now)	🟥	🟥	🟩	🟩	🟩
11:00	▨	🟩	🟥	🟩	🟩	🟩	🟩
12:00	▨	🟩	🟦	🟩	🟩	🟩	🟩
13:00	▨	🟦	🟦	🟩	🟩	🟩	🟩
14:00	▨	🟦	🟦	🟩	🟩	🟩	🟩
15:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
16:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
17:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
18:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
19:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
20:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
21:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
22:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
23:00	▨	🟩	🟩	🟩	🟩	🟩	🟩

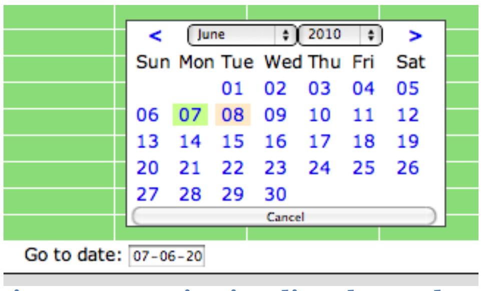
<< Previous Week Go to date: 07-06-20 Next Week >>

Return to previous screen

Figure 50 – the booking screen

Colour coding aside, this is almost identical to the screen mockup presented in Figure 35. Bookings are made immediately upon clicking an hour slot on the calendar; AJAX is used to avoid the need for intrusive screen refreshes.

A student can move between weeks using the *Next* and *Previous* links, and they can also navigate to a specific date by clicking on the field next to the *Go to date* text:



< June 2010 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		01	02	03	04	05
06	07	08	09	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Cancel

Go to date: 07-06-20

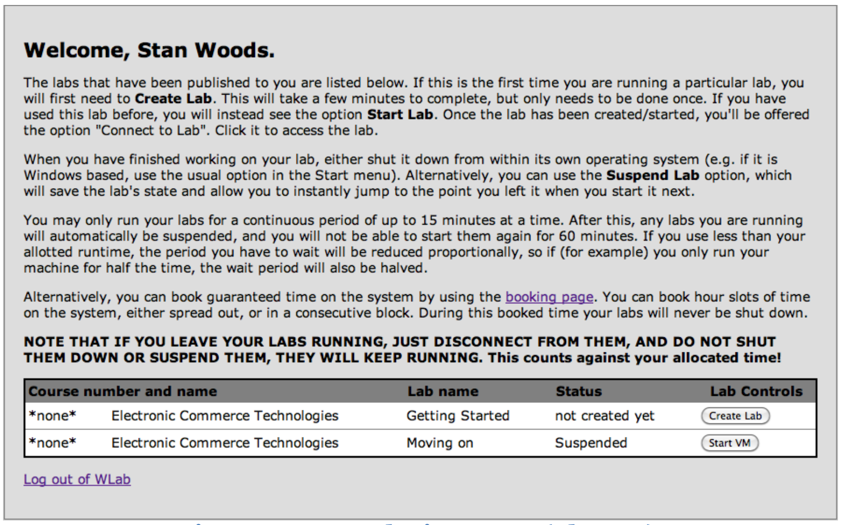
Figure 51 – navigating directly to a date

The introduction of the booking system mandated a number of small changes; system capacity at a given moment is now measured not only by how many VMs are running, but also by how many bookings there are. Additionally, one consideration that had not been anticipated but that emerged during development (and has been catered for) is that of "weighting". A booking made by student who has the ability to run 5 concurrent VMs has the potential to take up more resources than that made by a student who can only run 1. Thus system capacity measurements now additionally take this factor into account.

Sprint 2

The development continued to make good progress. Sprint 2 successfully extended the previous work to incorporate the features outlined for phase 3 and maintain the project at a week ahead of schedule.

Upon logging into the system, the student sees a screen like Figure 52. This is similar to the previous iteration, but there are some important differences:



Welcome, Stan Woods.

The labs that have been published to you are listed below. If this is the first time you are running a particular lab, you will first need to **Create Lab**. This will take a few minutes to complete, but only needs to be done once. If you have used this lab before, you will instead see the option **Start Lab**. Once the lab has been created/started, you'll be offered the option "Connect to Lab". Click it to access the lab.

When you have finished working on your lab, either shut it down from within its own operating system (e.g. if it is Windows based, use the usual option in the Start menu). Alternatively, you can use the **Suspend Lab** option, which will save the lab's state and allow you to instantly jump to the point you left it when you start it next.

You may only run your labs for a continuous period of up to 15 minutes at a time. After this, any labs you are running will automatically be suspended, and you will not be able to start them again for 60 minutes. If you use less than your allotted runtime, the period you have to wait will be reduced proportionally, so if (for example) you only run your machine for half the time, the wait period will also be halved.

Alternatively, you can book guaranteed time on the system by using the booking page. You can book hour slots of time on the system, either spread out, or in a consecutive block. During this booked time your labs will never be shut down.

NOTE THAT IF YOU LEAVE YOUR LABS RUNNING, JUST DISCONNECT FROM THEM, AND DO NOT SHUT THEM DOWN OR SUSPEND THEM, THEY WILL KEEP RUNNING. This counts against your allocated time!

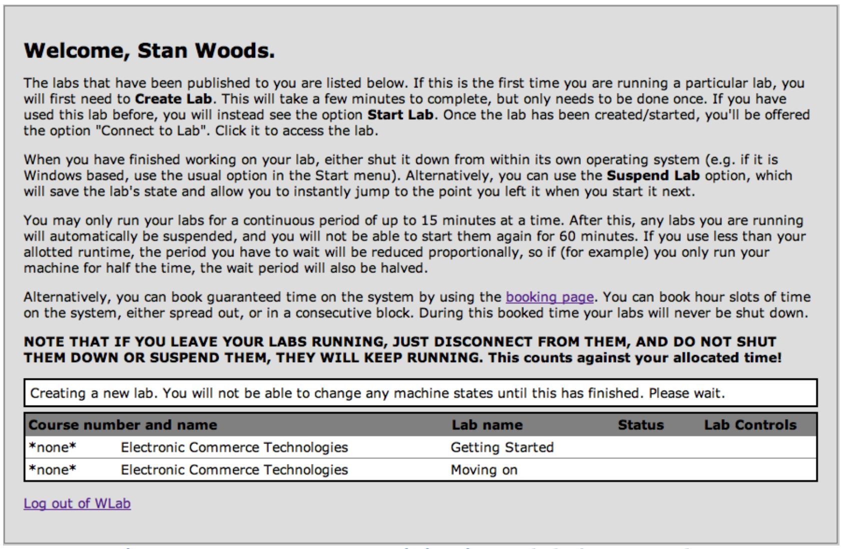
Course number and name		Lab name	Status	Lab Controls
none	Electronic Commerce Technologies	Getting Started	not created yet	Create Lab
none	Electronic Commerce Technologies	Moving on	Suspended	Start VM

Log out of WLab

Figure 52 – post-login screen (phase 3)

All references to "virtual machines" have been removed, replaced with the term "lab". These labs still consist of a single virtual machine, but in contrast to the previous iteration where VMs had to be set up in advance for each student, here the VMs are created on demand.

The screenshot in Figure 52 shows a student who has already used one lab, the *Moving On* lab. However, the *Getting Started* lab shows a status of "not created yet". Clicking the *Create Lab* button will result in Figure 53:



Welcome, Stan Woods.

The labs that have been published to you are listed below. If this is the first time you are running a particular lab, you will first need to **Create Lab**. This will take a few minutes to complete, but only needs to be done once. If you have used this lab before, you will instead see the option **Start Lab**. Once the lab has been created/started, you'll be offered the option "Connect to Lab". Click it to access the lab.

When you have finished working on your lab, either shut it down from within its own operating system (e.g. if it is Windows based, use the usual option in the Start menu). Alternatively, you can use the **Suspend Lab** option, which will save the lab's state and allow you to instantly jump to the point you left it when you start it next.

You may only run your labs for a continuous period of up to 15 minutes at a time. After this, any labs you are running will automatically be suspended, and you will not be able to start them again for 60 minutes. If you use less than your allotted runtime, the period you have to wait will be reduced proportionally, so if (for example) you only run your machine for half the time, the wait period will also be halved.

Alternatively, you can book guaranteed time on the system by using the booking page. You can book hour slots of time on the system, either spread out, or in a consecutive block. During this booked time your labs will never be shut down.

NOTE THAT IF YOU LEAVE YOUR LABS RUNNING, JUST DISCONNECT FROM THEM, AND DO NOT SHUT THEM DOWN OR SUSPEND THEM, THEY WILL KEEP RUNNING. This counts against your allocated time!

Creating a new lab. You will not be able to change any machine states until this has finished. Please wait.

Course number and name		Lab name	Status	Lab Controls
none	Electronic Commerce Technologies	Getting Started		
none	Electronic Commerce Technologies	Moving on		

Log out of WLab

Figure 53 – system provisioning a lab for a student

Once the lab has been created, it is automatically started, and the student is presented with the usual options to connect to it, or suspend it. All other functions, such as the console screen where the student interacts with a virtual machine, booking functions and connection management and timeouts, are as before.

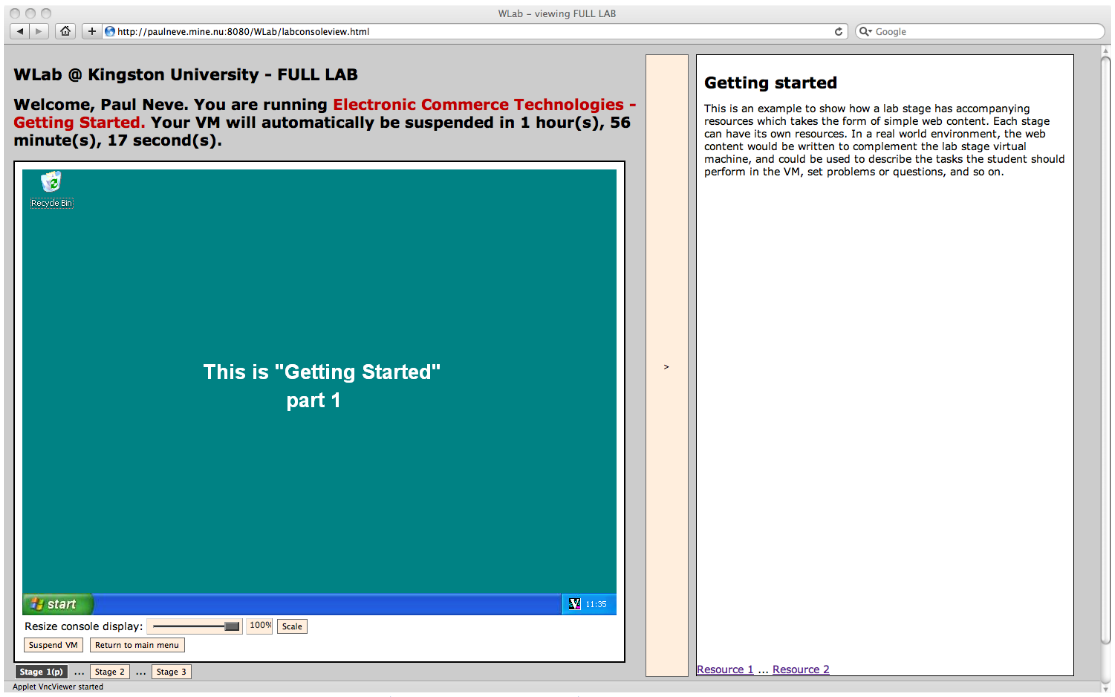
The need to make Windows VMs unique before they can be cloned has been discussed previously, and to facilitate the use of tools like *wsname* a tutor VM has a configuration setting, *rebootAfterCloning*. Because a change to a Windows computer name requires a reboot, if one sets the computer name of a freshly created clone to a unique value on first boot, it will require a reboot to take effect – meaning the "first boot" is actually two boot cycles. The *rebootAfterCloning* value allows the VM author to specify how many boot cycles to expect after cloning before the clone is available for the student. In most cases, assuming that the new VM contains a Windows OS and requires Windows network functionality, this value would be set to 2. In this scenario, the system will boot the clone and then wait for it to shut down to a fully powered off state. It will then start the clone a second time and only then, once this second boot has finished, allow student connections. If an alternative method other than *wsname* is being used that requires more than two boot cycles, you would increase this value as appropriate. A value of 1 here would make the new cloned VM immediately available to the student, with no additional shut down and boot cycles. This might be an appropriate setting for non-Windows OSes, or for environments where students' lab exercises required no network functionality and the VMs could be isolated from each other.

In any case, the assumption by the system is that any tutor VM that is cloned by the system and allocated to a student for use was left in a state whereby, on next reboot, it will either run a script to make itself unique, or it does not require such considerations in the first place.

Sprint 3

Development continued at the same pace maintaining the week ahead of schedule. During sprint 3, the feature set proposed for phase 4 was delivered. In the previous sprint, a "lab" was defined as a dynamically created virtual machine that is built per-student on demand at the point of their first request for it. This was now extended as per the phase 4 metaphor, and the complete lab / lab stage / VM and resource(s) composite learning object was implemented for the first time (see **Error! Reference source not found.**).

The login and lab listing screens are unchanged from previous sprints. However, when connecting to a lab, the student now sees Figure 54:


WLab – viewing FULL LAB

http://paulneve.mine.nu:8080/WLab/labconsoleview.html

WLab @ Kingston University - FULL LAB

Welcome, Paul Neve. You are running Electronic Commerce Technologies - Getting Started. Your VM will automatically be suspended in 1 hour(s), 56 minute(s), 17 second(s).

This is "Getting Started"
part 1

Resize console display: 100% Scale

Suspend VM | Return to main menu

Stage 1(p) ... Stage 2 ... Stage 3

Applet VncViewer started

Getting started

This is an example to show how a lab stage has accompanying resources which takes the form of simple web content. Each stage can have its own resources. In a real world environment, the web content would be written to complement the lab stage virtual machine, and could be used to describe the tasks the student should perform in the VM, set problems or questions, and so on.

Resource 1 ... Resource 2

Figure 54 – the "in-lab" console

This is analogous to Figure 36. The VM console of the current lab stage is displayed on the left, and the resources for the lab stage on the right hand side of the screen. If the lab stage has no resources, the right hand pane is suppressed to give more screen space to the VM console. The student can also click the button between the two panes to collapse the resource pane, and then again to make it reappear.

Underneath the VM console screen, buttons that represent the individual stages of the lab exercise are displayed. The student starts on lab stage 1. Clicking on one of the buttons will move the student to that particular lab stage – this means:

- If the student has never visited that particular lab stage before, a new VM will be dynamically created for them.
- The running VM for the currently running lab stage will be suspended.
- The VM for the destination lab stage will be started.
- The screen reloads, and the new lab stage VM console is displayed, along with its resources (if any).

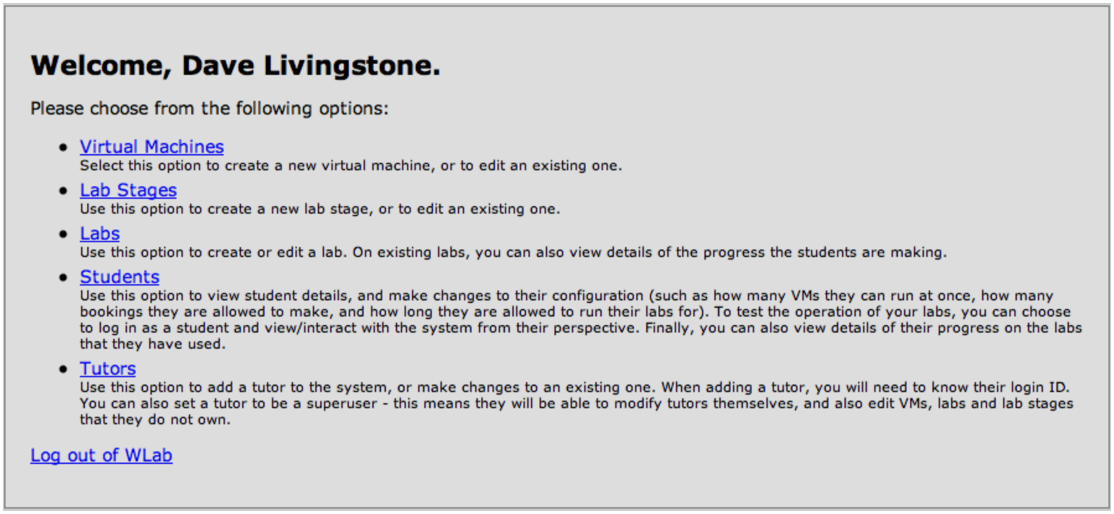
Similarly, the links at the bottom of the resource pane will move the student from resource to resource, if there is more than one for this lab stage.

Sprint 4

Sprint 4 was primarily concerned with implementing the functionality earmarked for phase 5, which consisted for the most part of the management interface for tutors to create labs, lab stages etc.

Main tutor menu

When a tutor logs in, they are confronted with a main menu. This is not directly analogous to any particular screen mockup, as the initial presentation to tutor was not specified during the Inception and Elaboration phases of the project.



Welcome, Dave Livingstone.

Please choose from the following options:

- Virtual Machines
  Select this option to create a new virtual machine, or to edit an existing one.
- Lab Stages
  Use this option to create a new lab stage, or to edit an existing one.
- Labs
  Use this option to create or edit a lab. On existing labs, you can also view details of the progress the students are making.
- Students
  Use this option to view student details, and make changes to their configuration (such as how many VMs they can run at once, how many bookings they are allowed to make, and how long they are allowed to run their labs for). To test the operation of your labs, you can choose to log in as a student and view/interact with the system from their perspective. Finally, you can also view details of their progress on the labs that they have used.
- Tutors
  Use this option to add a tutor to the system, or make changes to an existing one. When adding a tutor, you will need to know their login ID. You can also set a tutor to be a superuser - this means they will be able to modify tutors themselves, and also edit VMs, labs and lab stages that they do not own.

Log out of WLab

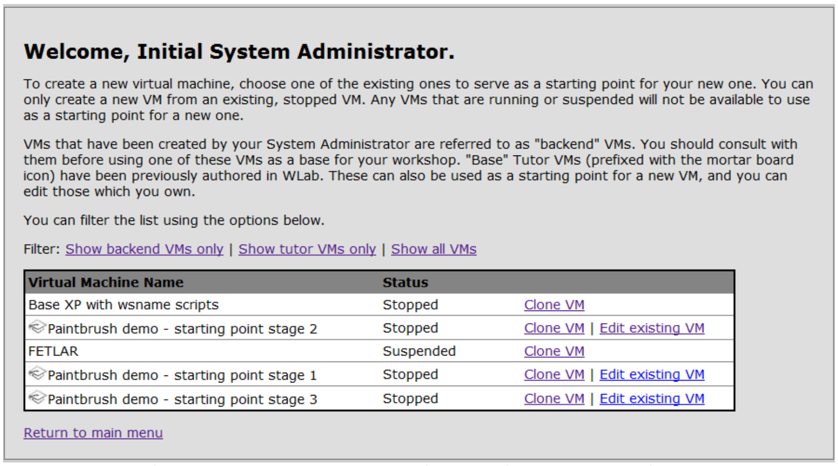
Figure 55 – the main tutor menu

The main menu presents various areas that a tutor must traverse to create a lab in a logical order if read from top to bottom. To set up a lab exercise, a tutor must create at least one tutor VM, followed, by at least one lab stage (to which the VM will be attached), before creating the lab itself and adding the lab stage(s) to it. The tutor might subsequently go back and add additional VMs and stages, but the process of creating a lab will inevitably require the creation of those objects in that order.

The *Students* and *Tutors* menu options are not directly implied by any use cases or screen mockups, but one can anticipate that there is be a need to make changes to student configuration settings (e.g. their maximum lab run time) without resorting to editing XML. Similarly, adding new tutors to the system should also be XML-free. Given that development at this point is approximately a week ahead of schedule, it is not unreasonable to add these menu options with the intention of populating them with the appropriate functionality in the remaining sprints.

Virtual Machines

Selecting the *Virtual Machines* link will display a screen similar to Figure 56:



Welcome, Initial System Administrator.

To create a new virtual machine, choose one of the existing ones to serve as a starting point for your new one. You can only create a new VM from an existing, stopped VM. Any VMs that are running or suspended will not be available to use as a starting point for a new one.

VMs that have been created by your System Administrator are referred to as "backend" VMs. You should consult with them before using one of these VMs as a base for your workshop. "Base" Tutor VMs (prefixed with the mortar board icon) have been previously authored in WLab. These can also be used as a starting point for a new VM, and you can edit those which you own.

You can filter the list using the options below.

Filter: Show backend VMs only | Show tutor VMs only | Show all VMs

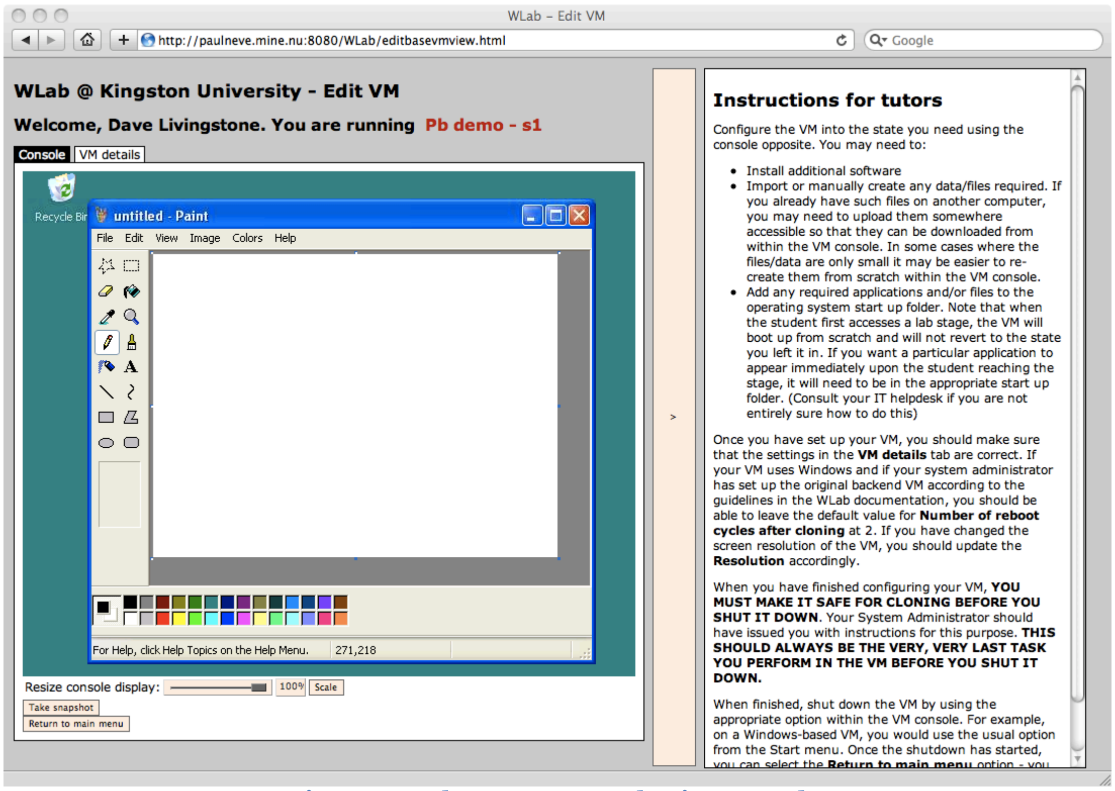
Virtual Machine Name	Status	
Base XP with wsname scripts	Stopped	Clone VM
Paintbrush demo - starting point stage 2	Stopped	Clone VM | Edit existing VM
FETLAR	Suspended	Clone VM
Paintbrush demo - starting point stage 1	Stopped	Clone VM | Edit existing VM
Paintbrush demo - starting point stage 3	Stopped	Clone VM | Edit existing VM

Return to main menu

Figure 56 – the tutor's list of virtual machines

This screen is analogous to the mockup shown in Figure 41, but has been extended considerably. While the list does come straight from the virtualisation backend, tutor VMs and distinguished by a mortarboard icon and can be edited by their owning tutor or by a superuser. In contrast, a backend VM may not be edited. The assumption is that the system administrator will create a small number of backend VMs that will provide a foundation for tutors to use as a base their labs around, and then add to in order to design their courses; it makes sense for such foundation VMs to be constants and not easily editable at end user level.

Selecting *Clone VM* prompts the tutor for a name for the clone, creates it, and then connects them to it. Similarly, *Edit existing VM* starts a VM and then connects the tutor to it. Either way, the tutor will be presented with the VM console:



WLab @ Kingston University - Edit VM

Welcome, Dave Livingstone. You are running Pb demo - s1

Instructions for tutors

Configure the VM into the state you need using the console opposite. You may need to:

- Install additional software
- Import or manually create any data/files required. If you already have such files on another computer, you may need to upload them somewhere accessible so that they can be downloaded from within the VM console. In some cases where the files/data are only small it may be easier to re-create them from scratch within the VM console.
- Add any required applications and/or files to the operating system start up folder. Note that when the student first accesses a lab stage, the VM will boot up from scratch and will not revert to the state you left it in. If you want a particular application to appear immediately upon the student reaching the stage, it will need to be in the appropriate start up folder. (Consult your IT helpdesk if you are not entirely sure how to do this)

Once you have set up your VM, you should make sure that the settings in the **VM details** tab are correct. If your VM uses Windows and if your system administrator has set up the original backend VM according to the guidelines in the WLab documentation, you should be able to leave the default value for **Number of reboot cycles after cloning** at 2. If you have changed the screen resolution of the VM, you should update the **Resolution** accordingly.

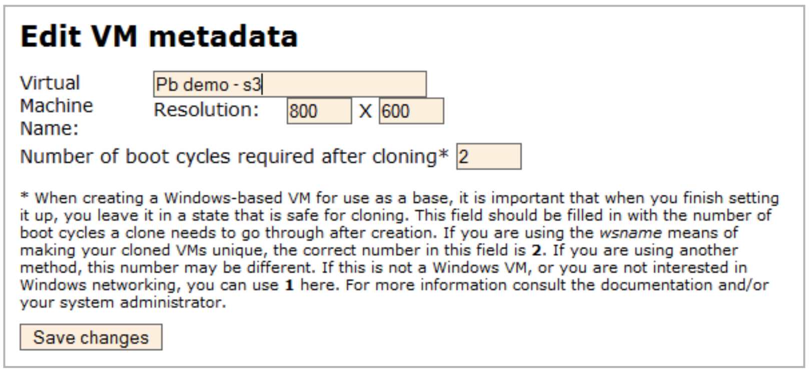
When you have finished configuring your VM, **YOU MUST MAKE IT SAFE FOR CLONING BEFORE YOU SHUT IT DOWN**. Your System Administrator should have issued you with instructions for this purpose. **THIS SHOULD ALWAYS BE THE VERY, VERY LAST TASK YOU PERFORM IN THE VM BEFORE YOU SHUT IT DOWN.**

When finished, shut down the VM by using the appropriate option within the VM console. For example, on a Windows-based VM, you would use the usual option from the Start menu. Once the shutdown has started, you can select the **Return to main menu** option - you

Figure 57 – the tutor VM authoring console

This screen is based on the screen mockup in and is a simple refactoring of the existing code that delivers a VM to students from previous sprints.

One addition beyond the screen mockups is the *Console* and *VM details* tabs. The latter allows the tutor to set a number of important configuration settings:



Edit VM metadata

Virtual Machine Name: Pb demo - s3

Resolution: 800 X 600

Number of boot cycles required after cloning* 2

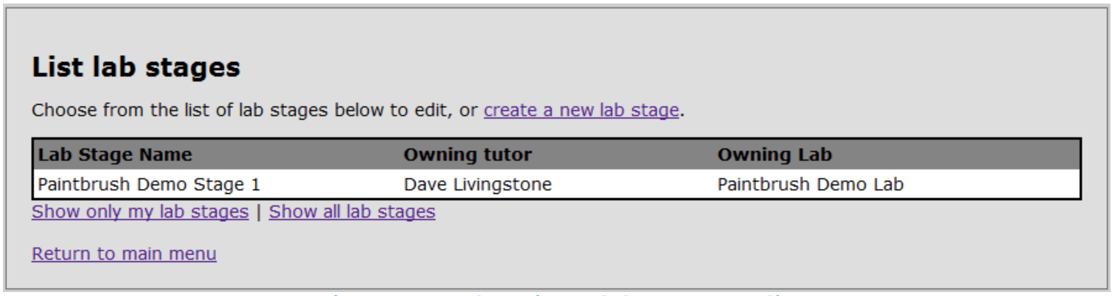
* When creating a Windows-based VM for use as a base, it is important that when you finish setting it up, you leave it in a state that is safe for cloning. This field should be filled in with the number of boot cycles a clone needs to go through after creation. If you are using the *wsname* means of making your cloned VMs unique, the correct number in this field is **2**. If you are using another method, this number may be different. If this is not a Windows VM, or you are not interested in Windows networking, you can use **1** here. For more information consult the documentation and/or your system administrator.

Figure 58 – editing tutor VM metadata

The importance of making new VMs unique has been discussed previously, along with the potential for a newly created clone to require more than one boot cycle; the field here allows the tutor to set an appropriate value to take this into account. The tutor can also specify the screen resolution of the VM – this is important because it will tell view components how much screen space to give the VNC applet within the student's browser.

Lab Stages

Selecting the *Lab Stages* function from the main menu yields Figure 59. (This screenshot shows the system with only a single lab currently extant, but multiple labs would appear as one would expect if present)



List lab stages

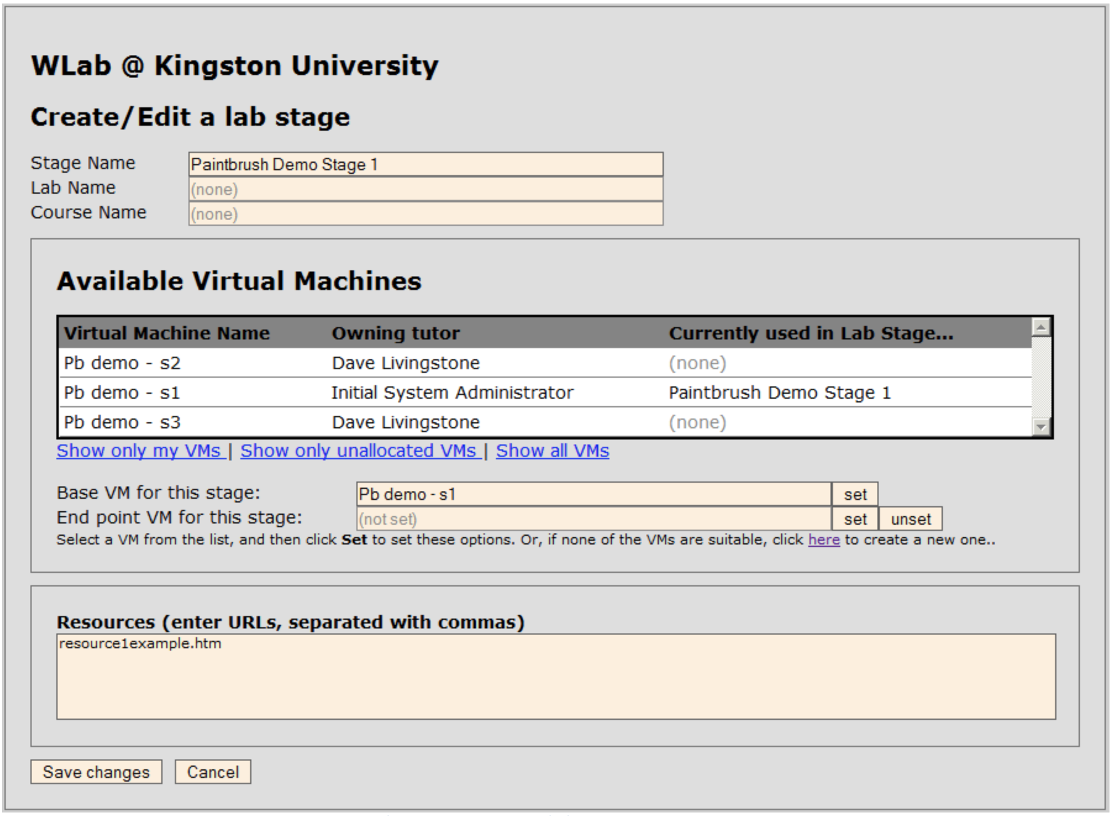
Choose from the list of lab stages below to edit, or create a new lab stage.

Lab Stage Name	Owning tutor	Owning Lab
Paintbrush Demo Stage 1	Dave Livingstone	Paintbrush Demo Lab

Figure 59 – choosing a lab stage to edit

There are options to filter the labs shown, so that on a "busy" system a tutor can more easily disregard superfluous data.

Clicking on a lab results in Figure 60, which – some small cosmetic considerations aside – is virtually identical to the design indicated by the mockup in Figure 44:



WLab @ Kingston University

Create/Edit a lab stage

Stage Name	Paintbrush Demo Stage 1
Lab Name	(none)
Course Name	(none)

Available Virtual Machines

Virtual Machine Name	Owning tutor	Currently used in Lab Stage...
Pb demo - s2	Dave Livingstone	(none)
Pb demo - s1	Initial System Administrator	Paintbrush Demo Stage 1
Pb demo - s3	Dave Livingstone	(none)

Show only my VMs | Show only unallocated VMs | Show all VMs

Base VM for this stage:	Pb demo - s1
End point VM for this stage:	(not set)

Select a VM from the list, and then click **Set** to set these options. Or, if none of the VMs are suitable, click here to create a new one..

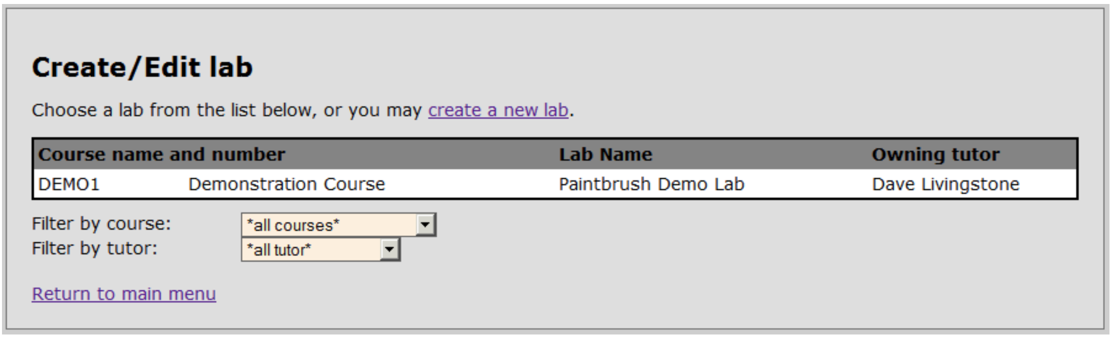
Resources (enter URLs, separated with commas)

resource1example.htm

Figure 60 – editing a lab stage

Labs

Selecting the *Labs* function from the main menu results in Figure 61, which is derived from the mockup in Figure 38. Again, only a single item is shown, but on a fully-populated system multiple items would appear as one would expect:



Create/Edit lab

Choose a lab from the list below, or you may create a new lab.

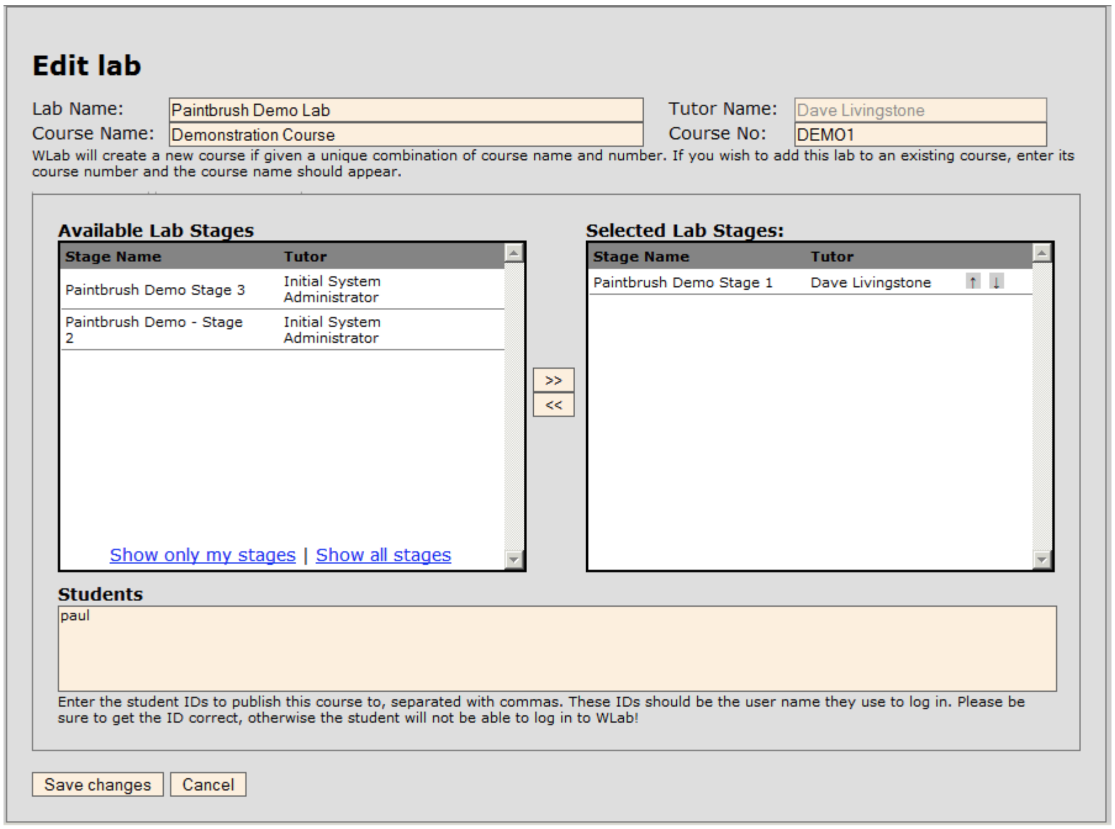
Course name and number		Lab Name	Owning tutor
DEMO1	Demonstration Course	Paintbrush Demo Lab	Dave Livingstone

Filter by course:	*all courses*
Filter by tutor:	*all tutor*

Return to main menu

Figure 61 – selecting a lab to edit

Selecting a lab (or using the *Create a new lab* option) would result in Figure 62, which follows Figure 39 extremely closely, and functions as per the accompanying discussion there:



Edit lab

Lab Name:	Paintbrush Demo Lab	Tutor Name:	Dave Livingstone
Course Name:	Demonstration Course	Course No:	DEMO1

WLab will create a new course if given a unique combination of course name and number. If you wish to add this lab to an existing course, enter its course number and the course name should appear.

Available Lab Stages

Stage Name	Tutor
Paintbrush Demo Stage 3	Initial System Administrator
Paintbrush Demo - Stage 2	Initial System Administrator

Show only my stages | Show all stages

Selected Lab Stages:

Stage Name	Tutor
Paintbrush Demo Stage 1	Dave Livingstone

Students

paul

Enter the student IDs to publish this course to, separated with commas. These IDs should be the user name they use to log in. Please be sure to get the ID correct, otherwise the student will not be able to log in to WLab!

Figure 62 – editing a lab

One aspect that was not anticipated in the screen mockups was the need to be able to re-order lab stages, which is achieved by the up/down arrows next to a selected lab stage. The delete buttons next to each selected lab stage are also superfluous – the $\boxed{<<}$ button already performs this task.

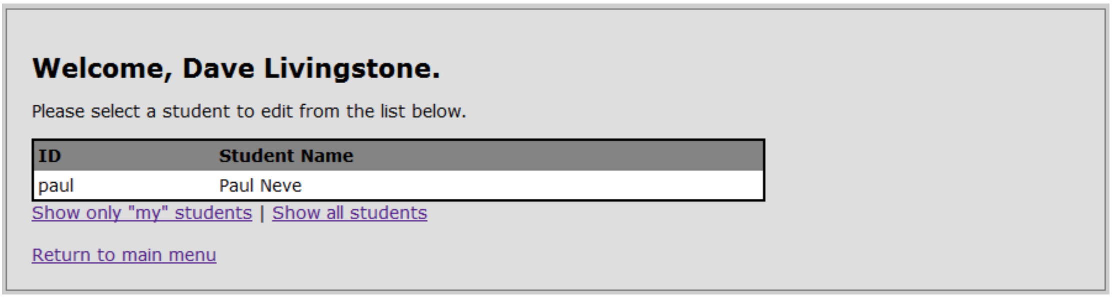
Another conscious decision has been taken to remove the option to create a new lab stage from the Edit Lab screen, to avoid inexperienced tutors forgetting which point of the authoring process they are at. The intent is for the user documentation to drive them down the path outlined previously, i.e. author VM ➔ lab stage ➔ lab. The feeling is that encouraging tutors to dive out of that simple route might create risk potential confusion.

Sprints 5 and 6

Coming to the end of the allocated development time period, the five phases anticipated have all been delivered. Consequently, the remaining development time will seek to implement functionality that was suggested once an attempt was made to use the system in an end-to-end fashion[8]. Note that these functions have not been explicitly mandated by the various project artefacts, so they do not correspond with (for example) screen mockups.

Student option in tutor's management menu

Clicking the *Student* menu now displays a screen similar to Figure 63. (As with certain other screenshots, only one item is displayed, but multiple items would appear on a fully-populated system.)



Welcome, Dave Livingstone.

Please select a student to edit from the list below.

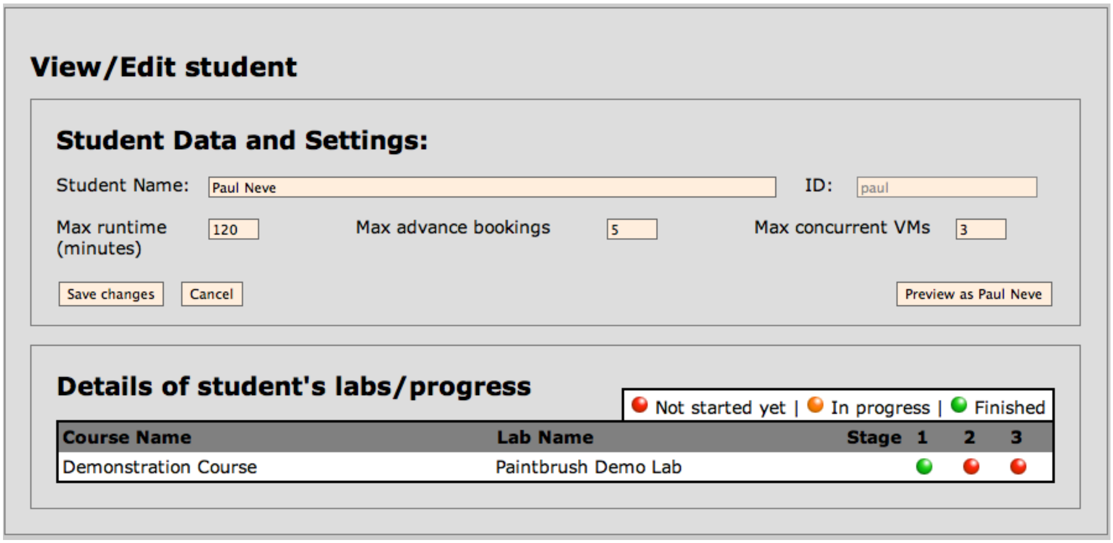
ID	Student Name
paul	Paul Neve

Show only "my" students | Show all students

Return to main menu

Figure 63 – selecting a student to edit

A tutor can filter out students who are not allocated to one of their courses, or display all students. Clicking on a student results in a screen like Figure 64:



View/Edit student

Student Data and Settings:

Student Name:	Paul Neve			ID:	paul
Max runtime (minutes)	120	Max advance bookings	5	Max concurrent VMs	3

Details of student's labs/progress

🔴 Not started yet | 🟠 In progress | 🟢 Finished

Course Name	Lab Name	Stage 1	2	3
Demonstration Course	Paintbrush Demo Lab	🟢	🔴	🔴

Figure 64 – editing a student

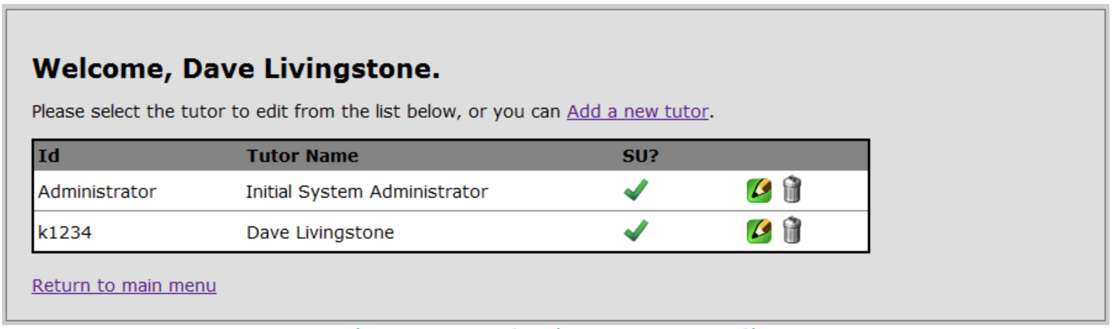
At the top of the screen, the configuration settings for the student are displayed, and can be modified. Underneath, a list of labs that have been published to the student appears, along with details of the progress that has been made by the student in each lab.

The *Preview as*... button allows the tutor to assume the identity of a student from the perspective of the WLab system. The intended use for this function is for tutors to test their labs, and ensure the student experience is as they'd expect.

[8] i.e. starting with a clean slate, logging in assuming the role of the initial tutor, attempting to author VMs, lab stages, labs, using the lab from the perspective of a "student", then re-assuming the tutor role and attempting examine the student's progress

Tutor option in tutor's management menu

Clicking on the *Tutors* menu option will result in a screen like Figure 65:



Welcome, Dave Livingstone.

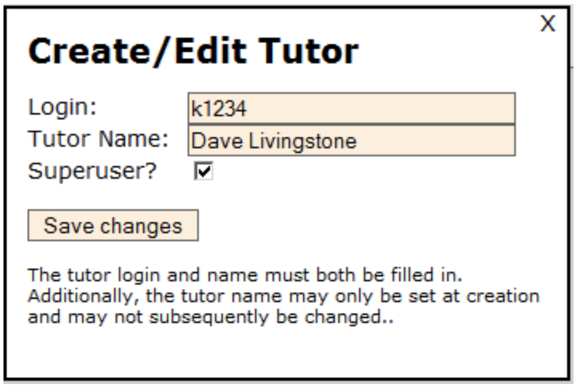
Please select the tutor to edit from the list below, or you can Add a new tutor.

Id	Tutor Name	SU?	
Administrator	Initial System Administrator	✓	
k1234	Dave Livingstone	✓	

Return to main menu

Figure 65 – selecting a tutor to edit

A tutor's details may be edited by clicking on the pencil icon, or a new tutor can be added with *Add a new tutor*:



Create/Edit Tutor

Login:	k1234
Tutor Name:	Dave Livingstone
Superuser?	☑

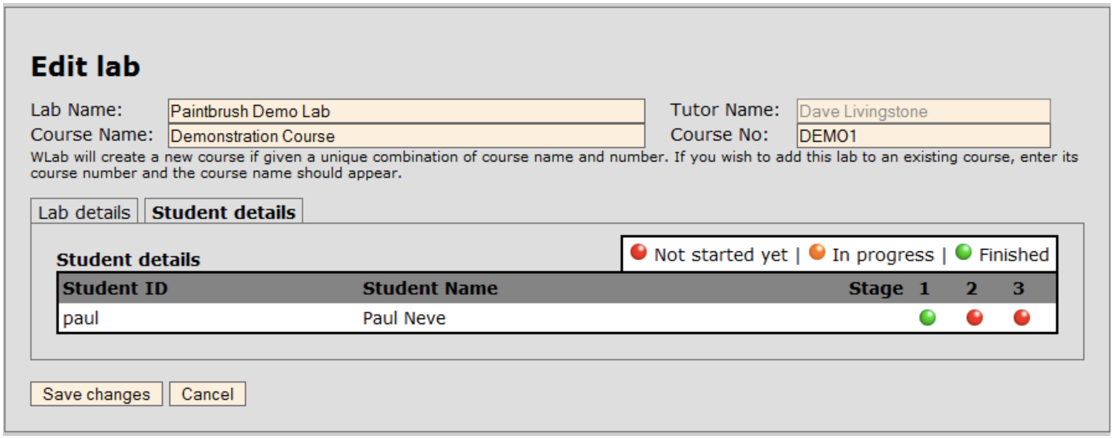
The tutor login and name must both be filled in. Additionally, the tutor name may only be set at creation and may not subsequently be changed..

Figure 66 – editing a tutor

The login ID cannot be modified once a tutor has been entered, but the tutor name and their superuser status is changeable.

Student progress details within lab screen

When editing an exiting lab, two tabs have been added. *Lab details* will display the usual screen where lab stages and the student list for the lab can be modified. However, *Student Details* will show a screen akin to that shown in Figure 67:



Edit lab

Lab Name:	Paintbrush Demo Lab	Tutor Name:	Dave Livingstone
Course Name:	Demonstration Course	Course No:	DEMO1

WLab will create a new course if given a unique combination of course name and number. If you wish to add this lab to an existing course, enter its course number and the course name should appear.

Student details

● Not started yet | ● In progress | ● Finished

Student ID	Student Name	Stage 1	2	3
paul	Paul Neve	●	●	●

Figure 67 – student progress details by lab

Here, you can find details of all the students' progress on this particular lab. While this particular lab has is only currently published to one student, this is a full list of students – a lab that is published to 20 students would see all 20 displayed there.

Other work during these sprints

An amount of "behind the scenes" work has taken place during the remaining sprints:

- **Removal of all hard-coded user viewable strings in Java code and JSP pages**
  Up until this point, all string values that appear to the user (e.g. "Tutor name", "Lab name", "WLab will create a new course if given a unique combination of course name and number") were either hard-coded within the controller classes, or inline on the JSP pages. Those wishing to customise such values (for example, for translating the application into a language other than English) would need to build a complete development environment, make their changes, and rebuild the application – a non-trivial task. Consequently, all such values have been moved into an easily editable XML file (conf/strings.xml) and Spring dependency injection used to place these values into the appropriate places in Java and/or JSP code.

- **Modifications/enhancements to TightVNC client applet**
  A degree of inter-connectivity between the applet and the JSP-generated web page it is embedded in has been implemented. When VNC fails to connect or its connection is dropped, the applet resubmits the page. This is a cleaner approach than that used previously, where the VNC applet itself attempted to re-establish the connection but the page did not reload. It means that the appropriate Java controller code is re-run, which means that functions such as IP address detection take place. In the event that the connection failure was due to an environmental issue such as the VM's IP address changing, the student will simply see a page reload before their session resumes – as opposed to a blank VNC applet attempting to connect to an IP address that is no longer valid.

- **Distinguishing between "superuser" and normal-level tutors**
  Additional code was added to restrict normal-level tutors so that they can only modify their own labs, lab stages and tutor VMs. They can still clone other tutor's VMs, if they wish. Superusers continue to be able to edit all objects regardless of owners.

- **Refactoring of code and development environment to support Apache Maven**
  Maven is a software and dependency management tool that greatly simplifies the compile and build process. As the intent is for the final application to be available under an open source licence, it is important that potential developers who wish to make use of the application can easily import it into their development environment. The use of Maven allows such developers to quickly build the application with simple commands, without having to manually acquire the various libraries (e.g. JiBX, JWBem, etc) on which the application depends.

Appendix E – End-to-end functionality test suite

The below details a series of tests, intended to confirm the functionality of the application from both a tutor and student's perspective. Further detail is given in section 4.4.1.

Test Number	Details of activities	Use cases/functions tested
1	Log into system as system administrator	This tests: • Sub-function #1 • Initial set up and automatic creation of first tutor. This is not specifically part of a use case, but is part of the initial "first run" set up – see development log for sprint 4.
2	Create a new "superuser" tutor: • Select **Tutors** from management menu • Select **Add a new tutor** • Enter tutor name and login ID[9] • Click **Save changes** • Edit the new tutor by clicking on the "pencil" icon next to their name • Enable the **Superuser** option • Click **Save changes** • Log out of the system (**Return to main menu**, followed by **Log out of WLab**)	Intended to test the creation of new tutors, and the ability to edit existing tutors. These functions were never directly enumerated by the user community or documented in use cases, but are required system management tools.
3	Configure new VMs based on the minimal XP image: • Log into the system as the tutor created in test 1 • Select **Virtual Machines** • Find the minimal XP image in the list of VMs and select **Clone VM**. Name the new VM "Test VM 1" and click **OK.** • Once the VM console appears, configure the VM as follows: ○ Create a new text file on the desktop called "stage 1". Make the VM safe for cloning (see System Administrator and Tutor documentation) ○ Shut down the VM • Repeat the process describe above to create a second VM. This time, use the VM created in the previous step as the basis for your new one. Call it "Test VM 2" and the file on the desktop should be called "stage 2". • Select **Return to main menu**.	This tests: • Use case #7 • Sub-function #10 • Sub-function #11 • Sub-function #13 • Sub-function #18 • Sub-functions #19, #20 and #21 (both backend VMs and lab stage VMs are displayed, and are filterable) • Sub-function #14
4	Create a new lab stage: • From the management menu (i.e. the menu that appears when a tutor logs in), select **Lab Stages**. • Select **Create a new lab stage**. • Give the new stage the name, *Test Stage 1*. • Select the VM created in step 3. Click **Set** next to the field **Starting point VM for this stage**.	This tests: • Use case #6 • Sub-function #22 • Sub-functions #24 and #25

[9] The login ID needs to be a valid login on the backend directory server.

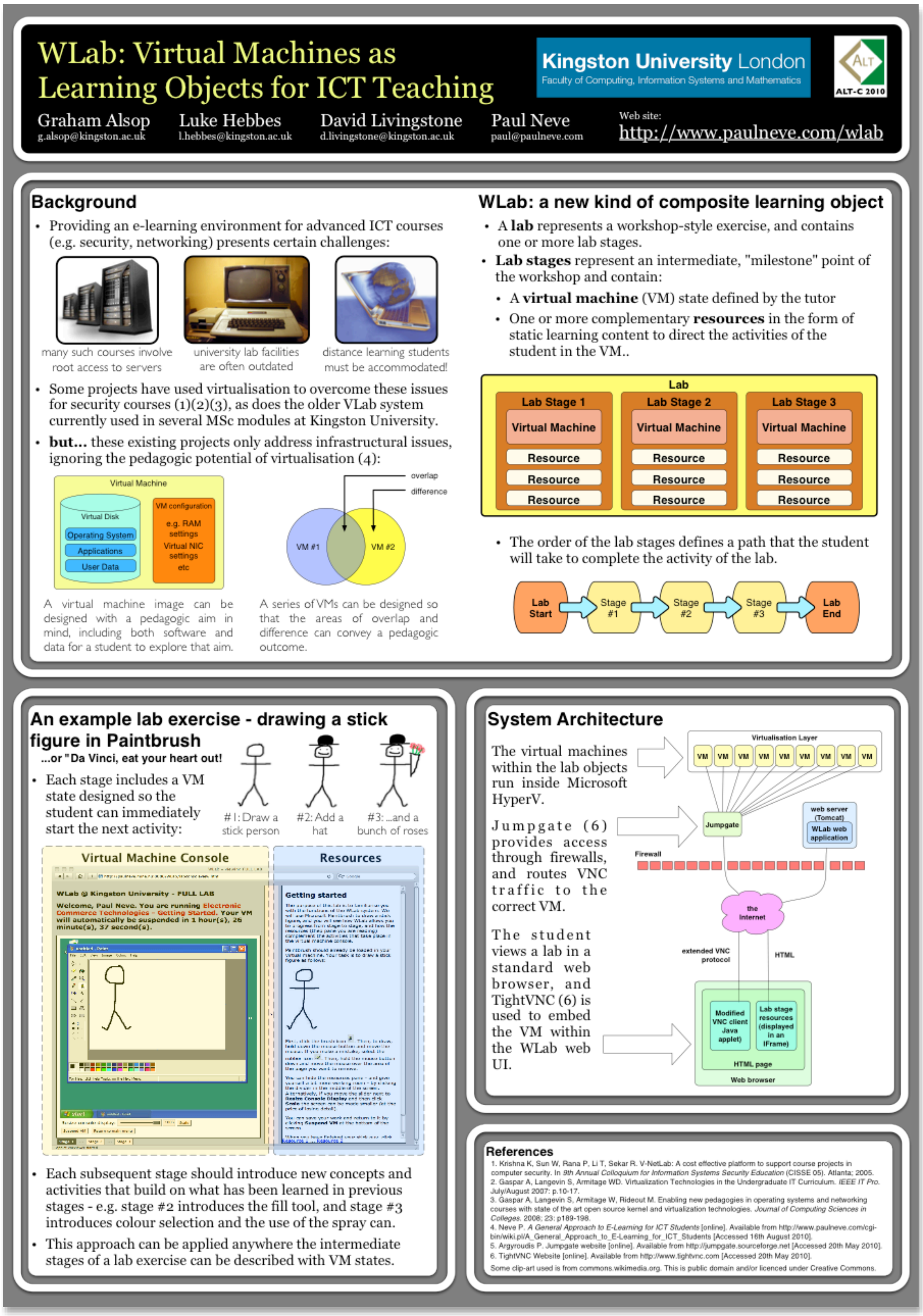
Test Number	Details of activities	Use cases/functions tested
	• Enter **http://www.google.com** into the resources field. • Click **Save changes**.	
5	Create a new lab for the Paintbrush example exercise: • From the management menu, select **Labs**. • Select **Create a new lab**. • Give the lab the name *Test Lab,* a course name *Test Course,* and a course number *TEST1*. • Select the lab stage created in step 4, and use the >> button to add it to the lab. • Add a student ID in the students field. • Click **Save changes.**	This tests: • Dynamic creation of course objects where appropriate (see development log for sprint 4) • Dynamic creation of student objects • Use case #8 v2 • Sub-function #17 • Sub-function #26 • Sub-function #30 • Sub-function #31
6	Examine initial details of student and their progress in the lab: • From the management menu, select **Students**. • Select the student created in stage 5 (i.e. the one added to the lab). Check the lab is shown, with a single lab stage, and shows “Not started yet”. • Select **Cancel** to return to the previous screen, and **Return to main menu**. • Select **Labs**. • Select the new lab created in stage 5. • Click the **Student Details** tab. • Similarly, check the single student is displayed, with the single lab stage shown as “Not started yet”. • Log out of WLab.	This tests: • Dynamic creation of student objects when student ID is specified on a lab. • Initial population of lab stage/student progress records. • Sub-function #27.

Test Number	Details of activities	Use cases/functions tested
7	Test student functionality: • Log in as the student who was previously added to the lab created in previous stages. • There should be a single lab listed – i.e. the one created previously. Its status should show *Not created yet*. Click **Create Lab.** • After a few minutes – during which you should see the message *Creating a new lab* – you should see the options change to **Suspend lab** and **Connect to lab**. Click the latter. You should see the lab console displayed alongside the Google website (the "resource"). • Make a change in the console (for example, move a desktop icon). Then select **Take snapshot.** • Click **Revert to stage start** and reply **OK** to the warning that will appear. This should undo all your changes (so the icon should move back to where it started). • Click **Revert to snapshot** and reply **OK** to the warning that will appear. This should restore your changes. • At the top of the screen, click the appropriate link to mark the stage as finished. • Click **Suspend lab.** • Log out of WLab.	This tests: • Use case #1v2 • Use case #2 • Use case #5v3 • Sub-function #12 • Sub-functions #32 & #33 • Sub-function #29
8	Test automatic lab suspension facility: • Log into WLab as the tutor created in step 2. • Select **Students** and click on the student (there should still only be one). • Set their max runtime to three minutes. • Log out and re-login as the student. • Click **Start Lab**. Once the lab has started, you should see a countdown from approximately three minutes above the lab controls. • **Connect to lab.** The same countdown should be displayed above the VM console. • Wait until the countdown expires. You should be returned to the initial main menu. The lab controls will be suppressed and a countdown will show how long remains before the student can re-start their labs.	This tests: • Use case #3 v2

Test Number	Details of activities	Use cases/functions tested
9	Misc: • Log in to WLab as the original system administrator – NOT the tutor you created in step 2. • Select **Students**. • Select **Show only "my" students** (none should be displayed). • Select **Show all students**. • Click on the student used in step 7. • Confirm that the single lab and single lab stage shown as a progress of "Finished". • Click **Cancel** and **Return to main menu.**	This tests: • Tutor viewing of updaded student progress records • Sub-function #28
10	Test adding a new lab stage to an existing lab: • From the management menu (i.e. the menu that appears when a tutor logs in), select **Lab Stages**. • Select **Create a new lab stage**. • Give the new stage the name, *Test Stage 2*. • Select the *second* VM created in step 3. Click **Set** next to the field **Starting point VM for this stage**. • Enter **http://www.wikipedia.org** into the resources field. • Click **Save changes**. • Go back to the management menu and select **Labs**. • Choose *Test Lab*. • Highlight *Test Stage 2* and use the >> button to add it to the lab. • Select **Save changes.** • Select **Log out of WLab**.	This tests: • Sub-function #31
11	Test navigation between lab stages for students: • Log in as your student. • Click **Start Lab** against your lab. • Once it has started, **Connect to lab.** • At the bottom of the lab console screen, you should now see that there is a second box marked **Stage 2**. Click it. • After two minutes or so, you should be connected to the second lab stage: o You should have a desktop icon marked "stage 2" o The resource pane should be showing Wikipedia. • Suspend the lab.	This tests: • Student navigation between lab stages
12	Test student booking facilities: • Click the link to the **Booking Page**. • Click the **Go to date** field and select a date from the date-picker.	This tests: • Sub-functions #15 and #16.

Appendix E – Poster presentation at ALT-C 2010

The following is a poster and accompanying two page handout which was presented at ALT-C 2010:



WLab: Virtual Machines as Learning Objects for ICT Teaching

Kingston University London
Faculty of Computing, Information Systems and Mathematics

ALT-C 2010

Graham Alsop g.alsop@kingston.ac.uk · Luke Hebbes l.hebbes@kingston.ac.uk · David Livingstone d.livingstone@kingston.ac.uk · Paul Neve paul@paulneve.com

Web site: http://www.paulneve.com/wlab

Background

- Providing an e-learning environment for advanced ICT courses (e.g. security, networking) presents certain challenges:

many such courses involve root access to servers · university lab facilities are often outdated · distance learning students must be accommodated!

- Some projects have used virtualisation to overcome these issues for security courses (1)(2)(3), as does the older VLab system currently used in several MSc modules at Kingston University.
- **but...** these existing projects only address infrastructural issues, ignoring the pedagogic potential of virtualisation (4):

A virtual machine image can be designed with a pedagogic aim in mind, including both software and data for a student to explore that aim.

A series of VMs can be designed so that the areas of overlap and difference can convey a pedagogic outcome.

WLab: a new kind of composite learning object

- A **lab** represents a workshop-style exercise, and contains one or more lab stages.
- **Lab stages** represent an intermediate, "milestone" point of the workshop and contain:
  - A **virtual machine** (VM) state defined by the tutor
  - One or more complementary **resources** in the form of static learning content to direct the activities of the student in the VM..

- The order of the lab stages defines a path that the student will take to complete the activity of the lab.

An example lab exercise - drawing a stick figure in Paintbrush

...or "Da Vinci, eat your heart out!

- Each stage includes a VM state designed so the student can immediately start the next activity:

#1: Draw a stick person
#2: Add a hat
#3: ...and a bunch of roses
Virtual Machine Console
Resources
WLab @ Kingston University - FULL LAB
Welcome, Paul Neve. You are running Electronic Commerce Technologies - Getting Started. Your VM will automatically be suspended in 1 hour(s), 26 minute(s), 37 second(s).
Getting started

- Each subsequent stage should introduce new concepts and activities that build on what has been learned in previous stages - e.g. stage #2 introduces the fill tool, and stage #3 introduces colour selection and the use of the spray can.
- This approach can be applied anywhere the intermediate stages of a lab exercise can be described with VM states.

System Architecture

The virtual machines within the lab objects run inside Microsoft HyperV.

Jumpgate (6) provides access through firewalls, and routes VNC traffic to the correct VM.

The student views a lab in a standard web browser, and TightVNC (6) is used to embed the VM within the WLab web UI.

References

1. Krishna K, Sun W, Rana P, Li T, Sekar R. V-NetLab: A cost effective platform to support course projects in computer security. In *9th Annual Colloquium for Information Systems Security Education* (CISSE 05). Atlanta; 2005.
2. Gaspar A, Langevin S, Armitage WD. Virtualization Technologies in the Undergraduate IT Curriculum. *IEEE IT Pro.* July/August 2007: p.10-17.
3. Gaspar A, Langevin S, Armitage W, Rideout M. Enabling new pedagogies in operating systems and networking courses with state of the art open source kernel and virtualization technologies. *Journal of Computing Sciences in Colleges.* 2008; 23: p189-198.
4. Neve P. *A General Approach to E-Learning for ICT Students* [online]. Available from http://www.paulneve.com/cgi-bin/wiki.pl/A_General_Approach_to_E-Learning_for_ICT_Students [Accessed 16th August 2010].
5. Argyroudis P. Jumpgate website [online]. Available from http://jumpgate.sourceforge.net [Accessed 20th May 2010].
6. TightVNC Website [online]. Available from http://www.tightvnc.com [Accessed 20th May 2010].

Some clip-art used is from commons.wikimedia.org. This is public domain and/or licenced under Creative Commons.

WLab: Virtual Machines as Learning Objects for ICT Teaching

Kingston University London
Faculty of Computing, Information Systems and Mathematics

Graham Alsop (*g.alsop@kingston.ac.uk*)
David Livingstone (*d.livingstone@kingston.ac.uk*)
Luke Hebbes (*l.hebbes@kingston.ac.uk*)
Paul Neve (*paul@paulneve.com*)

1. The difficulties of delivering practical workshop exercises in ICT teaching

ICT subjects present unique difficulties when attempting to deliver practical workshops as part of the learning experience. Institutional IT labs often contain outdated equipment, pre-configured to a "lowest common denominator" template, ill-suited to the needs of ICT teaching. Many subjects within ICT require administrative access to a server if the student is to be able to practice the skills being taught. Finally, distance-learning students present their own problems, and are usually expected to acquire and/or reconfigure home equipment for use during a course.

2. Introducing WLab

The WLab project addresses these issues by using virtual machines to deliver a workshop environment to the student which can be designed by the tutor to specifically suit the needs of the course being taught. At the heart of the system is an innovative composite learnng object:

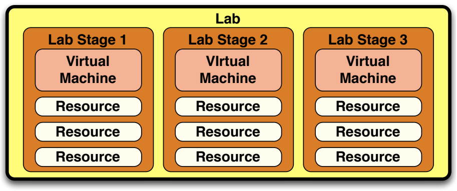
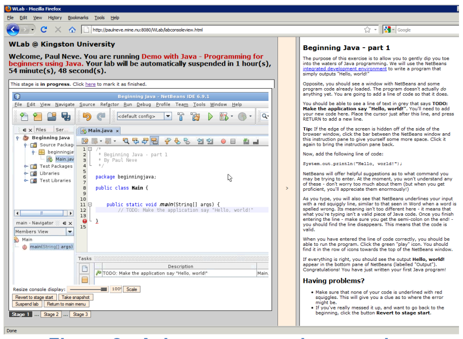


Figure 1 - WLab's composite learning object

Lab exercises are divided into stages, each of which contains a distinct virtual machine. This virtual machine contains all of the software and data required for this stage of the lab. Alongside the virtual machine, static learning materials are presented which describe the activities and provide a learning context:



WLab @ Kingston University

Welcome, Paul Neve. You are running Demo with Java - Programming for beginners using Java. Your lab will be automatically suspended in 1 hour(s), 54 minute(s), 48 second(s).

This stage is **in progress**. Click here to mark it as finished.

```
/*
 * Beginning Java - part 1
 * By Paul Neve
 */

package beginningjava;

public class Main {


    public static void main(String[] args) {
        // TODO: Make the application say "Hello, world!"

}
```

TODO: Make the application say "Hello, world!" | Main.

Beginning Java - part 1

The purpose of this exercise is to allow you to gently dip you toe into the waters of Java programming. We will use the NetBeans integrated development environment to write a program that simply outputs "Hello, world!"

Opposite, you should see a window with NetBeans and some program code already loaded. The program doesn't actually *do* anything yet. You are going to add a line of code so that it does.

You should be able to see a line of text in grey that says **TODO: Make the application say "Hello, world!"**. You'll need to add your new code here. Place the cursor just after this line, and press RETURN to add a new line.

Tip: If the edge of the screen is hidden off of the side of the browser window, click the bar between the NetBeans window and this instruction pane to give yourself some more space. Click it again to bring the instruction pane back.

Now, add the following line of code:

```
System.out.println("Hello, world!");
```

NetBeans will offer helpful suggestions as to what command you may be trying to enter. At the moment, you won't understand any of these - don't worry too much about them (but when you get proficient, you'll appreciate them enormously!)

As you type, you will also see that NetBeans underlines your input with a red squiggly line, similar to that seen in Word when a word is spelled wrong. Its meaning isn't too different here - it means that what you're typing isn't a valid piece of Java code. Once you finish entering the line - make sure you get the semi-colon on the end! - you should find the line disappears. This means that the code is valid.

When you have entered the line of code correctly, you should be able to run the program. Click the green "play" icon. You should find it in the row of icons towards the top of the NetBeans window.

If everything is right, you should see the output **Hello, world!** appear in the bottom pane of NetBeans (labelled "Output"). Congratulations! You have just written your first Java program!

Having problems?

- Make sure that none of your code is underlined with red squiggles. This will give you a clue as to where the error might be.
- If you've really messed it up, and want to go back to the beginning, click the button **Revert to stage start**.

Figure 2 - A Java programming exercise

WLab can be used during conventional, on-campus workshop sessions to alleviate the issues presented by limited equipment available in on-site IT labs. It can also be used to deliver a consistent workshop environment to distance learning students. However, WLab not only resolves these infrastructural difficulties, but also provides pedagogic benefits.

4. The authoring and delivery processes in a nutshell

- The tutor designs a lab exercise by using an existing virtual machine (supplied by a system administrator, or a previous one created by them or another tutor) as an initial template.
- The tutor modifies the virtual machine to provide an environment for the 1st stage of the lab exercise.
- The tutor creates subsequent virtual machines for the next stages, by cloning the previous and using that as the base. This essentially means that the tutor "works through" the activity of the lab as they author it:

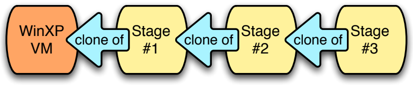
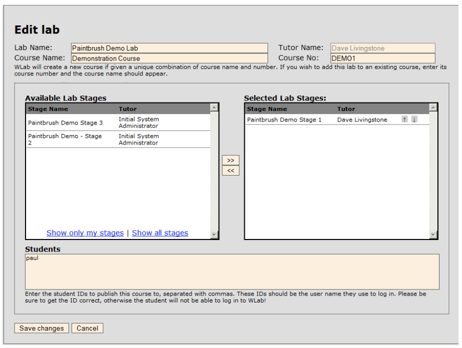


Figure 3 - Virtual machines within lab stages build on previous stages

- The tutor creates lab stages, and assigns the virtual machines to them.
- The tutor publishes the lab to students.



Lab Name:	Paintbrush Demo Lab	Tutor Name:	Dave Livingstone
Course Name:	Demonstration Course	Course No:	DEMO1

WLab will create a new course if given a unique combination of course name and number. If you wish to add this lab to an existing course, enter its course number and the course name should appear.

Available Lab Stages

Stage Name	Tutor
Paintbrush Demo Stage 3	Initial System Administrator
Paintbrush Demo - Stage 2	Initial System Administrator

Selected Lab Stages:

Stage Name	Tutor
Paintbrush Demo Stage 1	Dave Livingstone

Students
paul

Enter the student IDs to publish this course to, separated with commas. These IDs should be the user name they use to log in. Please be sure to get the ID correct, otherwise the student will not be able to log in to WLab!

Figure 4 – Assembling the lab stages into a lab

- When a student runs the lab, on demand WLab will create clones of the virtual machines within the lab for the student's exclusive use.
- Students are also provided with a booking system, so that they can specify times that suit them for their workshop activities.

4. What IT people will need to know

WLab is a Java web application that requires Apache Tomcat or similar servlet container. The virtualisation backend currently supported is Microsoft Hyper-V (although the application is written so that developers could easily write support for other products e.g. VMWare). The application authenticates against an LDAP server – currently Microsoft Active Directory is tested and supported.

To provide access to virtual machine consoles, the TightVNC Java Applet (1) is used; virtual machines will thus require a VNC server to be installed within them.

For access remotely (i.e. outside your institution) the port forwarding application Jumpgate (2) is used to forward traffic from a single external network address/port to internally addressed virtual machines.

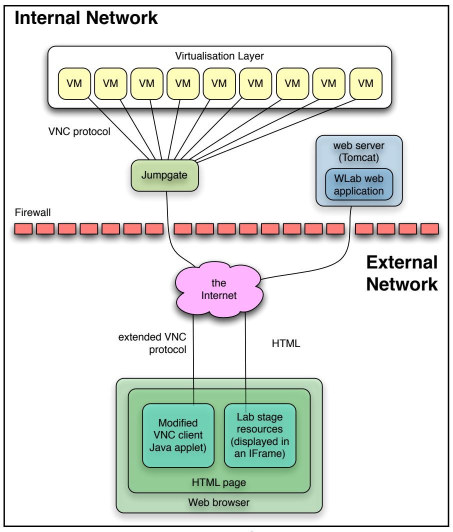


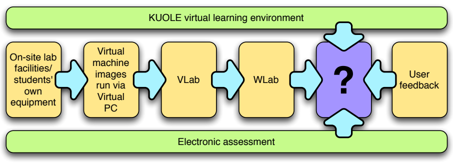
Figure 5 – The topology of a WLab installation

5. The pedagogic advantages of the WLab approach

- A student can navigate between lab stages and compare their current work with the exemplar provided by subsequent or future stages. This quick navigation provides worthwhile learning content that has no obvious analogue in a conventional workshop environment.
- Williams notes the importance of the context of learning objects (3). In WLab, a practical activity is put into context via its accompanying static materials. These will always be delivered alongside the practical aspect of the lab activity as a composite, coherent learning object. In a conventional approach, one might use printed handouts as an analogue. However, consider the scenario of a student who takes such a printed handout away from a workshop for home study. Without having the equipment and software environment discussed in the handout available to refer to, the handout may be of limited or even no value to the student in isolation.
- An entire lab exercise is represented as computer files. Tutors can share their ICT labs as fully self contained, ready to run learning objects, which can then be deployed with little or no changes on any WLab installation. This opens up the possibility of an inter-institutional repository of such labs as open educational resources.

5. Future work

WLab represents an iteration of KU's work to create learning objects that describe a complete ICT workshop exercise, and the tools required to deliver them to students:


KUOLE virtual learning environment
On-site lab facilities/ students' own equipment
Virtual machine images run via Virtual PC
VLab
WLab
?
User feedback
Electronic assessment

The WLab application has recently reached the end of its first development phase and is considered to be in a usable state for an initial rollout. Work in the short term will therefore concentrate on its roll out to students and tutors within Kingston University, and the subsequent analysis of the application's effectiveness within a real-world teaching environment. The authors would be extremely interested in collaborating with other institutions and/or individuals who can see a role for WLab in their own activities, and who wish to experiment with the application in their own environment.

Future work will seek to integrate WLab with other e-learning research occurring at KU. Work on electronic assessment and integration with VLEs, including KU's home-grown KUOLE is also seen as key.

6. For more information

The WLab web site contains comprehensive documentation, with sections targeted at tutors, IT support staff, developers and students. There is also a community forum for discussion and collaboration. It can be found at http://www.paulneve.com/wlab.

7. References

1. TightVNC website [online]. Available from http://www.tightvnc.com. (Accessed 17th August 2010).
2. Jumpgate website [online]. Available from http://sourceforge.jumpgate.net. (Accessed 17th August 2010).
3. Williams DD. Evaluation of learning objects and instruction using learning objects. In: Wiley DA (ed.) *The Instructional Use of Learning Objects* [online]. Association for Instructional Technology; 2001. Available from http://www.reusability.org/read/chapters/williams.doc.

Appendix F – Licencing

The author's intention was always to make WLab an open source project, as it was felt that this would increase the likelihood of subsequent community involvement. However, the use of code fragments from the JASMINe project potentially limits the licence used for WLab to the LGPL.

There is an argument that the amount of code used is so small, representing the tiniest fraction of both WLab and JASMINEe, that the LGPL does not apply, particularly as the activities involved deal with common tasks e.g. "start virtual machine", "suspend virtual machine" that realistically can only be achieved in one way in code, given WLab's use of the same Java libraries in order to access a Hyper-V virtualisation layer.

However, an expert opinion was sought from OSS Watch (www.oss-watch.ac.uk). Their response was as follows:

> Thanks for your question, I'm afraid we are not able to give as clear an answer as you would perhaps like. In the strictest interpretation of the licence any reuse invoks the LGPL licence.
>
> Arguing that the use of "String" would invoke the licence is not appropriate as "String" itself does not contain any IP in isolation, nor is it possible to say that it was copied from a specific source since it is a standard language construct. However use of "String" in a given context may indicate the primary source e.g.
>
> **public String aReallyUnusualName = "fooBar";**
>
> However, it can be argued there is no IP in the above, but if we then also copy another line we may be starting to bring IP into the equation, e.g.
>
> **encode(aReallyUnusualName);**
>
> If we now copy the "encode" method, which uses a complex mathematical operation on the string it starts to become clear that IP has been copied.
>
> The issue is not so much "how much" code, but rather the IP embodied in the copied code and the risk that the owner of that IP might seek to enforce the applied licence.
>
> You need to perform a risk evaluation with respect to the IP embodied in the code you have adapted. The zero risk approach would be to reuse none of the open source code, even to the extent of avoiding reading other peoples code (too late now).
>
> Once you have completed this risk assessment you may decide that there is no IP embodied in the code and that there is little or no risk of a legal case being brought against you. In which case you may choose to proceed without applying the LGPL (NOTE: for your own protection you should consult a lawyer before adopting this approach).
>
> An alternative approach would be to approach the copyright holders of the original code and ask for permission to use your modified code without invoking the LGPL.

Consequently, the simplest approach appears to be to use the LGPL for WLab's own distribution.

The other components used in WLab are distributed under the following licences:

Software Name and URL	Usage	Licence
Binny V Abraham's calendar Javascript http://www.openjs.com/scripts/ui/calendar/	In entirety, with minor changes	New BSD
Carpe Slider Javascript http://carpe.ambiprospect.com/slider/archive/v1.3/	In entirety	See below
Jumpgate http://jumpgate.sourceforge.net/	In entirety, with modifications	Simplified BSD
TightVNC Java applet client http://www.tightvnc.com	In entirety, with modifications	LGPL
Robert Hashemian's Javascript countdown script http://www.hashemian.com	In enterety, with modifications	See below

In contrast to the LGPL, the BSD licence used in Jumpgate is very liberal and simply requires you to acknowledge the copyright and disclaimers in the licence, but otherwise there are few restrictions on use of the software, and there is no obligation to make any modified source available if you distribute the binaries of your modifications. However, in the interests of promoting community development our modifications are available with the rest of the source code in the WLab SVN repository.

The Carpe slider is "linkware for non-commercial use in normal web pages (forms). Put a small link near the slider or at the bottom of your page **or with your other copyright info**" (our emphasis). An appropriate link has therefore been placed on the WLab website in the appropriate place. This fulfils the terms of the licence.

Finally, Robert Hashemian's countdown script specifies in the script source, "You may use this code in any manner so long as the author's name, Web address and this disclaimer is kept intact".

Appendix G – XML Schema

The initial versions of this schema were created using the first cut data model in Figure 20 as a basis. The schema evolved further during the development process. The final version below was created using a combination of the code-to-XML features of JiBX, combined with hand-tailoring in order to fully describe the final data model as per **Error! Reference source not found.**.

Certain elements of the schema used in the KUOLE VLE are incorporated here, so as to simplify future integration.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://paulneve.org/wlab/datamodel" elementFormDefault="qualified"
targetNamespace="http://paulneve.org/wlab/datamodel">

    <!-- XML schema for WLab data model. -->

    <!-- all WLab objects extend WLabObject -->
    <xs:element type="tns:WLabObject" name="WLabObject"/>
    <xs:complexType name="WLabObject">
        <xs:sequence>
            <xs:element type="xs:string" name="id" minOccurs="1" maxOccurs="1"/>
            <xs:element type="xs:string" name="name" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Tutor - surprisingly, represents a tutor on the system.
        Set superUser to indicate that they are a superuser, and can thus view/edit ALL
        courses/labs etc, and can also create new tutors -->
    <xs:element type="tns:tutor" substitutionGroup="tns:WLabObject" name="tutor"/>
    <xs:complexType name="tutor">
        <xs:complexContent>
            <xs:extension base="tns:WLabObject">
                <xs:attribute type="xs:boolean" use="required" name="superUser"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Student - again, surprisingly, represents a student on the system -->
    <xs:element type="tns:student" substitutionGroup="tns:WLabObject" name="student"/>
    <xs:complexType name="student">
        <xs:complexContent>
            <xs:extension base="tns:WLabObject">
                <xs:sequence>
                    <!--  student may have 0..* of these -->
                    <xs:element type="xs:string" name="labProgressId" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element type="xs:string" name="bookingId" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element type="xs:string" name="vmId" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute type="xs:int" use="required" name="maxRuntime"/>
                <xs:attribute type="xs:int" use="required" name="maxBookings"/>
                <xs:attribute type="xs:int" use="required" name="maxConcurrentVMs"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Lab object. Represents a lab exercise. -->
    <xs:element type="tns:lab" substitutionGroup="tns:WLabObject" name="lab"/>
    <xs:complexType name="lab">
        <xs:complexContent>
            <xs:extension base="tns:WLabObject">
                <xs:sequence>
                    <!-- strictly speaking there will always be an owningTutor and
                       for labs created in the UI, but it's not compulsory, so minOccurs
                       = 0 to assist those creating XML by hand -->
                            <xs:element type="xs:string" name="owningTutorId"
        minOccurs="0" maxOccurs="1"/>
                            <xs:element type="xs:string" name="courseId" minOccurs="0"
        maxOccurs="1"/>
                            <xs:element type="xs:string" name="labStageId" minOccurs="0"
        maxOccurs="unbounded"/>
                            <xs:element type="xs:string" name="labProgressId"
        minOccurs="0" maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>

            <!-- LabStage. represents a single point of a lab exercises.  -->
            <xs:element type="tns:labStage" substitutionGroup="tns:WLabObject"
        name="labStage"/>
            <xs:complexType name="labStage">
                <xs:complexContent>
                    <xs:extension base="tns:WLabObject">
                        <xs:sequence>
                            <xs:element type="xs:string" name="owningLabId" minOccurs="0"
        maxOccurs="1"/>
                            <!--  a labStage created by hand might not need an owning
                              tutor -->
                            <xs:element type="xs:string" name="owningTutorId"
        minOccurs="0" maxOccurs="1"/>
                            <!-- must have a VM, not much of a stage without one -->
                            <xs:element type="xs:string" name="vmId" minOccurs="1"
        maxOccurs="1"/>
                            <!-- can only be max one endPointVm, but not compulsory -->
                            <xs:element type="xs:string" name="endPointVmId"
        minOccurs="0" maxOccurs="1"/>
                            <!-- 0..* resources -->
                            <xs:element type="xs:string" name="resourceId" minOccurs="0"
        maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>


            <!-- A labProgress represents a link between the student, a lab and the
               stageProgresses (and thus the stages of the lab). Many (or no)
               stageProgresses may exist here (but only one lab and student) -->
            <xs:element type="tns:labProgress" substitutionGroup="tns:WLabObject"
        name="labProgress"/>
            <xs:complexType name="labProgress">
                <xs:complexContent>
                    <xs:extension base="tns:WLabObject">
                        <xs:sequence>
                            <xs:element type="xs:string" name="stageProgressId"
        minOccurs="0" maxOccurs="unbounded"/>
                            <!-- must have a lab and student - meaningless without -->
                            <xs:element type="xs:string" name="labId" minOccurs="1"
        maxOccurs="1"/>
                            <xs:element type="xs:string" name="studentId" minOccurs="1"
        maxOccurs="1"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>

            <!-- A stageProgress represents a link between a labProgress and a stage. See
               above. It also contains details of the progress made by the student in
               this stage. -->
            <xs:element type="tns:stageProgress" substitutionGroup="tns:WLabObject"
        name="stageProgress"/>
            <xs:complexType name="stageProgress">
                <xs:complexContent>
                    <xs:extension base="tns:WLabObject">
                        <xs:sequence>
                            <xs:element name="progress" minOccurs="1" maxOccurs="1"
        type="tns:itemProgress"/>
                            <xs:element type="xs:string" name="labProgressId"
        minOccurs="1" maxOccurs="1"/>
                            <xs:element type="xs:string" name="labStageId" minOccurs="1"
        maxOccurs="1"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>

            <!-- ItemProgress is a simple enum type indicating progress stages. This is
               taken straight from KUOLE's Progress.xsd/NodeStatus. At the moment this
               is only used in stageProgress and also only the NotStarted, InProgress
               and Finished states are used. The complete set of values is here to
               assist future integration efforts -->
            <xs:simpleType name="itemProgress">
                <xs:restriction base="xs:string">
```

```
                <xs:enumeration value="NotStarted"/>
                <xs:enumeration value="InProgress"/>
                <xs:enumeration value="Finished"/>
                <xs:enumeration value="Reviewing"/>
                <xs:enumeration value="MarkedFinish"/>
                <xs:enumeration value="Frozen"/>
            </xs:restriction>
        </xs:simpleType>

        <!-- Course -->
        <xs:element type="tns:course" substitutionGroup="tns:WLabObject"
    name="course"/>
        <xs:complexType name="course">
            <xs:complexContent>
                <xs:extension base="tns:WLabObject">
                    <xs:sequence>
                        <xs:element type="xs:string" name="owningTutorId"
    minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <!-- resource -->
        <xs:element type="tns:resource" substitutionGroup="tns:WLabObject"
    name="resource"/>
        <xs:complexType name="resource">
            <xs:complexContent>
                <xs:extension base="tns:WLabObject">
                    <xs:sequence>
                        <!-- one resource, one URL. -->
                        <xs:element type="xs:string" name="url" minOccurs="1"
    maxOccurs="1"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <!-- Booking represents a student booking. The booking runs for an hour after
           dateTime. -->
        <xs:element type="tns:booking" substitutionGroup="tns:WLabObject"
    name="booking"/>
        <xs:complexType name="booking">
            <xs:complexContent>
                <xs:extension base="tns:WLabObject">
                    <xs:sequence>
                        <xs:element type="xs:string" name="studentId" minOccurs="1"
    maxOccurs="1"/>
                    </xs:sequence>
                    <xs:attribute type="xs:dateTime" name="date"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <!-- There are several different types of VM in WLab. All extend this generic
           VM object. -->
        <xs:element type="tns:VM" substitutionGroup="tns:WLabObject" name="VM"/>
        <xs:complexType name="VM">
            <xs:complexContent>
                <xs:extension base="tns:WLabObject">
                    <xs:sequence>
                        <!-- must have a corresponding backend VM name set so that
                          the virtualisation layer knows what VM to start/stop
                          etc -->
                        <xs:element type="xs:string" name="backendName" minOccurs="1"
    maxOccurs="1"/>
                        <xs:element type="xs:string" name="labStageId" minOccurs="0"
    maxOccurs="1"/>
                    </xs:sequence>
                    <!--  must have the X-Y size of the VM desktop set, otherwise
                          client side won't know how big to make the applet -->
                    <xs:attribute type="xs:int" use="required" name="screenSizeX"/>
                    <xs:attribute type="xs:int" use="required" name="screenSizeY"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <!-- A TutorVM is a VM created by a tutor based on an existing backend VM -->
        <xs:element type="tns:tutorVM" substitutionGroup="tns:VM" name="tutorVM"/>
        <xs:complexType name="tutorVM">
            <xs:complexContent>
                <xs:extension base="tns:VM">
                    <xs:sequence>
                        <xs:element type="xs:string" name="owningTutorId"
    minOccurs="0" maxOccurs="1"/>
                        <xs:element type="xs:string" name="rebootAfterCloning"
    minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <!-- A StudentAllocatedVM is cloned from a TutorVM, on demand, the first time
           a student accesses a lab stage. -->
        <xs:element type="tns:studentAllocatedVM" substitutionGroup="tns:VM"
    name="studentAllocatedVM"/>
        <xs:complexType name="studentAllocatedVM">
            <xs:complexContent>
                <xs:extension base="tns:VM">
                    <xs:sequence>
                        <!-- VMs created by hand might not have an original VM id,
                          hence minOccurs = 0 -->
                        <xs:element type="xs:string" name="originalVmId"
    minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>

    </xs:schema>
```

Appendix H – Resources from the "beginning programming in Java" example lab

The following is a textual representation of the HTML files that are used as resources within the "beginning programming in Java" example lab discussed in section 4.4.2, User testing / Sample labs.

Stage 1

Beginning Java - part 1

The purpose of this exercise is to allow you to gently dip you toe into the waters of Java programming. We will use the NetBeans integrated development environment to write a program that simply outputs "Hello, world!"

Opposite, you should see a window with NetBeans and some program code already loaded. The program doesn't actually *do* anything yet. You are going to add a line of code so that it does.

You should be able to see a line of text in grey that says **TODO: Make the application say "Hello, world!"**. You'll need to add your new code here. Place the cursor just after this line, and press RETURN to add a new line.

Tip: If the edge of the screen is hidden off of the side of the browser window, click the bar between the NetBeans window and this instruction pane to give yourself some more space. Click it again to bring the instruction pane back.

Now, add the following line of code:

```
System.out.println("Hello, world!");
```

NetBeans will offer helpful suggestions as to what command you may be trying to enter. At the moment, you won't understand any of these - don't worry too much about them (but when you get proficient, you'll appreciate them enormously!)

As you type, you will also see that NetBeans underlines your input with a red squiggly line, similar to that seen in Word when a word is spelled wrong. Its meaning isn't too different here - it means that what you're typing isn't a valid piece of Java code. Once you finish entering the line - make sure you get the semi-colon on the end! - you should find the line disappears. This means that the code is valid.

When you have entered the line of code correctly, you should be able to run the program. Click the green "play" icon. You should find it in the row of icons towards the top of the NetBeans window.

If everything is right, you should see the output **Hello, world!** appear in the bottom pane of NetBeans (labelled "Output"). Congratulations! You have just written your first Java program!

Having problems?

- Make sure that none of your code is underlined with red squiggles. This will give you a clue as to where the error might be.
- If you've really messed it up, and want to go back to the beginning, click the button **Revert to stage start**.

Stage 2

Beginning Java – part 2

The purpose of this exercise is now to introduce variables now that you've had some experience of using the lab environment and NetBeans.

One way to think of variables is as a box for "stuff". In this context, "stuff" can be a number, some text (referred to as a *string*), or an object (more about those later!).

We've added the following code to your program, after your Hello, World! line of code:

```
int mynumber = 7; System.out.println("My number is "+mynumber);
```

Run the program (the green play icon, remember?) and you'll see

```
Hello, world! My number is 7.
```

So, what is happening here? Firstly, there is a variable called *mynumber*. Before variables can be used in Java, they must be declared. To declare a variable, you simply specify its type followed by its name. In this case, we've declared a variable of type *int*. This is short for integer, meaning a whole (i.e. without any fractional part) number. You can also set an initial value when you declare a variable. In our case, we use this to set the initial value to 7.

Variables can be modified simply by assigning a new value, e.g.

```
mynumber = 12;
```

Try adding this line before the line where the value is printed, and run the program to see the effect. You should see that the second number assigned, i.e. 12, overwrites the first. If you go back to the "box" metaphor, what we've done is to take the first lot of stuff out of the box (i.e. the 7) and put new stuff into the box (i.e. 12).

Now move on to stage 3 of the exercise.

Stage 3

Beginning Java – part 3

The purpose of this exercise is now to introduce you to *conditional processing*, meaning the ability to make your program's behaviour different depending on certain conditions, such as the value of a variable.

Examine the code in NetBeans opposite. The variable declaration and the *System.out.println* command should be familiar to you now, but note the new command, *if*.

if works exactly as you'd expect, and is basically a case of you as the programmer telling the computer, "if ABC is the case, do XYZ". In Java, you specify the condition to be met between brackets, and the code that occurs if the condition is true between curly brackets (i.e. { and })

- Run the program. What is the result? Is it what you expected?
- Change the value assigned to *mynumber* to 15. Now run the program. Does that do what you expected?
- What do you think will happen if you change it to 10? Try it. Why do you get that result?
-

Now fix the program so that the response *10 is my favourite number* is displayed if *mynumber* is 10.

Tip: The way to compare values is using a double equals sign, i.e. ==

Appendix I – User documentation

The following is intended to provide a comprehensive suite of documentation for individuals wishing to make use of the WLab system. Our assertion is that this is a key deliverable if WLab is to acquire a user base beyond those who have been directly involved in its development thus far. This documentation is also available in wiki form, at http://www.paulneve.com/wlab.

There are four sections:

Documentation for Tutors
Intended for tutors, and others who will be using the tutor management functions to author labs. This takes the reader through the process of creating tutor VMs, placing them into lab stages and creating a lab by way of an example.

Documentation for System Administrators
Intended for those who will be providing technical support to tutors and students, and who will have overall responsibility for the installation and maintenance of the WLab system. Provides details of how to configure and set up a binary distribution of WLab.

Documentation for Developers
Intended for those who wish to modify the WLab source code for their own purposes. Provides information of how to acquire the source code, how to build an appropriate development environment, and detailed information about the structure of the application, relationships between Java classes and JSP pages, etc.

Documentation for Students
Intended for students and others who will be accessing the labs published by tutors. Takes the user through logging into the system, connecting to a lab, and using the booking system.

Documentation for Tutors

Introduction

WLab is a web application that provides a virtual lab environment for any workshop activity that requires the use of a computer. WLab makes use of virtual machines to deliver an IT environment in which a student can perform the tasks involved in the lab exercise, regardless of the local computer they are using.

For those unfamiliar with virtual machines, a virtual machine or VM is a complete simulation of an entire computer system – hardware, software and any accompanying user data. This means that the simulated computer can include everything the student needs to perform the tasks required of them in the lab exercise. Consequently, the student does not need any specific software or configuration parameters on their local PC.

WLab allows tutors to design and publish lab exercises that are divided into stages; each stage contains a complete virtual machine, along with – if required –appropriate static learning material that complements the activities of the lab. Students access these labs through any standard web browser.

Assumptions

WLab is primarily intended for use in teaching ICT-related subjects. As such, the assumption is that tutors are computer literate and are able to follow IT concepts and terminology at the level required to teach ICT. While WLab does provides a friendly user interface for tutors to create their labe, and attempts to hide the cumbersome technical details such as editing XML files and/or configuring individual VMs for each student, you will need to understand the underlying realities of what the application is actually *doing* behind the scenes.

As a yardstick, if you do not understand the instruction

*Save the file to the root of the C:\ drive as **stickguy.bmp**.*

then WLab is probably not for you - or, at the very least, you will need to co-opt some additional help in order to be able to use it!

It is also assumed that your System Administrator has already installed WLab in your environment, and made the appropriate preparations outlined in the Documentation for System Administrator. In particular, you should have been supplied with the URL of WLab at your institution, and you should also have been given details of how your System Administrator would like you to make your VMs safe for use in lab stages. If either of these assumptions are not the case, you should NOT proceed, and should consult your System Administrator for advice.

WLab elements and terminology

The following terms are used throughout this documentation, and within the WLab application itself:

Lab
A workshop-style practical exercise that a student performs with a computer intended to achieve a learning objective.

Lab stage
Each lab may be subdivided into intermediate "goals" that occur throughout the activities. These are referred to *lab stages* and a WLab lab object can contain as many or as few as you like. The end point of a given lab stage is usually considered the correct "answer" for that part of the lab exercise. Usually, subsequent stages will start from the end point of the previous, or at the very least build on the activities of the previous stage.

Virtual machine (VM)
A virtual machine that includes the required software and data, and starts up at a point appropriate for a lab stage. There are two types of VMs in WLab, a *backend VM* and a *tutor VM* – the difference will be explained later. A lab stage always contains a tutor VM.

Resource
A static document in the form of a URL, intended to provide complementary learning content to inform, instruct and assist the student with the activities of a given lab stage. A lab stage can contain multiple resources; equally, it may not contain any resources at all.

The diagram below shows how these concepts are layered to form a coherent learning object:

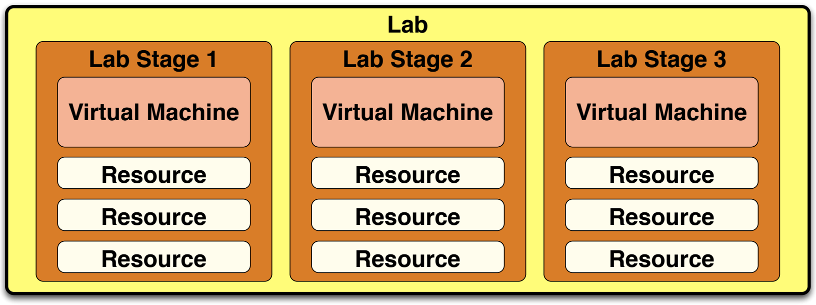


Figure 68

The lab stages, when taken in order, ultimately describe the path that the student would take from start to finish while completing the target activity of the lab:

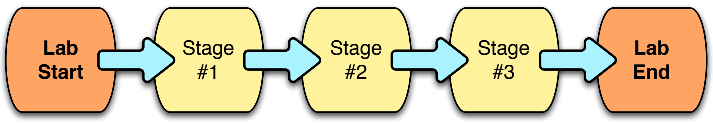


Figure 69

As previously noted, in most cases the end state of a given stage represents the starting point of the next stage.

Planning a Lab

Before attempting to build a lab in the WLab application itself, you should devote some time to planning. You will probably have a broad outline in your mind already if you are considering using WLab and are reading this documentation, but before jumping in consider the following points:

- **What is the purpose of the lab exercise?**
  You should think about what you want to actually achieve with the exercise. What are you trying to impart to your students? Do you intend the lab to be a stand-alone teaching device, with no reference to other modes of teaching? Or, is the lab simply a means for the student to practice techniques and skills that have been communicated via a different method of teaching (e.g. a lecture given previously)?

- **What activities will the student undertake during the lab?**
  Having spent some time considering the lab's purpose, you should now devote some time to examining the actual activities the student will undertake during its execution. Put yourself in a student's place and make some bullet-points as to what you would actually be *doing*, step by step.

- **What intermediate milestones or goals are there during the lab?**
  Think about where there may be natural breaks, or specific goal points where the lab can be broken up – these will define your lab stages. If you are struggling to break the lab down, you might try to identify the points where new techniques and/or functions are being introduced.

- **What software/computer configuration is needed in order to do the lab? How will this change with each subsequent lab stage?**
  Having put yourself in the place of a student undertaking the lab, consider what actual *programs* you will be using if you were to go from start to finish. You should also think about the *data* that will be needed within the programs at each lab stage.

- **Will there be any accompanying static resources?**
  If your intention was to give the students paper handouts (or indeed, any piece of static prose) containing the lab directions, you might consider breaking this material into small, bite-sized chunks, re-authoring as HTML and publishing them on the web. They can then be added as resources to the lab and will appear alongside the lab environment as the student works.

Spend some time making some bullet-point type responses to the above. Then, it is recommended that you sketch out the overall form of the lab using the diagram in Figure 68 as a guideline. This does not have to be a work of art – the motivation here is to help you firm up your thoughts about the lab, not produce something aesthetically pleasing!

Consider a very simple lab exercise – teaching students to draw in Microsoft Paintbrush. We might address the questions raised above as follows:

- The purpose of the lab is to teach basic use of MS Paintbrush.
- The activity they will perform in the lab will be to draw a stick figure. As the exercise progresses, they will incrementally add additional aspects to their drawing. Each new aspect will introduce new functions of the Paintbrush program.
- The lab will be divided into three stages:
  - Stage 1, the student will draw a basic stick figure
  - Stage 2, the student will add a hat and face to the figure
  - Stage 3, the student will add a "bunch of flowers" to their drawing
- The student will only need an environment with a version of MS Windows that includes Paintbrush. A basic installation of Windows XP will be ample. Lab stages 2 and 3 will need to include a pre-drawn version of the starting point for that stage.
- Each stage will have a single resource that gives an overview of the activities and objectives of the stage.

Ultimately, the student will be aiming for one of three “goal” pictures at the end of each of the three stages:

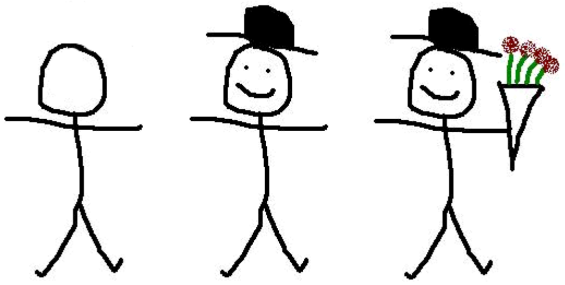
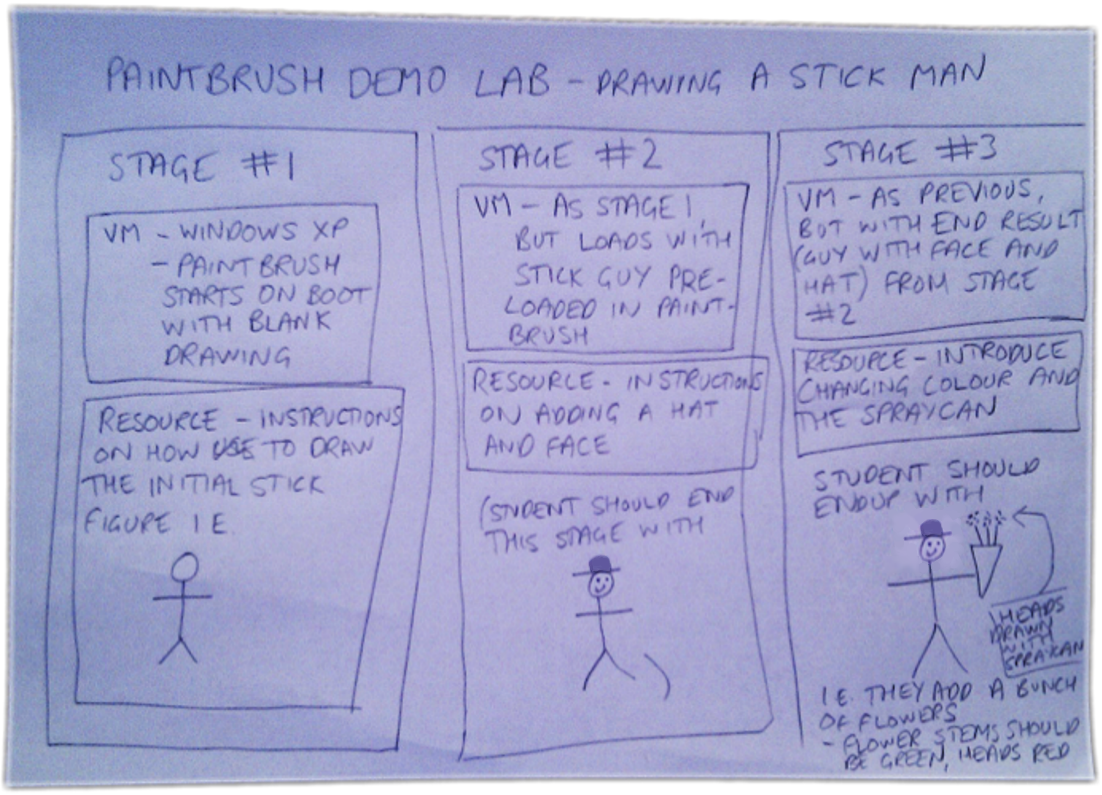


Figure 70

Note how each new stage brings in a new function of Paintbrush. The first stage introduces drawing by hand with the brush tool. The second stage introduces the fill tool, and the third introduces both colour selection and the spray can (for the “heads” of the flowers). A well-designed lab exercise will do the same, and through the activities of each subsequent stage introduce new skills to the student.

If we were following the advice given previously, and sketching out a skeleton of our lab structure, we might end up with something like the below:



PAINTBRUSH DEMO LAB - DRAWING A STICK MAN

STAGE #1

VM - WINDOWS XP
- PAINTBRUSH STARTS ON BOOT WITH BLANK DRAWING

RESOURCE - INSTRUCTIONS ON HOW ~~USE~~ TO DRAW THE INITIAL STICK FIGURE I.E.

STAGE #2

VM - AS STAGE 1, BUT LOADS WITH STICK GUY PRE-LOADED IN PAINT-BRUSH

RESOURCE - INSTRUCTIONS ON ADDING A HAT AND FACE

(STUDENT SHOULD END THIS STAGE WITH

STAGE #3

VM - AS PREVIOUS, BUT WITH END RESULT (GUY WITH FACE AND HAT) FROM STAGE #2

RESOURCE - INTRODUCE CHANGING COLOUR AND THE SPRAYCAN

STUDENT SHOULD ENDUP WITH

HEADS DRAWN WITH SPRAYCAN

I.E. THEY ADD A BUNCH OF FLOWERS
- FLOWER STEMS SHOULD BE GREEN, HEADS RED

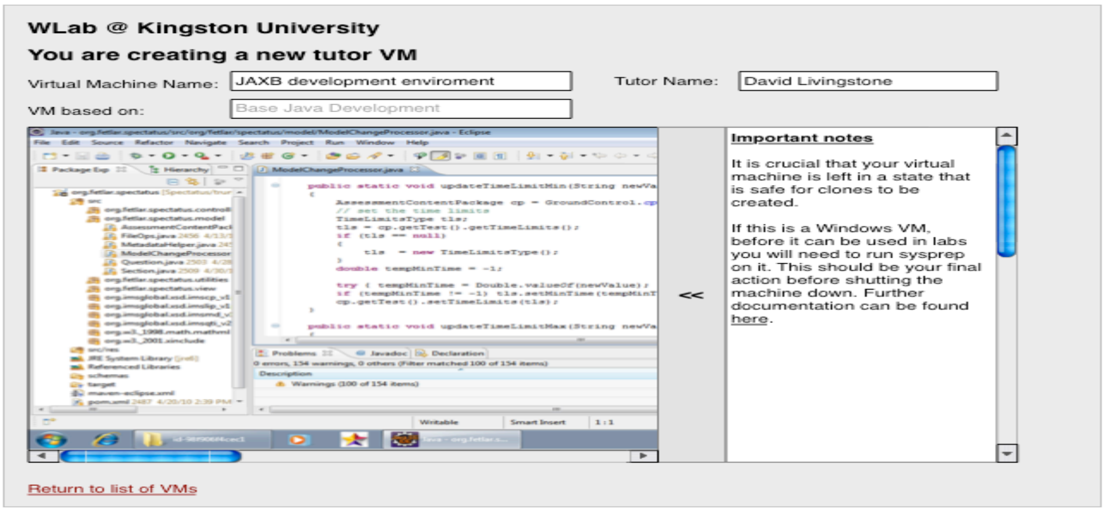
Figure 71

The best way for you to become familiar with WLab and creating your own labs is to work through an example – and given that we now have it all planned out, what better example than this Paintbrush lab? With our planning documentation in hand and in mind, we are now ready to log into WLab and start building this example lab.

It is highly recommended that you work through this entire example before trying to create an original lab of your own. This example will take you through everything you need to know to author WLab content. Once you have this grounding, you should easily be able to apply the procedure to your own content.

Building the lab

Log into WLab by opening a web browser and going to the appropriate URL (your System Administrator should have given you this). Login with your user name and password. You should be presented with the following menu:



WLab @ Kingston University

You are creating a new tutor VM

Virtual Machine Name:	JAXB development enviroment	Tutor Name:	David Livingstone
VM based on:	Base Java Development		

<<

Important notes

It is crucial that your virtual machine is left in a state that is safe for clones to be created.

If this is a Windows VM, before it can be used in labs you will need to run sysprep on it. This should be your final action before shutting the machine down. Further documentation can be found here.

Return to list of VMs

Figure 72

The menu options are ordered in a logical sequence, following the recommended order in which a tutor should create the constituent parts of a lab:

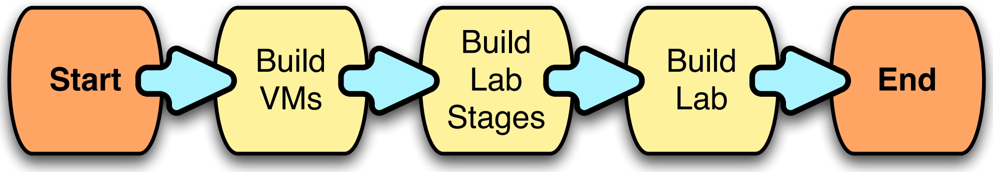


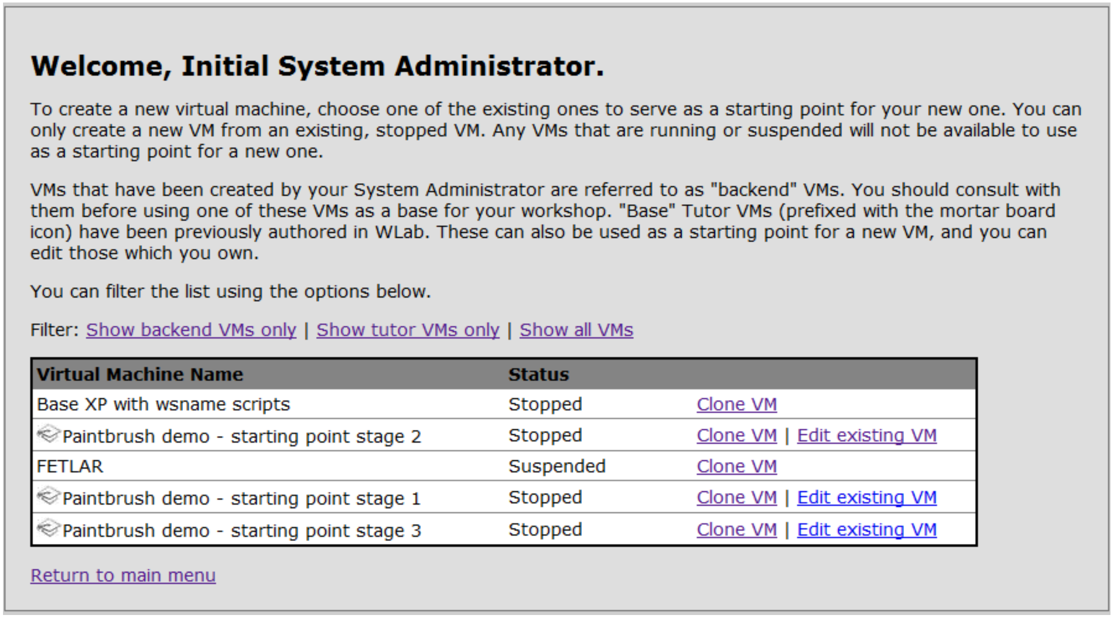
Figure 73

We will start by creating the virtual machines required for the Paintbrush example lab.

Building the virtual machines

IMPORTANT: Before attempting to create a virtual machine, you should make sure you are familiar with the procedures required to make a VM safe for use in a lab stage. Your System Administrator should have issued you with documentation that explains how to do this in your environment. If you do not have this documentation, or you do not fully understand it, you should NOT proceed. Instead, consult your System Administrator for advice.

To create a virtual machine, click on **Virtual Machines** from the main menu. You will be presented with something similar to the screen below (although you may have fewer or more VMs listed)



Welcome, Initial System Administrator.

To create a new virtual machine, choose one of the existing ones to serve as a starting point for your new one. You can only create a new VM from an existing, stopped VM. Any VMs that are running or suspended will not be available to use as a starting point for a new one.

VMs that have been created by your System Administrator are referred to as "backend" VMs. You should consult with them before using one of these VMs as a base for your workshop. "Base" Tutor VMs (prefixed with the mortar board icon) have been previously authored in WLab. These can also be used as a starting point for a new VM, and you can edit those which you own.

You can filter the list using the options below.

Filter: Show backend VMs only | Show tutor VMs only | Show all VMs

Virtual Machine Name	Status	
Base XP with wsname scripts	Stopped	Clone VM
Paintbrush demo - starting point stage 2	Stopped	Clone VM | Edit existing VM
FETLAR	Suspended	Clone VM
Paintbrush demo - starting point stage 1	Stopped	Clone VM | Edit existing VM
Paintbrush demo - starting point stage 3	Stopped	Clone VM | Edit existing VM

Return to main menu

Figure 74

At this point, it is important to distinguish between the two different types of virtual machines available to you:

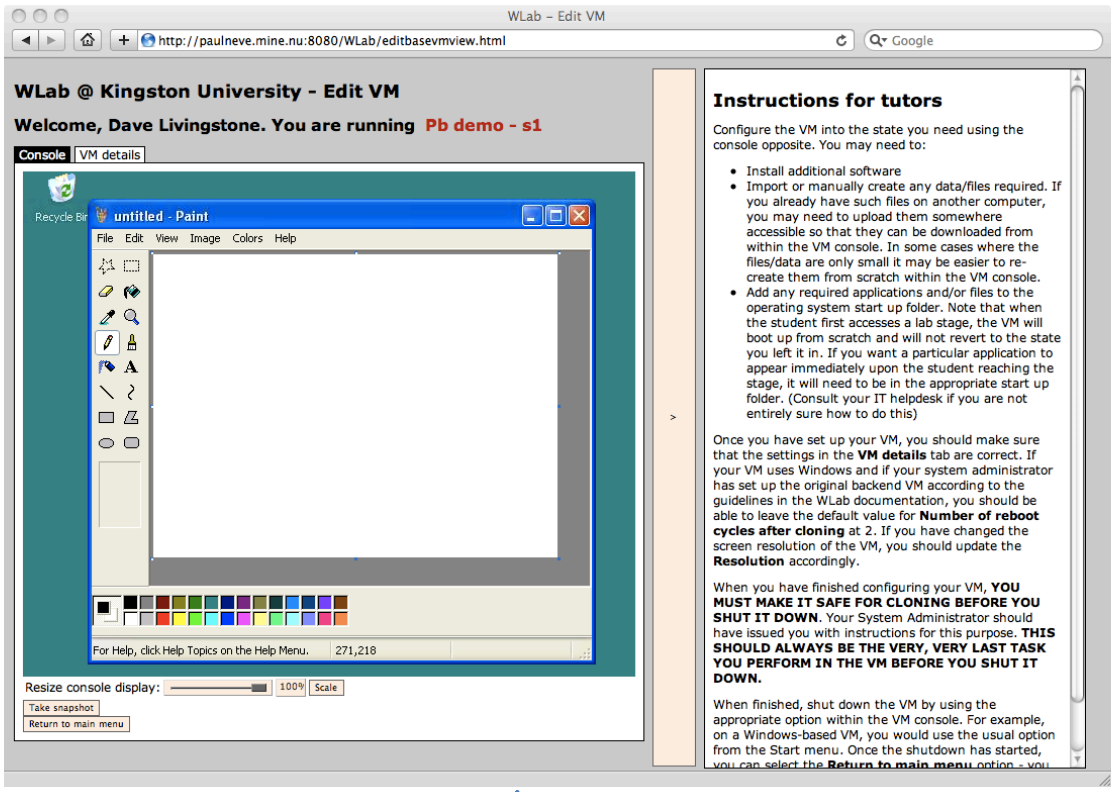
- A **backend VM** is set up by your system administrator, and can only be used as a starting point for you to create a new virtual machine for your labs. You cannot create or edit them yourself.
- A **tutor VM** is a VM that has been created by a tutor for direct use in WLab labs. These VMs can be edited after their creation (although is it highly advised not to modify them once they have been used in a lab and the students have started to use that lab). They are denoted by the mortar board icon next to them.

Creating a VM requires you to choose an existing VM that will serve as a starting point. Your system administrator should have set up some initial backend VMs that you can choose from. You should liaise with them for further details, or if you need any additional VMs created. For the example Paintbrush lab, you only need a VM with a basic installation of Windows XP. Confirm with them which particular VM this is on your system, find it in the list, and click **Clone VM**.

You will be prompted to give the new VM a name. This name needs to uniquely identify your VM, and cannot subsequently be changed. Ideally, the name should allow you to determine the lab and lab stage that the VM will be used in, so that when it appears in a list you can easily determine its purpose.

The VM we are creating at this point will eventually be used in stage 1 of the Paintbrush example lab. A name like *Paintbrush Demo – VM for stage 1* would be appropriate. Once you have supplied a name for the new VM, click *OK*.

It will take several minutes for the new VM to be created. During this time, you will see the message **Waiting for your session to come on line. Please wait.** The page may refresh itself several times during this period, and the **Waiting for your session…** message might, towards the end of the process, disappear to be replaced by a smaller message that states **Connecting to…** All of this is normal. After a few minutes, you should eventually be presented with a screen that looks similar to the following:



WLab @ Kingston University - Edit VM

Welcome, Dave Livingstone. You are running Pb demo - s1

Instructions for tutors

Configure the VM into the state you need using the console opposite. You may need to:

- Install additional software
- Import or manually create any data/files required. If you already have such files on another computer, you may need to upload them somewhere accessible so that they can be downloaded from within the VM console. In some cases where the files/data are only small it may be easier to re-create them from scratch within the VM console.
- Add any required applications and/or files to the operating system start up folder. Note that when the student first accesses a lab stage, the VM will boot up from scratch and will not revert to the state you left it in. If you want a particular application to appear immediately upon the student reaching the stage, it will need to be in the appropriate start up folder. (Consult your IT helpdesk if you are not entirely sure how to do this)

Once you have set up your VM, you should make sure that the settings in the **VM details** tab are correct. If your VM uses Windows and if your system administrator has set up the original backend VM according to the guidelines in the WLab documentation, you should be able to leave the default value for **Number of reboot cycles after cloning** at 2. If you have changed the screen resolution of the VM, you should update the **Resolution** accordingly.

When you have finished configuring your VM, **YOU MUST MAKE IT SAFE FOR CLONING BEFORE YOU SHUT IT DOWN**. Your System Administrator should have issued you with instructions for this purpose. **THIS SHOULD ALWAYS BE THE VERY, VERY LAST TASK YOU PERFORM IN THE VM BEFORE YOU SHUT IT DOWN.**

When finished, shut down the VM by using the appropriate option within the VM console. For example, on a Windows-based VM, you would use the usual option from the Start menu. Once the shutdown has started,

Figure 75

On the left hand side of the screen you will see the VM console. If you are following our example, this should be a simple Windows desktop. You can interact with the VM itself in much the same way as you would any computer – clicking on the various icons and other elements of the operating system will perform their usual tasks. On the right hand side of the screen there is an instructions pane. This is intended to remind expert users of the steps required during VM authoring.

If you want to see more of the VM console – it may be that one side of it is obscured as a result of the browser window being too small – you can hide the instructions pane by clicking the separator bar that appears between the two panes. You can also resize the VM console itself with the slider control underneath, next to the text **Resize console display**. Move the slider to the desired value, and click **Scale**. If, for example, you scale to 50%, this means that the VM console will be displayed at half of its normal size – meaning there is room for twice as much of the screen within the browser page. (Of course, the price you pay for this is a loss of detail).

If we go back and consider stage 1 of our example Paintbrush lab, we now need to configure the VM so that when it starts up, a copy of Paintbrush appears with a blank canvas ready for drawing in. For Windows XP, the simplest way to do this is to add a shortcut in **Start / All Programs / Startup** to **mspaint.exe**. Navigate the Start Menu to the Startup folder, right-click it, and select **Open.** In the window that appears, right-click again, select **New / Shortcut** and then type **mspaint.exe.** Click **Next**, and then **Finish**. (If your System Administrator has supplied you with a newer version of Windows such as Vista in the backend VM, you may need to adapt these steps slightly.)

The VM is now ready to use in stage 1 of the Paintbrush exercise. If you were creating a VM to use in a lab of your own, you might also need to install additional software, upload/create files and/or data for the lab exercise, and so on. The objective during the VM configuration is to prepare the VM so that it contains everything the student needs to perform the task required of them during the lab, and that it boots up into a position where the student can immediately continue this stage of the lab[10].

[10] A handy checklist of common activities you may need to perform when configuring a VM can be found in the Tutors' Quick Reference section.

Once the configuration of the VM is complete, **IT IS ABSOLUTELY CRUCIAL that you follow the instructions given to you by your System Administrator to make the VM safe for use in a lab stage. If you do not follow these steps, students may not be able to connect to your lab, and you may adversely affect other tutors/students' labs. As stated previously, if you do not have these instructions and/or do not understand them, you should not proceed and instead should consult your System Administrator for advice.**

Once you have made the VM safe for use in a lab stage, you should shut it down. Our example uses a Windows-based VM, so you should simply use the usual Shut Down option in the Start Menu (note *shut down* – NOT *restart*!). You will not get any visual indicator that the VM has concluded its shut down – once you have issued and confirmed the shut down command, you can click the **Return to main menu** option.

Your VM is now ready for use in a lab stage. However, what you should do now is create the VMs for stages 2 and 3. A common shortcut is to use the VM from the previous stage when you come to clone your new one, as illustrated below:

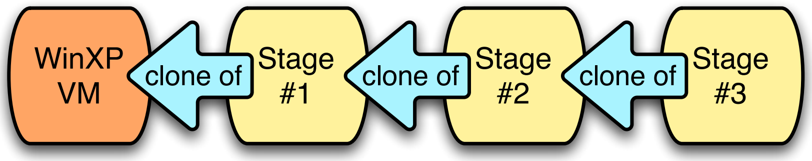


Figure 76

Because each subsequent VM is a clone of the previous, it includes all of the software, configuration and file/data of the last. The configuration of the new VM essentially entails you working through the same activities as the student would perform during that stage of the lab exercise. The advantage of this is twofold:

- You only need to perform the initial, "main" configuration required for your lab (e.g. software installation, uploading of data) once - i.e. when you create the first VM for the lab exercise.
- Working through the same activities as the student will during the lab allows you to iron out any bugs in the process as you work through creating the VMs. You might also want to, concurrently, write any accompanying handouts or documentation that will complement the lab exercise (which, later, can be turned into WLab resources).

So, to create the VMs for the Paintbrush demo lab stages 2 and 3 we will simply repeat the previous steps, but instead of cloning the original Windows XP VM, we will clone the VM from the previous stage. So, to create our VM for stage 2, find the VM for stage 1 that you just created and click **Clone VM**.

Once the cloning process is complete, you should find that Paintbrush is already visible in your VM console screen. Now draw the first stick figure (as per Figure 70). Save the stick figure to the root of the C:\ drive as **stickguy.bmp**.

You now need to modify the shortcut in the Startup menu so that it not only starts Paintbrush, but starts Paintbrush with the drawing loaded. Assuming you saved into the root of C:\ as advised, change the shortcut so it reads **mspaint.exe c:\stickguy.bmp**.

NOTE THAT THE CHANGES YOU ARE MAKING HAVE BEEN MADE ON THE FRESHLY MADE CLONE, AND NOT ON THE FIRST VM YOU CREATED. The VM for stage 1 is not being altered.

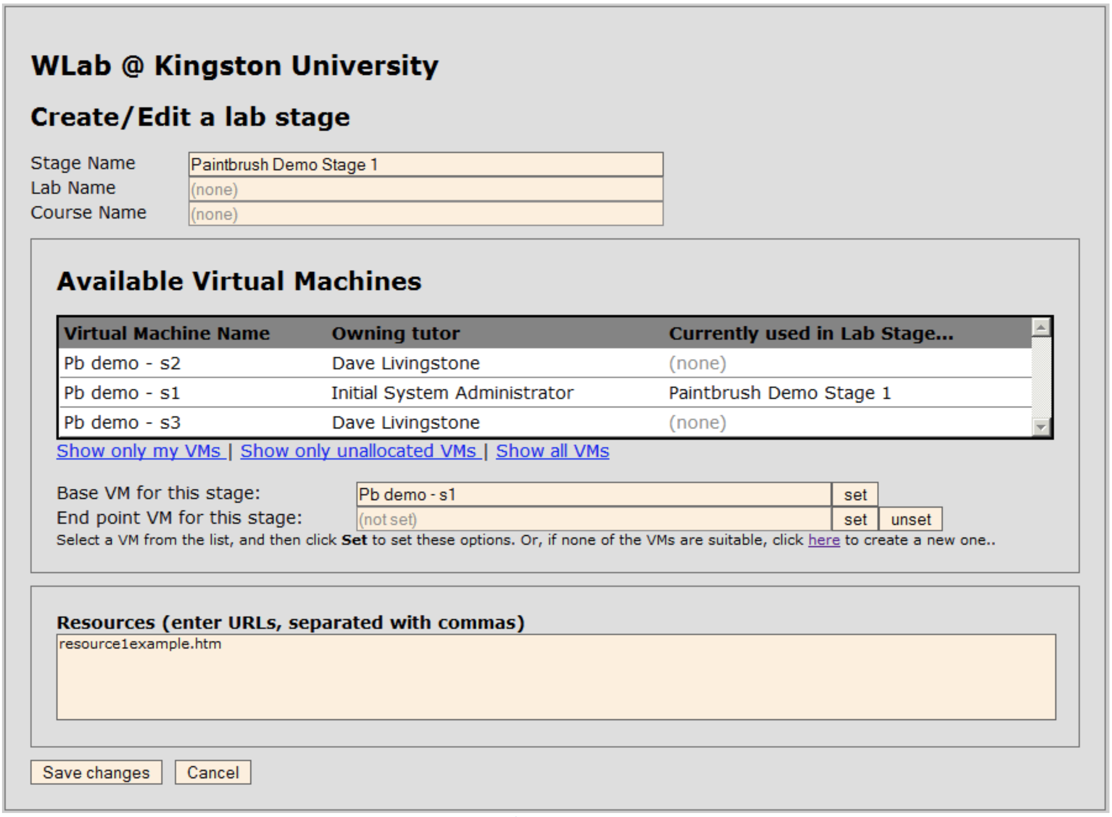
That concludes the configuration for the VM for stage 2 of this lab. As before, **follow the procedure for making the VM safe**, then shut the VM down. Select **Return to main menu.**

Now, to create the VM for stage 3, we repeat the process, except this time we will clone the stage 2 VM just created. Make the appropriate changes to the stick guy and save the graphic back to **c:\stickguy.bmp**. That's all you need to do to configure stage 3 – all the other settings have already occurred during your work on previous stages, and when you clone a previous VM to serve as the basis of the next, you get all these settings included. As always, **be sure to follow the procedure for making the VM safe,** then shut the VM down and **Return to main menu.**

You should now have three VMs set up and ready to be used in your Paintbrush lab!

Building lab stages

The next step in constructing the Paintbrush lab is to author its lab stages. Select **Lab Stages** from the main tutor menu, and from the screen that appears, select **Create a new lab stage**.



WLab @ Kingston University

Create/Edit a lab stage

Stage Name	Paintbrush Demo Stage 1
Lab Name	(none)
Course Name	(none)

Available Virtual Machines

Virtual Machine Name	Owning tutor	Currently used in Lab Stage...
Pb demo - s2	Dave Livingstone	(none)
Pb demo - s1	Initial System Administrator	Paintbrush Demo Stage 1
Pb demo - s3	Dave Livingstone	(none)

Show only my VMs | Show only unallocated VMs | Show all VMs

Base VM for this stage: Pb demo - s1 [set]
End point VM for this stage: (not set) [set] [unset]
Select a VM from the list, and then click **Set** to set these options. Or, if none of the VMs are suitable, click here to create a new one..

Resources (enter URLs, separated with commas)
resource1example.htm

[Save changes] [Cancel]

Figure 77

First, given the new stage a name – as was the case previously, this should be descriptive enough so that you can identify the lab stage when it appears among potentially many others like it in a list.

Then, select the virtual machine you created for lab stage 1 (as you will naturally have given your VMs a descriptive name, you should easily be able to determine which VM belongs to which stage!). Click the correct VM in the table underneath the text *Available Virtual Machines*; it will be highlighted in yellow. Then, click **set** next to the field *Starting point VM for this stage* to allocate the VM to the stage.

The resources field expects you to enter web URLs; the web page that these URLs will be displayed to the student when they undertake the lab in the same fashion as the instructions for tutors were displayed when you were creating your VMs. Any such URL should include the **http://** or **https://** aspect.

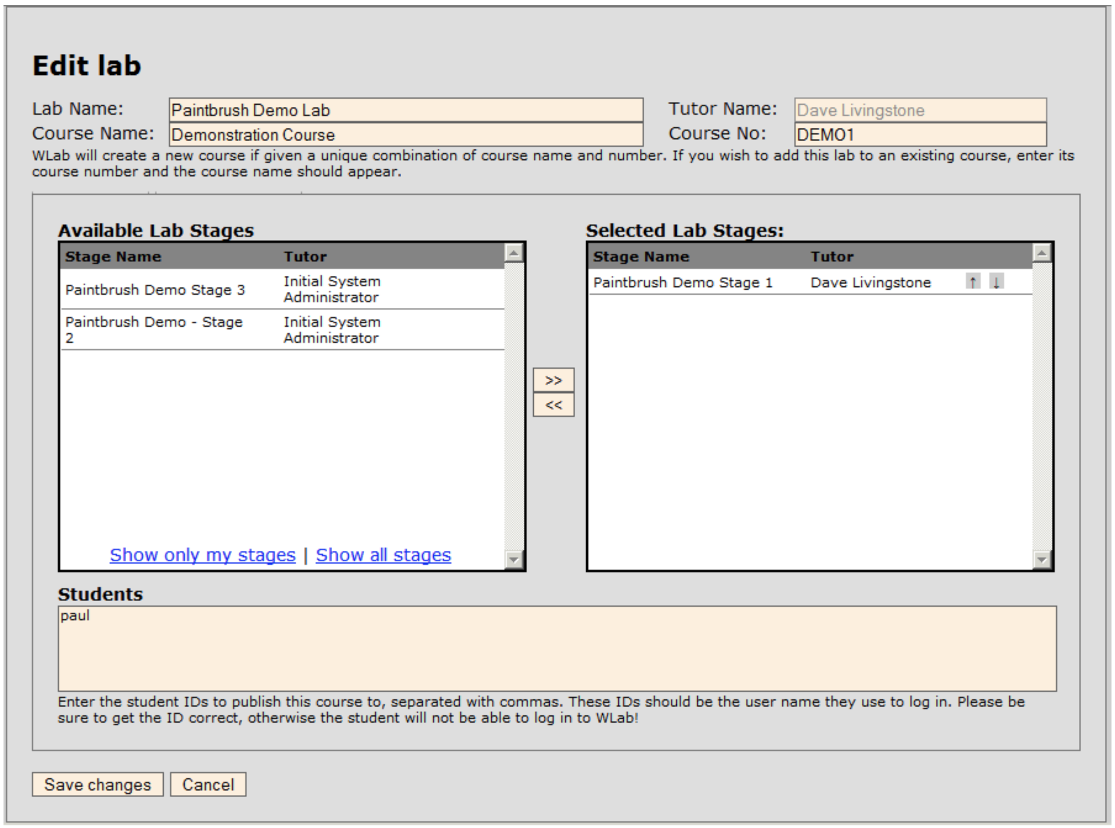
For example purposes, there are three internal resources within WLab that correspond to the Paintbrush example stages; these are *resource1example.htm, resource2example.htm* and *resource3example.htm* where the number corresponds to the stage. To use these internal resources, simply enter them into the field as shown here, *sans* **http://**.

When you have finished entering the details for stage 1 – the screen should look similar to that in Figure 77 – click **Save Changes**.

If you are working through the Paintbrush example, repeat this process for the other two lab stages.

Adding the components together in a lab

Having created all the constituent components for our Paintbrush example lab, we can now bring them together and create the lab itself. From the tutor's main menu, select **Labs**, and **Create a new lab.**



Edit lab

Lab Name:	Paintbrush Demo Lab	Tutor Name:	Dave Livingstone
Course Name:	Demonstration Course	Course No:	DEMO1

WLab will create a new course if given a unique combination of course name and number. If you wish to add this lab to an existing course, enter its course number and the course name should appear.

Available Lab Stages

Stage Name	Tutor
Paintbrush Demo Stage 3	Initial System Administrator
Paintbrush Demo - Stage 2	Initial System Administrator

Show only my stages | Show all stages

Selected Lab Stages:

Stage Name	Tutor
Paintbrush Demo Stage 1	Dave Livingstone

Students

paul

Enter the student IDs to publish this course to, separated with commas. These IDs should be the user name they use to log in. Please be sure to get the ID correct, otherwise the student will not be able to log in to WLab!

Figure 78

As always, give the lab a descriptive name.

You should now enter a course name and number. For the purposes of this example, you can use the example data as per Figure 78. However, when creating your own labs in the future, you should note the following:

- If you enter a combination of course name and number that does not match an existing course on the system, a new course will be created.
- If you enter a course number, and it matches an existing course on the system, it will automatically fill in the course name. So, when creating a new lab, if you want to add it to an existing course it is probably best to leave the course name blank and just enter the course number.

A green or red prompt text will appear underneath the course name advising whether or not a new course is going to be created, so you can check that the behaviour is as you would wish.

Once you have populated the basic lab data, you should select the lab stages you want to include in your lab. Use the << and >> buttons to move lab stages to and from the **Selected Lab Stages** pane – these lab stages are the ones that will be used in the lab. They will be delivered to the student in the order the appear in this pane – so if you want to change this order, use the up and down arrows to move a particular lab stage.

Finally, you need to populate the **Students** field. Here, you should enter the login IDs of the students to whom this lab stage will be published, separated with commas.

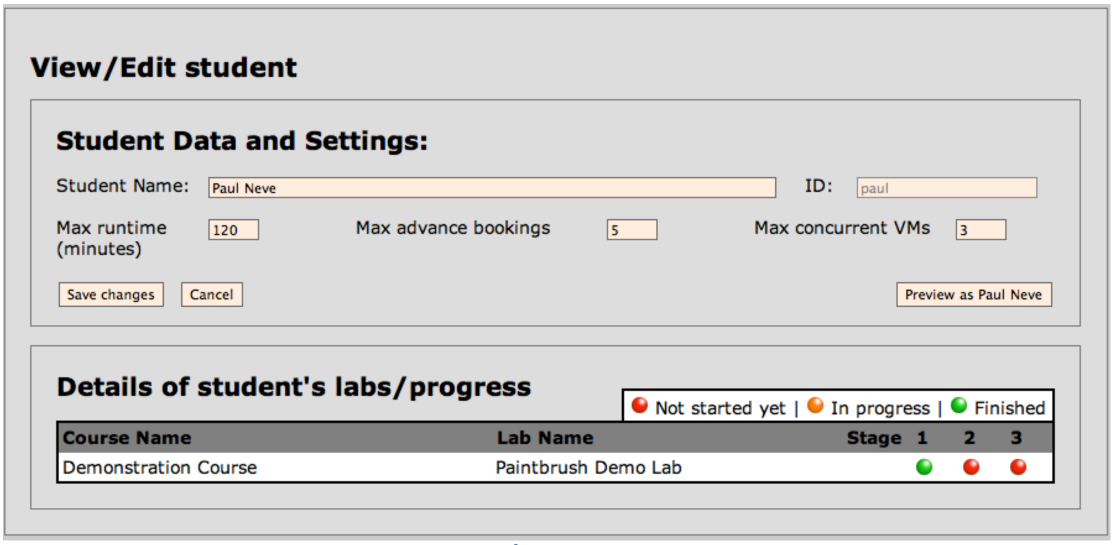
When you have finished creating your lab, click **Save changes**. At this point, the lab is immediately published to the students specified.

Congratulations! You have just authored your first lab in WLab!

Managing and examining student data

A tutor can use WLab without ever performing any student administration. When you add a student login name or ID to a lab, a student object is automatically created on the system and is populated from your institutional directory the first time they log in. There is no direct need for you to modify these student objects; however, you may wish to change certain settings on a per-student basis. Additionally, you might wish to check a particular student's progress. The **Students** option on the tutor main menu provides this functionality.

Having selected a particular student, you will see the following screen:



View/Edit student

Student Data and Settings:

Student Name:	Paul Neve	ID:	paul		
Max runtime (minutes)	120	Max advance bookings	5	Max concurrent VMs	3

Details of student's labs/progress

🔴 Not started yet | 🟠 In progress | 🟢 Finished

Course Name	Lab Name	Stage 1	2	3
Demonstration Course	Paintbrush Demo Lab	🟢	🔴	🔴

Figure 79

The fields displayed are as follows:

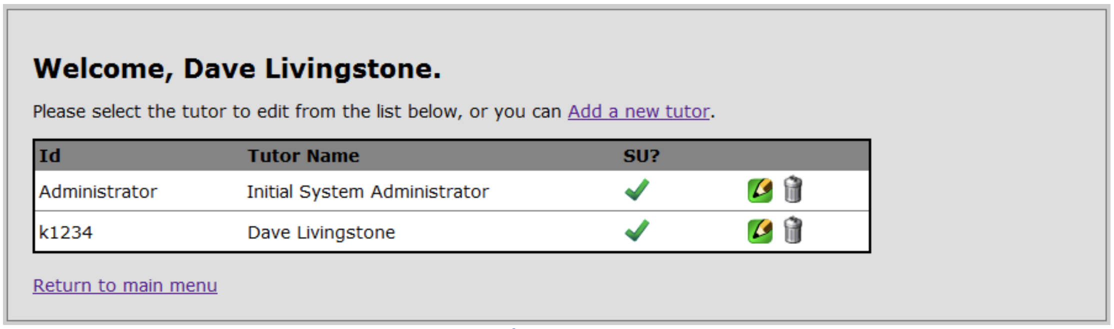
Student Name	The student's name. If the student has yet to log in to WLab, this will be blank. You can modify this name if you wish; such modifications will apply solely to WLab.
ID	The student's login ID. This cannot be changed.
Max runtime	This is a value, in minutes, that represents the maximum amount of time the student can run their lab sessions for. If they reach this maximum amount of time, all their sessions will be suspended. They will not be allowed to resume lab sessions until an interregnum period has elapsed – this is to ensure that students cannot "hog" the system.
Max advance bookings	So that students have some control over when they can work, a booking system is available to them. During a booking, they are allocated a guaranteed slot in which they can work on their labs, and their max runtime value will not apply for the duration of the booking. Each booking lasts an hour. The value in this field specifies how many such advance bookings the student may place.
Max concurrent VMs	This value specifies how many virtual machines – essentially, how many labs – a student can have running at the same time.

At the bottom of the screen, all labs that have been published to this student are displayed, along with a colour-coded indicator that shows how far they have progressed in each lab.

Student progress data can also be found in the lab editing screen – if you edit an existing lab, then select the **Student Details** tab, you will see progress indicators for all of the students to whom that particular lab has been published.

Managing Tutors

If you are a superuser on the WLab system, you can also add new tutors to the system, or edit exiting ones. Select the **Tutors** option from the main menu:



Welcome, Dave Livingstone.

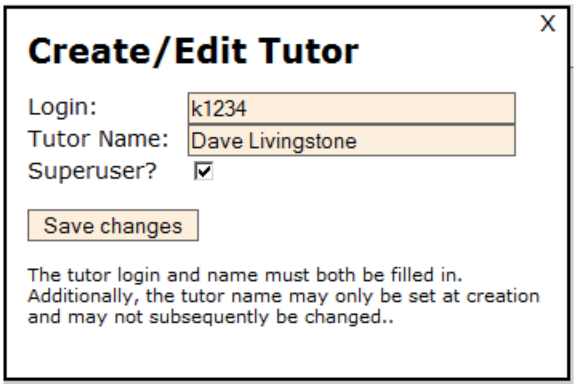
Please select the tutor to edit from the list below, or you can Add a new tutor.

Id	Tutor Name	SU?	
Administrator	Initial System Administrator	✓	
k1234	Dave Livingstone	✓	

Return to main menu

Figure 80

Clicking on the pencil icon, or selecting *Add a new tutor* will display a dialogue box like the following:



Create/Edit Tutor

Login:	k1234
Tutor Name:	Dave Livingstone
Superuser?	✓

The tutor login and name must both be filled in. Additionally, the tutor name may only be set at creation and may not subsequently be changed..

Figure 81

If creating a new tutor, you will need to supply a login ID (this should be the username they use to log into your institutional IT services - either they or your System Administrator should be able to tell you what this is). Once the login ID has been entered and the tutor saved, it cannot be susbequently modified, so be sure to enter it accurately.

The tutor's name can be anything you like and will apply to WLab only - it does not have to exactly match the format of their name on any other system.

Finally, the **Superuser** checkbox indicates whether or not the tutor will have the ability to create and edit tutors themselves. A superuser also has the ability to edit *all* labs and lab stages on the system.

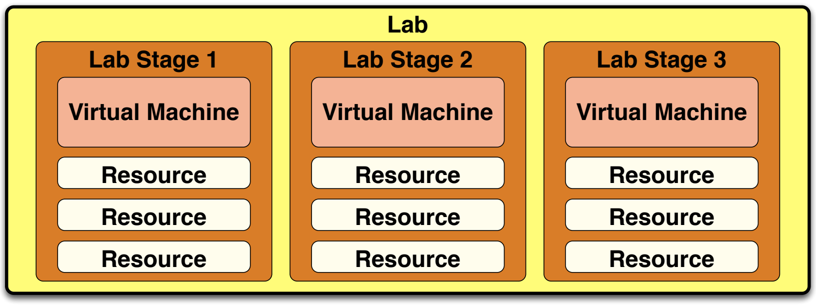
Click Save Changes when you have entered the tutor's details to your satisfaction.

Tutors' Quick Reference Guide

The following is intended to serve as a quick reference for tutors who are already familiar with the WLab system, to quickly remind them of certain important aspects of what has previously been covered. It is a "quick reference guide" not a "quick start guide"! If you have not previously worked through the other material in the tutors' documentation, or otherwise are not familiar with the differences between *labs, lab stages, backend VMs and tutor VMs*, please go back and work through the previous material.

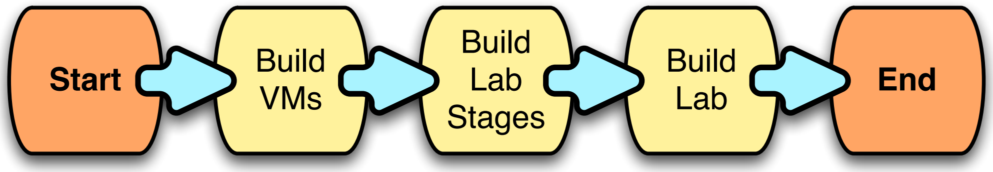
The structure of a WLab lab

A **lab** is divided into **lab stages**, which in turn contain **virtual machines** and **resources**:



Use this diagram as a basis to plan your labs prior to sitting in from of the system itself.

Suggested component authoring order



Suggested order of activities during VM authoring

1. Make any configuration changes needed on the base operating system.
2. Install any additional software required for the lab activities.
   - In most cases you will only need to do steps 1 & 2 once, for the first lab stage. If you then make a point of using previous stage VMs as the basis for your subsequent VMs, these subsequent VMs will already have the additional software installed.
3. Install any files, data or other non-software components required for the current lab stage
   - Depending on how big these are, it may be easier to simply author them from scratch within the VM. Alternatively, you will need to upload them to somewhere that is accessible from within the VM console (e.g. an FTP server, some webspace or a network drive).
4. Configure the VM so that it automatically loads any required software/documents/etc immediately after boot
   - On a Windows VM, this will usually involve adding options to the Startup folder in the Start Menu.
5. **CRUCIAL: MAKE THE VM SAFE, FOLLOWING THE INSTRUCTIONS GIVEN TO YOU BY YOUR SYSTEM ADMINISTRATOR! NEVER FORGET THIS!**
6. Shut down the VM using its console via the standard functions (e.g. on a Windows VM, go to the Start menu and select Shut Down).

Other tips

- This guide has assumed that VMs that are being authored are explicitly intended to be used in a lab stage. Nothing says this has to be the case. If you as a tutor will regularly use a specific suite of software across many courses, it makes sense for you to prepare a "foundation" VM that you will always use as your starting point.

  Consider the example of a tutor who teaches several Java programming courses, e.g. "Introduction to Java Programming", "Web Development in Java", "Desktop Application Development in Java". It would make sense for this tutor to take time right at the beginning and author a VM that includes his preferred IDE, a known-good version of the JDK, any common libraries he/she uses, etc. This foundation VM could then be cloned and used as the basis of stage 1 for labs for three Java courses. Ongoing stages of the labs would clone the previous stage's VM as normal.

- When authoring VMs, the snapshot functions can be extremely useful. Clicking on **Take snapshot** will copy the current state of the VM; **revert to snapshot** will drop any changes made since the snapshot. Snapshotting regularly means if you make any mistakes during VM configuration that are difficult to unravel, you can simply revert to a previously known-good state.

- The documentation advises that one should create all the VMs, then all the lab stages, then the lab itself. This is the process that was followed in the example lab. However, when it comes to creating your own content it may not be practical to build the entire lab in a single sitting. However, at the very least you should follow the process for the first lab stage – so build the VM for lab stage 1, then build lab stage 1 itself and attach the VM, and then build the lab and insert lab stage 1 into it. This one-stage lab can then be published to your students, and you can subsequently go back and add additional VMs and lab stages as and when you desire. The important thing is that you should create at least one VM, then at least one lab stage before creating the lab.

- The prevailing assumption is that the completed state of (for example) stage 1 of a lab is also the starting point of stage 2. This may not always be the case, and in such cases, there is a field on the lab stage editing screen *End point VM for this stage*. This allows you to set an additional VM against the stage to represent an exemplar "completed state". This is never seen by the student, but serves as a reference to you and other tutors as to the "correct answer" for the lab stage. You may well choose to use this as the basis of your clone when authoring the subsequent stage's VM.

  It is not mandatory, but good practice is always setting an end-point VM on the final stage of a lab, as there is no subsequent stage start point to serve as the "answer" to the lab stage.

- If you want to preview a lab from the perspective of a student, there is an option when you view/edit a student, **Preview as…** Selecting this option will essentially make you assume the identity (in a WLab context!) of the student, which means you can view the lab exactly as they would see it. In this mode, the student progress states are *not* set; however, any other changes you make in lab stage VMs are permanent and will be seen by the student. Before making any changes you should make a snapshot so that you can easily revert back to the state you found the lab stage in.

Documentation for System Administrators

Introduction

WLab is a web application that provides a virtual lab environment for any workshop activity that requires the use of a computer. WLab makes use of virtual machines to deliver an IT environment in which a student can perform the tasks involved in the lab exercise, regardless of the local computer they are using.

WLab allows tutors to design and publish lab exercises that are divided into stages; each stage contains a complete virtual machine, along with – if required –appropriate static learning material that complements the activities of the lab. Students access these labs through any standard web browser.

Assumptions

This documentation is intended for those who will have the responsibility of installing and configuring the WLab software, and who will then be responsible for providing support to tutors. The following is assumed on the part of the system administrator:

- They are familiar with the concept of virtualisation and virtual machines
- They have experience of managing a virtualisation solution at the backend, such as VMWare or Hyper-V
- They have experience of managing web server solutions, and know how to perform basic administration tasks (e.g. stop, start) on a web server.
- They understand the implications of making changes to an institutional firewall, and are either capable of making such changes themselves or arranging for someone else to make them
- They are comfortable working at a command line level and editing configuration files.

This documentation will also be useful to developers who wish to make changes to the WLab software, as they will need to know how to configure a new installation.

It is very important that any potential system administrator of a WLab system reads, understands and is aware of the implications raised by this document, and by the manner in which WLab operates. This will prevent inevitable support woes further down the line!

How WLab operates

From a system administrator's point of view, the most important thing to know is that WLab will dynamically create clones of virtual machines as and when they are required. When a tutor designs a lab exercise, they will make a clone of an existing virtual machine, and then modify it to suit the needs of the lab. This modified clone is then referred to as a *tutor VM*.

When a student connects to a lab, WLab will automatically create a copy of any tutor VMs the lab may contain, exclusively for that student's use. This means that, depending on class sizes, WLab may create dozens or even hundreds of virtual machines for a given lab exercise. However, these VMs use differential disks, so should not take up great amounts of disk space. More discussion of such considerations takes place in *System Requirements*.

Responsibilities of a WLab system administrator

Above and beyond the normal requirements for administering a web-based application, administering a WLab system comes with a number of other responsibilities.

- For the WLab application to be useful there must be at least one backend VM available to tutors to clone in the very first instance – otherwise they will not be able to author their tutor VMs. The system administrator will need to author useful VMs, and configure them accordingly. This is discussed in more detail later.
- If one were to clone a virtual machine several times, and then start up all the clones simultaneously issues would almost certainly arise because they would have identical network configurations. On Windows VMs in particular, running several machines with the same computer name can mean that none of them will be able to access the network properly.

  Consequently, the last activity a tutor undertakes when finalising a tutor VM is to make sure that, on the next reboot, the Windows computer name is made unique. In the tutor documentation this is referred to "making the VM safe" and tutors will expect you to provide instructions on how they can achieve this. The procedure will vary, depending on the approach you choose. Further discussion on this subject and exemplar instructions "making VMs safe" for tutors can be found in *Dealing with WLab Virtual Machines*.

System requirements

At the current time, WLab requires the following:

- A web server capable of delivering Java servlets and JSP pages. Tomcat and Glassfish have been tested. Tomcat is assumed in this document.
- Access to a Microsoft Hyper-V virtualisation server.
- Access to an LDAP directory service for authentication.

While the application requires *access* to a Hyper-V server (which inherently requires a Windows Server installation), WLab itself can be hosted on any platform, and has been tested on all three of the major operating systems (i.e. Windows, OS X and Linux).

The WLab application must be run on a server that is on the same physical network as the Hyper-V server, with no intermediate routers in between. Ideally, your configuration should have the Hyper-V server and the Tomcat server that runs WLab both allocated static IP addresses. VMs should then obtain IP addresses via DHCP on the same subnet. For example:

Hyper-V server:	**192.168.1.5**
Tomcat/WLab server:	**192.168.1.6**
Jumpgate server:	**192.168.1.7** (this is explained later)
DHCP range:	**192.168.1.10 – 192.168.1.210** (gives room for 200 VMs)

Where possible it is recommended to allocate a dedicated Hyper-V server to WLab, simply because of the number of dynamically created virtual machines involved.

Consider a course with 20 students. The tutor has created 8 lab exercises for this course, each divided into 4 stages. This means that there will be a maximum of **8 x 4 = 32** tutor VMs. These will then be cloned for each student, i.e. **32 x 20 = 640**. Thus the total number of VMs that will be created for this – rather small! – course is **672**.

This number is not as enormous as it may sound in terms of system resources – assuming that the students only run 1 VM at a time, the worst case scenario is that 20 of VMs would run concurrently. The VMs also use differential disks, so the amount of disk space occupied will not be excessive. However, if this number of VMs were created on a production Hyper-V server that also contained other, non-WLab VMs, it would become impossible to administer these non-WLab VMs among the “noise”.

Setup and Configuration

Initial Installation

These instructions assume that you are using Tomcat. If you are using an alternative servlet container, you will need to adapt what follows accordingly – consult the documentation for your servlet container for further details.

1. Download the latest WLab WAR file from the WLab website (http://www.paulneve.com/wlab/WLab.war).
2. Stop your Tomcat server.
3. Place the WAR file into your **webapps** directory.
4. Restart your Tomcat server. This will explode the WAR file and create a directory called **WLab** in your **webapps** directory.
5. Stop your Tomcat server. If you wish, you can now delete the WAR file from **webapps**.

Tailoring the configuration file

In the new **WLab** directory, there should be a **conf** directory. This contains two XML files, **application-properties.xml** and **strings.xml**. The former is the main configuration file, which must be edited to suit your environment.

This file is divided into sections:

- General application parameters
- Hyper-V server parameters
- Data access parameters
- Authentication/directory server parameters
- Auto-suspension parameters

Apart from the latter section, the configuration sections all follow a similar pattern. As an example, the below is the default authentication section:

```
<bean id="authenticationAccess"
class="org.paulneve.wlab.authentication.AuthenticationAccessLdapImpl">
      <property name="ldapServer" value="192.168.1.7"/>
      <property name="ldapPort" value="389"/>
      <property name="adMode" value="true"/>
      <property name="adDomain" value="demoserver.local"/>
</bean>
```

The parts marked in red must NOT be changed. The blue parts show configurable settings – although again, you should not alter these. The areas you can – and indeed, will have to – alter are shown in green.

Compulsory configuration settings

The following settings will almost always need to be changed before WLab will run in your environment. Before proceeding any further, you MUST configure these:

- **General application parameters**
  - **defaultVncPassword**
    When you come to configure the virtual machines for your tutors to clone, you will need to install VNC. You should standardise on a single VNC password across all WLab VMs, and this parameter should contain this standard password.
  - **adminLogin**
    This should be set to the login name of a user on your directory/domain who will have responsibility for administrating the WLab system. This user can then log into WLab and create additional tutors. If they wish, they can specify that such additional tutors are themselves superusers (which means they too can create tutors).

- **maxConcurrentVMs**
  This value indicates how many VMs can be run at the same time before WLab will prevent students from starting any more. In most cases, the default value of 5 will be too low and should be changed.

  The value comes directly from the virtualisation backend, and thus includes any VMs that may have been started manually i.e. outside of the WLab system. If, for example, there will always be 5 VMs running on your virtualisation server regardless of anything WLab might start, you should take such “permanent” VMs into account.

- **Hyper-V server parameters**

  - **username / password**
    This should be set to a user who has administrative access on the Hyper-V server, i.e. can stop/start/create/delete VMs. This MUST be in the form *domain\username***.** The *password* parameter should be set to this user’s password.

  - **serverName**
    This should be set to the IP address or domain name of the Hyper-V server.

  - **diskPath**
    This should be directory where the Hyper-V server stores its virtual hard disks. Note that this is relative to the Hyper-V server! If you the configuration of Hyper-V is unchanged from the default there is no need to change this.

  - **octet1 / octet2 / octet3**
    These three values should be set according to the IP subnet where the virtual machines will be found. If your configuration was as the example IP addresses given in *System Requirements*, the default values of 192, 168 and 1 would be appropriate. When the system attempts to locate a VM, it will search the entire class C range for it.

- **Data access parameters**

  - **dataAccess**
    This should be set to a fully-enumerated directory where WLab can store its data files. An example might be */usr/share/WLabDataFiles*. This should be read/writeable by the Tomcat process. Note that this is relative to the WLab server. If the directory does not exist WLab will attempt to create it. If in this event WLab cannot create it, an error will occur.

- **Authentication access parameters**

  - **ldapServer / ldapPort**
    Set the IP or domain address and port number of the LDAP directory server here. Note that this does not have to be on the same subnet as the VMs and other servers used by WLab as long as the Tomcat server can establish a connection to this address.

  - **ldapConnectionString**
    Set the LDAP connection string here. This will be used to perform a directory search on the LDAP server, so as to ascertain whether or not the user name entered is a valid user. It must contain the placeholder *USER* - the user name entered will be inserted into the connection string at this point. An example value for this setting might be:

    ```
    uid=*USER*,cn=users,DC=myuniversity,DC=ac,DC=uk
    ```

    This setting is ignored if you are using Microsoft Active Directory, and *adMode* is set to true (see below).

- **ssl**
  Set this value to true if the server/port combination you are connecting to uses SSL. If your LDAP server uses a self-signed certificate, you will also need to follow these instructions:

  1. Download the InstallCert.java program, available from http://blogs.sun.com/andreas/resource/InstallCert.java. Compile it with `javac InstallCert.java`.
  2. Your Java keystore is usually found in the `/lib/security/` directory of your JRE, and is called `cacerts`. You might want to make a backup before making changes to it.
  3. Run the command `java InstallCert [server name]:[LDAP SSL port] [keystore location]`. Alternatively, if you omit the keystore location parameter, it will create a file in the current directory called `jssecacerts`. You might prefer to do this, and then manually move/rename the file produced into the correct location.

- **adMode**
  Set this value to true if you are connecting to Microsoft Active Directory. If this is true, *ldapConnectionString* is not used.

- **adDomain**
  Set this value to the Active Directory domain name. This should be fully specified, e.g. *myuniversity.ac.uk* rather than just *myuniversity*. Ignored if *adMode* is false.

Optional configuration settings

The following settings do not have to be altered for WLab to run, but you may wish to tailor them to suit your needs and environment:

- **General application parameters**

  - **port**
    If you will run VNC on non-standard ports within your virtual machines, you can specify an alternative port number here.

  - **autoSuspendPeriod**
    If a student overruns their allocated runtime, their VMs will be suspended and they are prevented from starting new VMs for this period of time (specified in minutes).

  - **defaultStudentConcurrentVMs**
    Specifies how many VMs a student can run at the same time. Note that this can be overridden on a per-student basis by tutors.

  - **defaultStudentMaxBookings**
    Students can book time on the system during which they will be guaranteed an uninterrupted runtime slot. These bookings last one hour. The value in this parameter value specifies how many such bookings a student may make in advance. This can be overridden on a per-student basis by tutors.

  - **defaultStudentMaxRuntime**
    This specifies how long a student may run their VMs for before the student will automatically suspend them (so as to prevent a student from “hogging” the system). The value is in minutes, and can be overridden by tutors on a per-student basis.

- **Auto suspension parameter**
  This configuration section is formatted slightly differently to the others:

```
<bean id="vmSuspender" class="org.paulneve.wlab.utilities.VmSuspender"
autowire="byName">
<constructor-arg type="int"><value>10</value></constructor-arg>
</bean>
```

Only the number between the <value> tags shown in green should be altered. The value specifies in seconds how often the system will poll for overrunning virtual machines. Increasingly this value reduces server load, but might result in less accuracy. You should increase the value proportionally as the number of likely concurrent VMs increases – a rule of thumb is to make this value double the maximum number of concurrent VMs.

Note that any changes to *application-properties.xml* require a restart of Tomcat before they take effect. We also recommend you back up your *application-properties.xml*. When upgrading to a new version of WLab, your version of the file will be replaced with the default. If you have no backup, you will have to redo all your changes.

Additional configuration for Windows Server 2008 R2

Windows Server 2008 R2 introduced additional security features which require additional configuration before WLab can communicate with the Hyper-V server:

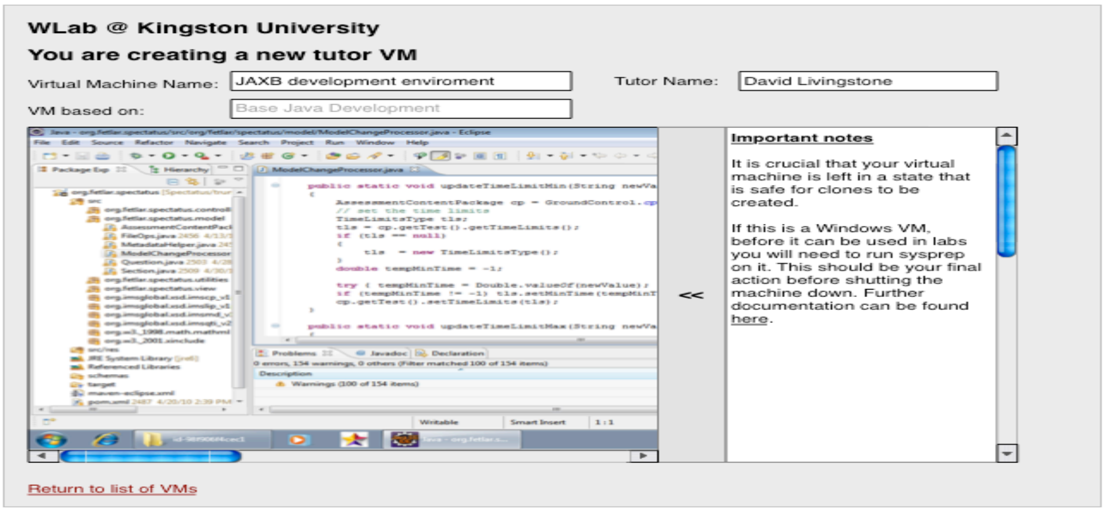
1. Make sure that **Windows Firewall with Advanced Security** has the rules **for COM+ Network Access** and **COM+ Remote Administration** enabled. They will be disabled by default.
2. You will need to change the permissions on the registry key `HKEY_CLASSES_ROOT\CLSID\{76A64158-CB41-11D1-8B02-00600806D9B6}`. First, change the owner of the registry key to your own user login; this will allow you to make changes to the permission of the key. Then, modify the read/write permissions on the key, and ensure that the user that your Tomcat server runs as has **Full Control**.

Adding tutors to the system

With the configuration file modified as appropriate, you should now be able to log in to the system and add tutors. WLab will be published at the URL

[your Tomcat server base address]/WLab/wlab.html

Open a browser and go to this URL. You should be able to log in as the user who was set in the *adminLogin* parameter. Once successfully logged in, you should see the following screen:



WLab @ Kingston University

You are creating a new tutor VM

Virtual Machine Name: JAXB development enviroment

Tutor Name: David Livingstone

VM based on: Base Java Development

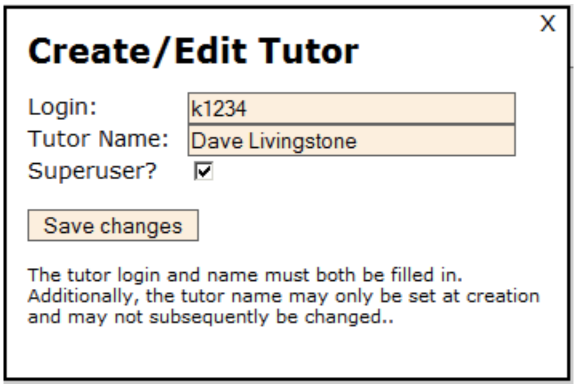
Important notes

It is crucial that your virtual machine is left in a state that is safe for clones to be created.

If this is a Windows VM, before it can be used in labs you will need to run sysprep on it. This should be your final action before shutting the machine down. Further documentation can be found here.

Figure 82

Click the option **Tutors**. You will be presented with a list of the tutors currently in the system – there should only be the *Initial System Administrator* shown. Here, you can **add a new tutor**; click this option to bring up the dialogue box below:



Create/Edit Tutor

Login: k1234

Tutor Name: Dave Livingstone

Superuser? ☑

The tutor login and name must both be filled in. Additionally, the tutor name may only be set at creation and may not subsequently be changed..

Figure 83

The login ID cannot be modified once a tutor has been entered, but the tutor name and their superuser status is changeable. Making a tutor a superuser will also give them the ability to create and delete new tutors.

Existing tutors can be modified and deleted from the tutor list, by clicking the pencil and trashcan icons respectively.

Dealing with WLab virtual machines

As noted previously, when creating a tutor VM tutors will need at least one VM to precede theirs, as their new VM must be cloned from an existing one. If no VMs pre-exist on the virtualisation backend, then WLab will not be usable because tutors will not be able to create VMs for labs.

The tutor documentation assumes that a VM exists that contains a basic installation of Windows XP, and nothing else. The example they will work through takes them through authoring a WLab lab exercise using this XP VM as a starting point. Therefore, this XP VM is the very least tutors will need to get started. Realistically, if you are serious about using WLab at your institution, you will probably need to create several VMs depending on the courses WLab will be used for, and the needs of the tutors.

We recommend that you liaise with your tutors at an early point of your WLab endeavours, and take the time up front to set up a suite of “foundation” VMs that the tutors can then build upon. For example, if your tutors are likely to be teaching Java, you might set up a VM that contains the various IDEs your institution uses and a known-good instance of the JDK. If you did not, and instead expected tutors to take the bare bones XP installation, then by themselves install the JDK, then install and set up Eclipse or NetBeans, almost certainly they would run into difficulties. By not making the early effort you will only be creating more support calls for yourself in the future.

You can use the usual Hyper-V management tools to build these VMs. For the most part, VMs that you intend for use by tutors in WLab are no different to any other Hyper-V VM you might create. However, there are several additional factors you will need to consider:

VNC installation

The TightVNC Java applet is used to embed a VM console within WLab-generated web pages. Consequently, all VMs for use with WLab must have a server installed and set to start on boot. **If you do not install such a VNC server, the tutor will not be able to connect to the virtual machines through the WLab web interface.**

For Windows VMs, it makes most sense to use TightVNC and thus match the VNC flavour being used by the client. TightVNC server – available from http://www.tightvnc.com/ - should be installed within the VM as a service, and the default VNC password for each VM should match that set on the **defaultVncPassword** parameter in **application-settings.xml**. Similarly, if you decide to change the VNC port number, you will also need to update the **port** parameter. **These settings MUST be consistent across every VM you intend for use in WLab.**

A common mistake is to leave the Windows Firewall on inside your VMs. **At the very least, in the firewall settings within the VM be sure to permit traffic on the VNC port you intend to use** (usually 5900 if you have not changed anything). In many cases it may be simpler to disable the Windows Firewall on the backend VMs your tutors will use.

On Linux VMs, there are a multitude of VNC server scenarios, and it is assumed that a system administrator who is au fait with Linux would be able to set up an appropriate solution.

“Making VMs safe”

During its lifetime, a tutor VM will be cloned many times – once for each student. In the majority of cases, these clones will be used concurrently (unless it is a one student class!). If from a network perspective the clones are not unique then the students will experience connectivity problems. As noted previously, when a tutor has finished designing their VM, their last act should be to ensure that, upon the first boot of any clone, the clone makes itself unique.

If you are using DHCP, this should not be an issue in terms of IP addressing. However, on Windows workstations there is also the computer name to consider – computer names must be unique. Some kind of script is therefore required that will change the computer name of a newly cloned VM to a unique (random) value. In the tutor documentation this is referred to as "making the VM safe" and it is stressed upon them that this must always be the last task they perform in a tutor VM before they attempt to publish it in a lab stage.

TUTORS WILL EXPECT YOU TO PROVIDE THEM WITH DOCUMENTATION AS TO HOW THEY SHOULD PERFORM THIS TASK.

The recommended Microsoft method of achieving this is by using Sysprep; however, this takes far too long to complete for it to be usable in WLab. When a student accesses the next stage of a lab exercise it is not acceptable to subject them to a 10 minute delay while Sysprep executes.

The best results have been achieved using Dave Clarke's *wsname* script, which can be found at the URL http://mystuff.clarke.co.nz/MyStuff/wsname.asp. The following is one method of using wsname to provide a facility for making a VM safe:

1. Create a folder **c:\wsname** on the VM.
2. Place the **wsname.exe** file into this folder.
3. Create a file called **rename.old** in this folder – this file should contain the following:

```
@echo off
c:\wsname\wsname /n:WLAB-$RANDOM[10]
shutdown –s –f –t 1
del c:\wsname\rename.bat
```

4. Open the Group Policy Editor (select **Run** from the Start Menu, and type **gpedit.msc**)
5. Select **Local Computer Policy / Computer Configuration / Windows Settings / Scripts**. Double click the **Startup** option in the right hand pane.
6. Click **Add**, then type **c:\wsname\rename.bat**. (Note the suffix, .bat and NOT .old as per the file created in step 3)

This configuration means that all a tutor needs to do to make a VM safe is to copy **c:\wsname\rename.old** to **c:\wsname\rename.bat** as their last action before they close the VM down. Any clones that are made after this point will start up, find the script in **c:\wsname** and execute the script.

A sample set of instructions for tutors to use this script might be as follows:

Tutor addendum – Making your tutor VMs safe

It is very important that, when you have finished configuring a Tutor VM, you make it safe for use by a student.

To do so, follow these steps:

1. Click on the **Start** menu at the bottom left of the screen.
2. Click **Run**.
3. Type **cmd** and press Return.
4. Type **cd \wsname** and press Return.
5. Type **copy rename.old rename.bat** and press Return. Note there are no spaces between the periods (.) and the letters around them. If successful, you should see the text **1 file(s) copied**.
6. Type **exit** and press Return.

Your tutor VM has now been made safe. You should now immediately shut it down, and it can now be used in a lab stage.

Note that a student will be provided with the same environment as the tutor. You might be tempted to create a batch file that performs these tasks, place it on the VM desktop, and tell tutors "to make your VMs safe, click the icon on the desktop". However, such a desktop icon would also be visible by students when they run their lab exercises. Some degree of obfuscation is therefore desirable to avoid support issues originating from the student body.

Here is a detailed breakdown of the steps in the script:

```
@echo off
```
Suppresses output to the console.

```
c:\wsname\wsname /n:WLAB-$RANDOM[10]
```
Uses **wsname** to change the computer name to WLAB-[a 10 character random string]

```
shutdown –s –f –t 1
```
Issues the command to shut down the virtual machine in one second's time – changes to the computer name only take effect after a reboot.

```
del c:\wsname\rename.bat
```
Delete the script so it is not executed on subsequent reboots.

If you do not wish to use this technique, you should note that WLab assumes that when a tutor VM is cloned in order to create a new VM for a student, the following will occur:

- The VM will boot itself normally,
- It will change its computer name to a random value,
- **It will shut itself down. Note that this is a complete shut down, i.e. halt – not a restart.**
- WLab then sends a restart request to the VM.
- The VM boots with the new computer name settings,
- The student is connected to the VM and they can resume their lab.

The point in bold is crucial, and you need to make sure that your alternative method of changing the computer name shuts the VM down completely at this point, rather than just issuing a restart. Also, if you use an alternative method of "making safe" bear in mind you will also need to provide the tutors with documentation for it. The tutor documentation explicitly mentions that you will provide them with such instructions, and advises them not to attempt to use the system if they do not have them, or do not understand them.

There may be instances where the "making safe" procedure is not necessary. If a tutor is designing a VM that has no need for Microsoft networking connectivity you could remove the Microsoft networking components (you would still need TCP/IP in order to establish a VNC connection), and there would be no need for the "making safe" procedure. Similarly, this would probably not be required on most Linux configurations (assuming that DHCP was being used and SAMBA was not). If, for some reason, a VM does not require the "making safe" procedure, the tutor should be advised to change the default setting for **Boot cycles after cloning** from 2 to 1 when they configure their VMs. Details of where to find this setting are in their documentation.

VM suffixes

When WLab creates a new virtual machine, it will use one of two suffixes:

BASE-	a tutor-authored VM, i.e. a *tutor VM*
DYNAMIC-	created by the system for use by a student, cloned from a tutor VM

Do not delete any such virtual machines via Hyper-V's own management tools. This may cause errors in WLab.

Configuring access from outside your institution

When a student connects to a lab session, and particularly a VM within that lab session, WLab determines the VM's IP address and then builds a web page with the VNC client application embedded. Part of this web page is a parameter to the VNC applet which specifies the IP address of the VM, so as a VNC connection can be made to it.

This is fine if the student's client computer can "reach" this IP address, but in many cases – and with certainty if they are running their lab from outside your institution – there will be no direct route to the address. This is illustrated in Figure 84.

Also shown in Figure 84 is the traditional method of creating a route through a firewall so that external clients can access an internal web server. One can thus easily arrange external access to the web application aspect of WLab. However, one cannot do the same for the dozens of VMs that WLab entails. Aside from the security implications, these addresses are not consistent and will frequently change.

In order to provide a route through institutional firewalls to WLab VMs, we have made use of an adapted instance of the Jumpgate project (http://jumpgate.sourceforge.net). Jumpgate runs on a server with a static IP address behind the firewall. A single external port is routed to a specified internal port on this server. Jumpgate then dynamically routes traffic through to the appropriate WLab VM. This is illustrated in Figure 85.

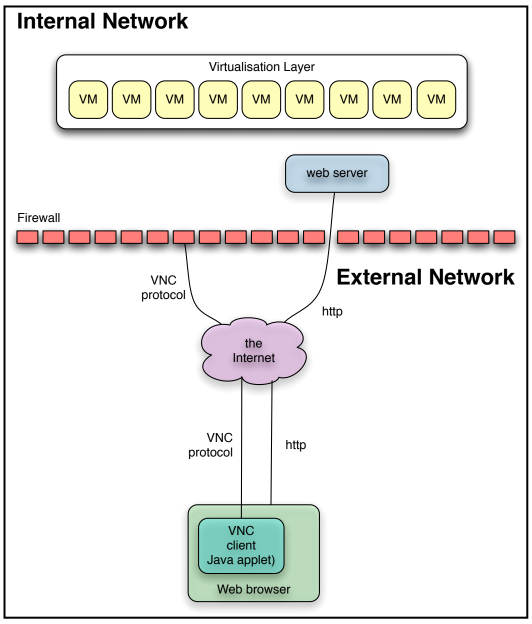


Figure 84

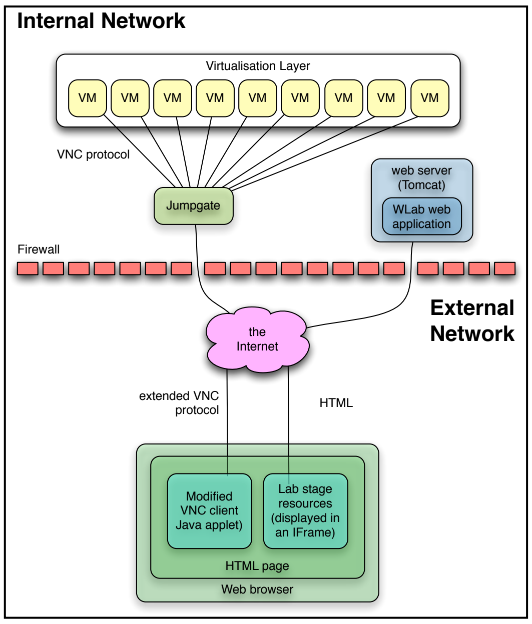


Figure 85

How to use Jumpgate with WLab

Download the appropriate version of Jumpgate for your platform from the WLab website.

Windows:	http://www.paulneve.com/wlab/jumpgate-windows.zip
OS X:	http://www.paulneve.com/wlab/jumpgate-mac.zip

On OSX, the zip file will contain a single executable simply called **jumpgate**. On Windows, you will see **jumpgate.exe** and **cygwin1.dll** – the DLL file must be kept in the same directory as the EXE file.

Unzip the zip file in a convenient location on the server where Jumpgate will be located.

On Linux, because of the many differences between distributions, you should build Jumpgate from source. Follow these instructions to do so:

- In a command line or terminal, create a work directory (e.g. `JumpgateSource` and `cd` to it.
- Type `svn co svn://wlab.paulneve.com/Jumpgate .` - be sure to include the period . at the end of that command!
- Type `make generic`
- You should end up with an executable file called `jumpgate` in the directory.

You can now start Jumpgate with the command

```
jumpgate –l [port number] –i –p [password]
```

or on Windows:

```
jumpgate.exe –l [port number] –i –p [password]
```

The port number can be any number that is not being used by other network services on the server running Jumpgate. Higher (four digit) values are recommended to avoid unexpected "collisions" with services you may not have taken into account. The password can be whatever you like and pertains to Jumpgate only. It is to ensure that if anyone were to telnet directly to your Jumpgate server, who had read the source code and knew what response it was expecting, could not then connect to other machines on your internal network.

You should now configure an external IP address and port on your firewall. This will need to route traffic received through to the internal IP address of the Jumpgate server on the port you specified in the Jumpgate command line.

Finally, you need to configure WLab's **application-properties.xml** as follows:

jumpgate	should be set to **true**
jumpgatePort	should contain the *external* firewall port number
jumpgateServer	should contain the *external* server domain name or IP address
jumpgatePassword	should contain the password set in the Jumpgate command line

It should be emphasised again that the port and server settings in the configuration file should correspond to the external address configured on your firewall, not the internal address of the Jumpgate server.

Note that Jumpgate should stay resident in memory. However, it does not specifically have a shutdown parameter and will continue to run unless terminated using whatever means your operating system provides (e.g. the **kill** command on a UNIX-like OS, or using **End Task** in Task Manager on Windows).

Modifying WLab using the strings.xml file

On your Tomcat server, in the **webapps/WLab/conf** directory there is a second XML file, **strings.xml**. This contains all of the string (text) values that a user of WLab sees. They can be modified as you see fit, e.g. if you wish to translate the application into a language other than English.

strings.xml is ordered by the WLab page the values correspond to. For example, the page *studentlablist.html* – i.e. the initial screen which greets a student and lists their labs when they log in – has a section in the file, *labconsoleview.html* has a section, and so on.

The ordering of this file is probably organised more with developers in mind than WLab system administrators, so the simplest way for you to find a particular value you wish to change is to locate it within the WLab application itself, and then do a search in **strings.xml** for it.

Each value is contained in an XML element such as the following:

```
<entry key="retMainMenu" value="Return to main menu"/>
```

The red part of the XML element should not be changed – the green part can be modified as you see fit. Above or near each element, you should find a comment that explains its purpose.

Longer values may run across several lines, such as:

```
<entry key="tooManyBookings" value="
Sorry! You have already exceeded your maximum allowed
number of bookings!
"/>
```

In these cases you may split the value across as many lines as you need for easy of reading. However, be very sure that you maintain the final closing **"/>** marker.

For values that contain HTML, the < and > characters have to be replaced with [and], so as to avoid breaking surrounding XML markup. An example is highlighted in blue below:

```
<entry key="studentLabListBookingBlurb" value="
Alternatively, you can book guaranteed time on the system
by using the [a href='BOOKINGPAGE']booking page[/a]. You can
book hour slots of time on the system, either spread out, or
in a consecutive block. During this booked time your labs will
never be shut down.
"/>
```

Some values may also contain placeholders, such that marked above in black. In most cases you *must* retain these placeholders somewhere in the string if you modify it. The comment above the element will explain such requirements. Similarly, any restrictions on elements that contain HTML will also be detailed in the comments.

Documentation for Developers

Introduction

WLab is a web application that provides a virtual lab environment for any workshop activity that requires the use of a computer. WLab makes use of virtual machines to deliver an IT environment in which a student can perform the tasks involved in the lab exercise, regardless of the local computer they are using.

WLab allows tutors to design and publish lab exercises that are divided into stages; each stage contains a complete virtual machine, along with – if required –appropriate static learning material that complements the activities of the lab. Students access these labs through any standard web browser.

Assumptions

This documentation is aimed at developers, i.e. those who intend to either modify or add to the functions of the application by modifying its source code. It is assumed that:

- You are familiar with the Java language and its conventions
- You are familiar with working in an integrated development environment (e.g. Eclipse, NetBeans, etc)
- You are comfortable using revision control and source code repository software such as SVN
- You are comfortable using a command line, and compiling code from such an interface

Developers will also need the *Documentation for System Administrators*, and it is assumed that developers will at least have read that document before they start trying to make use of the information contained in *this* document. At some point, if you intend to run WLab you will need to modify the **application-properties.xml** configuration file as per the directions given in *Documentation for System Administrators*.

Preparing Windows for WLab development – installing Cygwin

Note: if your development environment will be running on a UNIX-like OS such as Linux, OS X, etc, you can skip this section.

Most of the tools used to manage the WLab source code come from a UNIX pedigree. The simplest way to prepare an appropriate development environment under Windows is to install Cygwin (http://www.cygwin.com). Cygwin provides a simulated UNIX environment on Windows, and also includes most of the tools one needs to work on WLab.

Download the latest version of Cygwin from the website and run its setup program. When prompted to select packages, you should ensure the following packages are included. (To select a package, click the text "Skip" and it should change to a version number).

- **gcc** (enter the search string *gcc* and select *Devel / gcc-core*)
- **make** (enter the search string *make* and select *Devel / make* – it should be towards the bottom of the packages listed)
- **Subversion** (enter the search string *subversion* and select *Devel / subversion* – it should be towards the top of the packages listed)

Once you have selected these packages, click **Next** and allow the Cygwin installer to proceed through installation.

When it is complete, it should have installed the program **Cygwin Bash Shell.** Use this instead of the regular Windows command line when working on WLab development. Note that this is a simulated UNIX command line, so some conventions are different. Of most importance the the fact that the directory delimiter is forward slash / rather than backslash \. For more details of working in Cygwin's Bash shell, try the excellent *Bash Guide for Beginners,* available at http://www.tldp.org/LDP/Bash-Beginners-Guide/html/.

Obtaining the WLab source code

The WLab source can be obtained from its Subversion repository at svn://wlab.paulneve.com.

Create a directory that will store the WLab source, navigate to it in a command line[11], and then issue the following SVN command:

```
svn co svn://wlab.paulneve.com/ .
```

This will download a complete set of the WLab code, including the WLab application itself, and the modified versions of TightVNC and Jumpgate needed for its operation. If you only want WLab itself, and have no intention of working on the other components, append `WLab` to the end of this command line – note that this is case sensitive, so capital-W, capital-L, lower case A and B.

Building the WLab WAR file

WLab uses Apache Maven (http://maven.apache.org/) to manage its dependencies and to simplify the build process. If you do not have Maven installed you should install it now.

To build a WAR file for installation on Tomcat or other servlet container, navigate to the WLab directory in a command line, then issue the following commands:

```
mvn clean
mvn compile
mvn jibx:bind
mvn war:war
```

If you receive a build error, check that you are in the correct directory. There should be a file called **pom.xml** present. A common error is when the initial directory created to hold the

[11] We will assume from now on that, on Windows, "a command line" means "the Cygwin Bash shell".

source is itself named WLab – this would result in the directory structure WLab/WLab, and here it would be the second level down where the Maven commands should be issued.

The first time you use Maven for WLab, it may take some time as the required libraries are downloaded. After each step, you should see (among other output) the result **BUILD SUCCESSFUL.** Following a successful build, you can find the resulting WAR file in **WLab/target**.

Building Jumpgate

The build process for Jumpgate uses the more standard **make** command, and is simply a case of navigating into the Jumpgate directory and issuing the command:

```
make generic
```

Once the process is complete, you should find that you have a binary executable called simply **jumpgate** in the current folder (**jumpgate.exe** on Windows). Note that unlike the other components used in WLab which are Java based, Jumpgate is written in C, so the binary will only run on the platform it was originally compiled on.

If you are building a version for Windows, and you intend to run the binary on a machine that does not have Cygwin installed, you will need to include the **cygwin1.dll** file along with **jumpgate.exe**. The DLL file must be present in the same directory as the EXE at runtime.

Building the TightVNC client applet

The VNC client applet also uses the **make** command. Navigate into the **TightVNC-Java-pnmod/src** directory, and in a command line, simply type:

```
make
```

You should end up with a binary in the same directory, **vncviewer-pn.jar**.

For this binary to be useful in a WLab context, it *must* be digitally signed with the **jarsigner** tool. If not, the Java security model restricts any connections from the applet to the web server from which it was delivered. This would mean that the applet would not be able to establish a connection to any WLab virtual machines. The web page http://wiki.plexinfo.net/index.php?title=How_to_sign_JAR_files gives a quick overview of how one can sign a JAR with a test key, for those not familiar with the process.

Javadoc

A Javadoc is available for the application at http://www.paulneve.com/wlab/javadoc.

How to prepare the Eclipse IDE for WLab development

To build an Eclipse project for WLab, follow these steps:

1. You will need to install a number of Eclipse plugins:

Plugin name	Source Repository URL
Maven Integration for Eclipse	http://m2eclipse.sonatype.org/sites/m2e
Maven Integration for WTP	http://m2eclipse.sonatype.org/sites/m2e-extras
JIBX plugin	http://jibx.sourceforge.net/eclipse

   Many Eclipse installations will have the base Maven integration pre-installed, but not the WTP integration. This may cause problems when installing the latter if the versions do not match, and it may be necessary to upgrade the former before proceeding.

2. Create a new Eclipse workspace.

3. Navigate to the directory of the Eclipse workspace you just created in the command line. Check out the WLab source from SVN into this workspace using the command:

   ```
   svn co svn://wlab.paulneve.com/ .
   ```

4. In your new Eclipse workspace, select *Import / Maven / Existing Maven Projects*. Browse to the *WLab* folder which will have been created by the SVN checkout. Click *Finish*.

5. Wait while Eclipse and the Maven plugins set up the project.

6. Enable JIBX by right-clicking on the WLab folder in the project folder, and selecting *JIBX*. An error will appear; this is normal. To fix the error, right-click on the WLab folder, select *Properties,* and in the *JIBX* setting change *JIBX Mappings Folder* to `src/main/java`. Disable and then re-enable JIBX to make the change take effect.

7. "Create" a new server to run the application by clicking on the *Servers* tab, then right-clicking in the white space in the bottom right pane and selecting *New server*. Choose an Apache Tomcat 6.0 server. Browse to your Tomcat install directory (if you don't have Tomcat installed, then you can just download it from Apache, unzip it and point Eclipse at the resulting directory). Click *Finish*.

8. Right-click on the *Tomcat 6.0 Server at localhost* which will have appeared in the bottom right pane, and select *Add and Remove*. Click *WLab* and use the *Add* button to publish it onto the development server.

9. Edit the file in `src/main/webapp/conf/application-properties.xml` to suit your environment. Consult the system administration documentation for details regarding the settings to be found in that file.

10. The application can be started by selecting the server from the appropriate tab, and clicking the green "play" button. The application should be usable at http://[your IP address]:8080/WLab/wlab.html, assuming a standard configuration. It is advised to stop the server before making code changes.

How to prepare the NetBeans IDE for WLab development

1. Select **Open Project** from the file menu and browse to the WLab source folder.

2. That's it! NetBeans will use the Maven configuration to automatically set the development environment up for you.

Overview of the WLab source code

Directory structure and file locations

The directory structure used in the WLab code follows the usual Maven conventions for a Java web application:

WLab/src/main/java	Java classes
WLab/src/main/webapp	Web files – HTML, JSP, etc

Spring configuration

The application uses the Spring framework (http://www.springsource.org/) extensively. The Spring MVC facilities are used in particular; a Java class at the server side processes any input from the user, then builds a Spring *ModelAndView* (MaV) object. The MaV includes a reference to one of the JSP views, and the associated model the view will need in order to display meaningful data. The diagram below illustrates this:

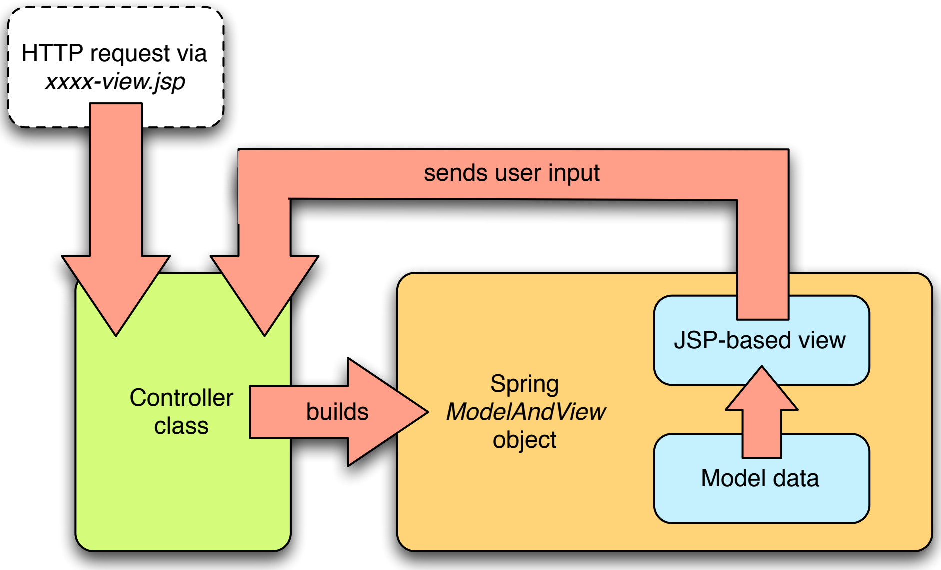


Figure 86

In order to hide the underlying application architecture from the user, Spring is configured to map URLs that end in HTML to the MVC model. The JSP view pages can be found in **webapp/WEB-INF/jsp** (the obscufated location prevents a user entering a URL ending in JSP and accessing the page directly, and bypassing the Spring MVC model).

The Spring configuration is split across two files:

WLab-servlet.xml
conf/application-properties.xml

The former contains the various bean configurations for the controller classes. If you add additional controllers and/or views to WLab, you will most likely need to add a corresponding entry here. The latter file contains the beans that have user-configurable settings. If you create alternative implementations of interfaces such as *DataAccess* or *VirtualisationAccess* (discussed later) this file will need to be updated accordingly.

WLab data model

The WLab data model is implemented by the classes in the org.paulneve.wlab.data package, and is described by the diagram below:

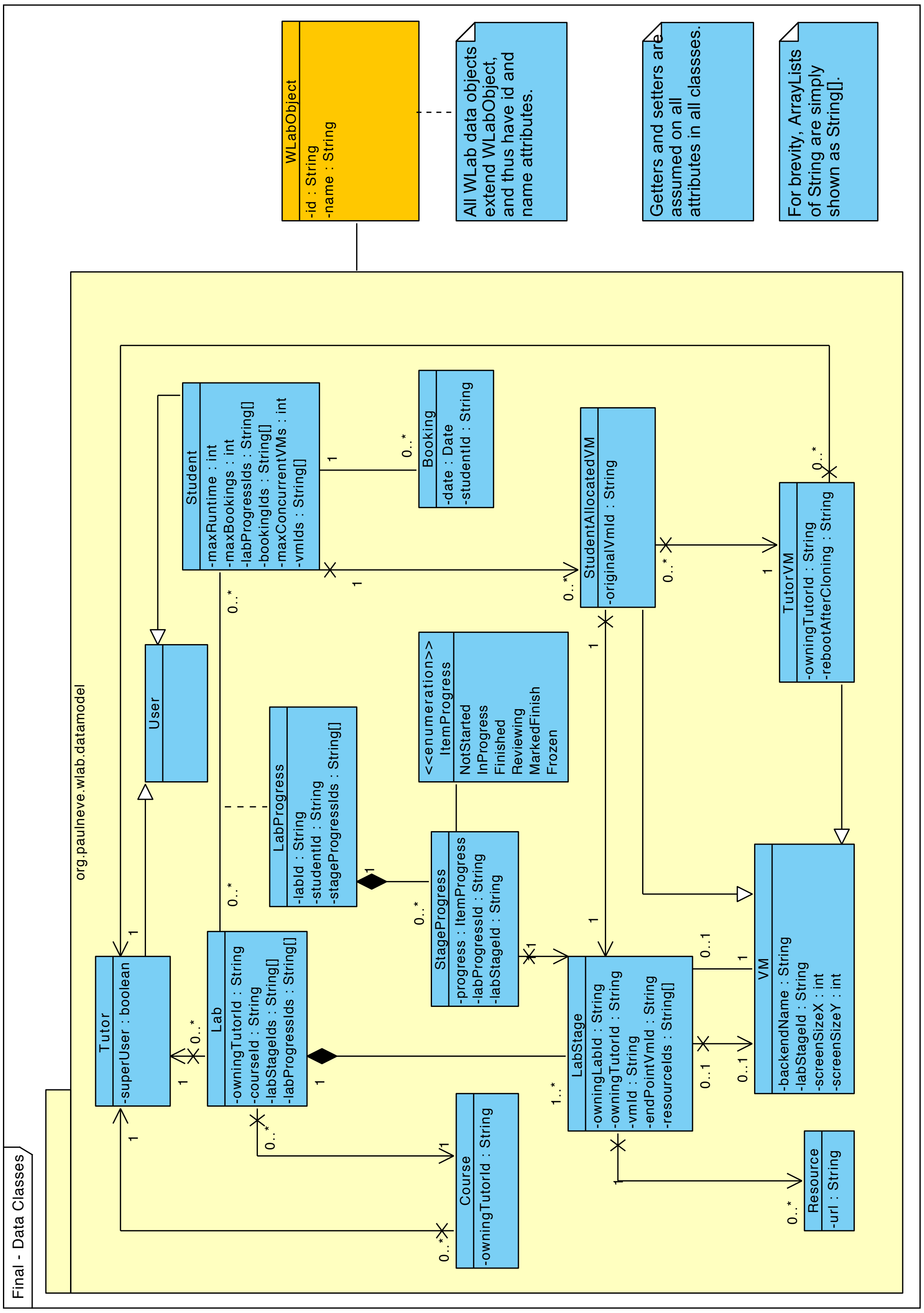


Figure 87

Associations between data objects are maintained by string values that contain the ID of the object being referenced.

Controller class/View JSP dependencies

The corresponding controller classes can be found in the org.paulneve.wlab.webui package, and the table below shows the correlations between the controller classes and the views. For the most part as a general guideline, *view.jsp is dependent on *Controller.java:

Student UI:

JSP view(s) / controller class(es)	Purpose and explanation
labconsoleview.jsp LabConsoleController.java	Handles the student's view of a lab exercise. Given a VM ID, the controller will build the appropriate parameters (e.g. IP address, screen size, etc) for the VNC applet. It also handles commands originating from the student via the JSP view, such as snapshot-related commands. Finally, it handles navigation to other stages of the lab exercise.
bookingview.jsp BookingController.java	The view/controller to handle the session booking screen. Much of the actual functionality of this screen is provided by a combination of Javascript within the JSP page, plus AJAX calls to some of the viewless servlets.
wlab.jsp studentlablist.jsp WebUIController.java	wlab.jsp provides the initial login screen. The controller handles authentication. If unsuccessful, an appropriate message is passed to the view. Once authentication is successful, if the user is a student, the controller then returns a *ModelAndView* object that includes studentlablist.jsp – essentially "switching" to this view. As part of the *MaV* object WebUIController will also supply any details of timeout countdowns, currently running bookings, status of the student's VMs and so on. The view provides the student with commands to start/stop VMs – these however are handled by one of the various viewless servlets (discussed later).
vmconsoleview.jsp VMConsoleController.java	This combination is never used unless you set the **vmOnly** flag in **application-properties.xml. THIS SHOULD NEVER BE DONE.** The configuration flag and these files are retained solely to show the evolution of the application through the iterative development process, for the purpose of an MSc dissertation. Almost certainly, after the completion of this MSc these files will be removed.

Tutor / Management UI:

JSP view(s) / controller class(es)	Purpose and explanation
backendvmlistview.jsp BackendVmListController.java	This combination generates and displays a list of VMs from the virtualisation backend. In the JSP view, a tutor may choose to clone or edit the VMs (this will invoke the viewless servlet CreateNewBaseVm – discussed later).
backendvmmdview.jsp BackendVmMDController.java	Provides the metadata page where a tutor can edit the basic details of a tutor VM. In the UI, this is shown by clicking on the **VM Details** tab of editbasevmview.jsp and is embedded there via an IFRAME.
editbasevmview.jsp EditVmConsoleController.java	The VM console view, used when a tutor is editing a tutor VM. Works similarly to the student's LabConsole view/controller combination, with options to navigate between stages removed.
labeditview.jsp LabEditController.java	Provides facilities for a tutor to edit a lab. Much of the functionality comes via client-side Javascript.
lablistview.jsp LabListController.java	Provides a list of labs so a tutor can select one to edit. Once one is selected, routes to labeditview.jsp. Also provides an option to create a new lab.
labstageeditview.jsp LabStageEditController.java	Provides facilities for a tutor to edit a lab stage.

JSP view(s) / controller class(es)	Purpose and explanation
labstagelistview.jsp LabStageListController.java	Provides a list of lab stages. When one is selected, routes to labstageeditview.jsp. Also provides an option to create a new lab stage.
studenteditview.jsp StudentEditController.java	Provides facilities for a tutor to edit the basic details of a student. Additionally, the controller assembles and the view displays details of the student's progress within the their labs.
studentlistview.jsp StudentListController.java	Provides the list of students the tutor chooses for editing via studenteditview.jsp.
tutoreditview.jsp TutorEditController.java	Provides facilities for a tutor to edit a tutor's details. This is actually displayed embedded within the tutorlistview.jsp page via an IFRAME.
tutorlistview.jsp TutorListController.java	Lists the tutors in the system, and provides links to edit or delete a tutor.
wlab.jsp tutormenuview.jsp WebUIController.java TutorMenuController.java	As with students, wlab.jsp provides the initial login screen. The WebUIController handles authentication, attempt to authenticate first as a student then as a tutor. If unsuccessful, an appropriate message is passed to the view. Once authentication is successful, if the user is a tutor, the controller then returns a *ModelAndView* object that includes tutormenuview.jsp – essentially "switching" to this view.

Viewless Servlets

A number of "viewless" – i.e. they have no corresponding JSP page and do not use Spring MVC – are used for certain common backend tasks (e.g. starting/stopping VMs, etc). For consistency's sake, they are configured in the **web.xml** configuration file to be accessible via a URL that ends in .html. They can be found in the org.paulneve.wlab.webui package, and are usually suffixed with *Servlet:

Servlet	Purpose and explanation
BasicVMControlServlet.java	Accepts a command to start or suspend a VM, based on a supplied name. Note the name corresponds with the virtualisation backend, not with WLab's own internal VM object names. This is used primarily by studentlablist.jsp to act upon commands to start or suspend labs.
CreateNewBaseVmServlet.java	When a tutor selects the **Clone VM** option to create a new tutor VM from an existing VM, this servlet handles the cloning process and then sees the new VM through the appropriate number of reboot cycles in order to "make it safe" (see the Documentation for Tutors and Documentation for System Administrators). The new VM is created and started, then a new thread monitors the VM until it shuts down. When the thread detects a shutdown, it re-starts the VM, and then the thread terminates. Meanwhile, a redirect will already have been sent to editbasevmview.jsp/EditVmConsoleController.java. This controller checks for the existence of this monitoring thread to determine whether or not the new VM is ready for actual editing to take place. Once the thread no longer exists it knows the VM is ready, and establishes a VNC connection. This servlet is also used when editing an existing tutor VM which has previously been made safe – meaning that it will also need to go through the reboot cycles before it can be edited. In this instance, the process is identical, except that no new VM is actually created at the start of the process.

Servlet	Purpose and explanation
CreateNewStudentVmServlet.java	This is called when a student attempts to access a Lab Stage that they do not yet have a StudentAllocatedVM for. A new VM is created from the tutor VM, and the servlet sees the new VM through the boot cycle(s). There are two modes. One is called with an AJAX request, and occurs when the student is actually running a lab console and attempts to navigate to a new stage. The code for this is found in labconsoleview.jsp and in this instance, the servlet itself "blocks" until the new VM is ready. Because the servlet was called via AJAX this does not impact the user experience. When the new VM is ready, the servlet unblocks and sends a success string, which in turn will fire the onreadystatechange event of the XMLHttpRequest object – in our case in labconsoleview.jsp, the Javascript function backFromCreate is called. The other mode does not use AJAX, and occurs when the student selects the **Create Lab** option from their initial list of labs (studentlablist.jsp). This uses the same process as CreateBaseVmServlet.java where a thread is started to see the boot cycle through to its conclusion, a redirect is sent back to studentlablist.jsp, and the continued existence of the thread is used to determine whether or not the new VM is ready. When the thread ends, the VM can be used.
GetCourseDetailsServlet.java	Solely used via an AJAX call in labeeditview.jsp. Given a search parameter, if the servlet can find a course ID that matches, it returns the course *name*. If it finds a course name that matches, it returns the *ID*. Thus, the servlet is used to populate the course name in labeditview.jsp if the tutor fills out the course number, and vice versa.
LogoutServlet.java	Does precisely what you'd expect, apart in one instance. If it finds both a tutor *and* a student in the session, it assumes that this is a tutor who has used the preview option to "simulate" the experience of a student. In this instance, the servlet clears the student from the session and redirect the tutor to their main menu. Otherwise, the session is invalidated and a redirect sent to wlab.html (which will display the login screen ready for a new login).
PreviewLabServlet.java	This servlet is called when a tutor selects the preview option against a student (essentially, it allows them to assume the "identity" of a student and thus preview their labs from the student's perspective). It simply accepts a student ID, then retrieves the appropriate student and adds them to the session. It then redirects to wlab.html which, if a student is in the session, bypasses the login screen and goes directly to studentlablist.jsp.
ToggleBookingServlet.java	This servlet is called via AJAX when a student clicks a date/time slot on the booking form. If the supplied date/time has a booking already, it is cancelled. If not, one is added.
ToggleStageProgressServlet.java	This servlet is called via AJAX from labconsoleview.jsp. Given a StageProgress ID, if the current status is InProgress it will be changed to Finished, and vice versa.

Classes in the Utilities package

The package **org.paulneve.wlab.utilities** contains a number of classes that do not directly fit elsewhere:

Class	Purpose and explanation
DesEncrypter.java	Adapted from http://www.exampledepot.com/egs/javax.crypto/PassKey.html. Primarily used to encrypt the various parameters supplied to the VNC applet, to avoid end-users selecting “View source” in their browser and discovering details of the internal network.
GeneralUtils.java	General utility class. Methods contained herein are, for the most part, controller aspects that are used across more than one controller class.
MacToIP.java	Converts a MAC address to an IP address. Works by attempting to establish a network connection to every possible machine on the class C subnet configured in the appropriate properties of **application-properties.xml.** The ARP cache is then read to find the corresponding IP address for the desired MAC address. This is required because there is no guaranteed method of extracting an IP address from a Hyper-V virtual machine, if it is not running Microsoft’s Integration Services.
Parameters.java	Used to create a singleton into which the values of **application-properties.xml** are injected.
Strings.java	Used to create a singleton into which the values of **strings.xml** are injected.
VmSuspender.java	Runs resident in a thread, and monitors running VMs to determine which of them should be suspended (if, for example, a student runs beyond their allocated runtime).

Creating alternative implementations of WLab interfaces

WLab's access to the virtualisation backend, authentication, and data storage was written to enable easy implementation of alternative solutions. Java interfaces and/or abstract classes are used for each function; implementations are provided to access Hyper-V, an LDAP-based directory and XML files respectively.

The diagram illustrates these interfaces and the corresponding implementations provided. Classes shown in white/dotted lines illustrate where future work might potentially take place to provide alternative functionality.

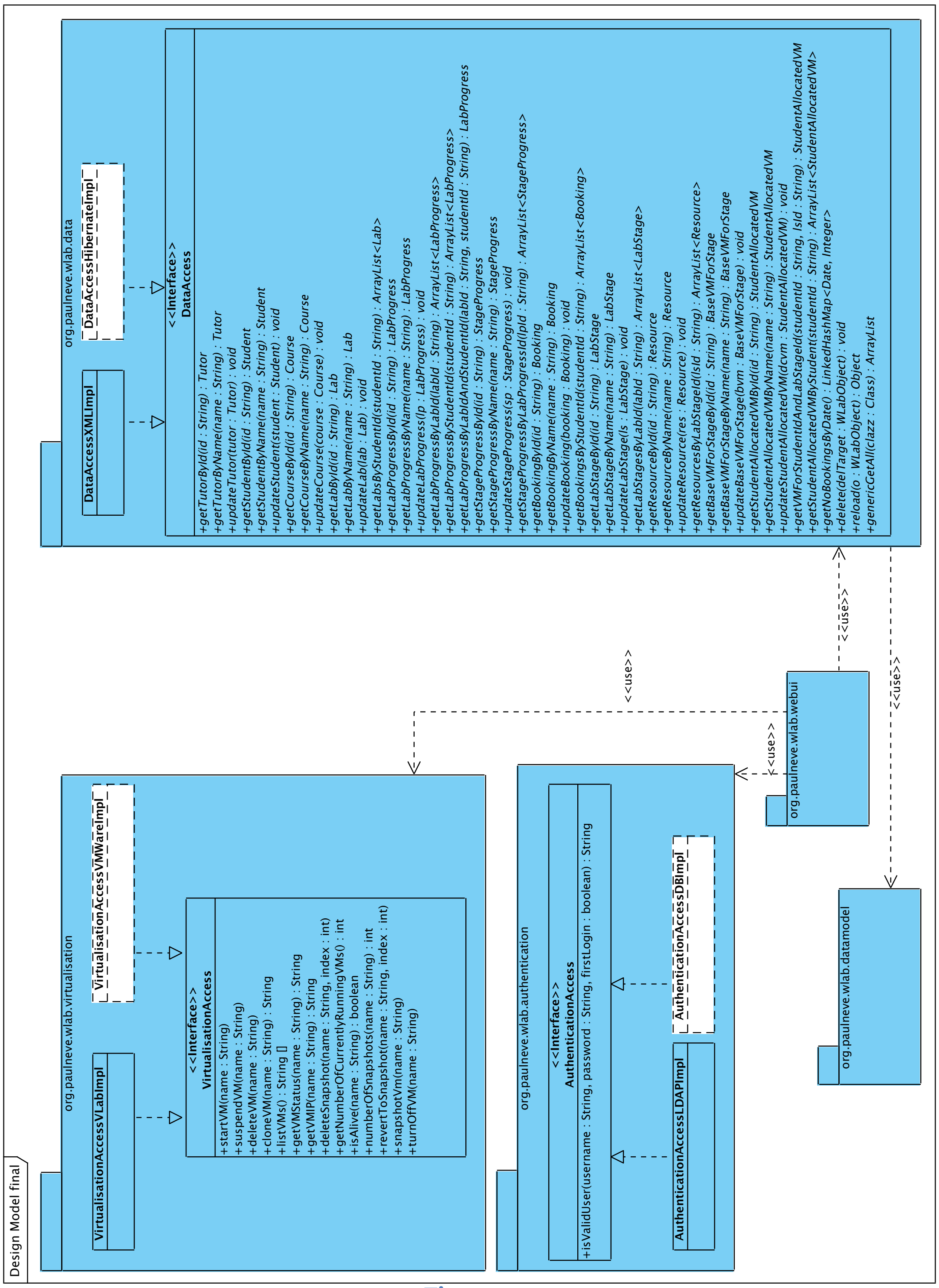

Design Model final
org.paulneve.wlab.data
DataAccessXMLImpl
DataAccessHibernateImpl
<<Interface>>
DataAccess
+getTutorById(id : String) : Tutor
+getTutorByName(name : String) : Tutor
+updateTutor(tutor : Tutor) : void
+getStudentById(id : String) : Student
+getStudentByName(name : String) : Student
+updateStudent(student : Student) : void
+getCourseById(id : String) : Course
+getCourseByName(name : String) : Course
+updateCourse(course : Course) : void
+getLabById(id : String) : Lab
+getLabByName(name : String) : Lab
+updateLab(lab : Lab) : void
+getLabsByStudentId(studentId : String) : ArrayList<Lab>
+getLabProgressById(id : String) : LabProgress
+getLabProgressByName(name : String) : LabProgress
+updateLabProgress(lp : LabProgress) : void
+getLabProgressByLabId(labId : String) : ArrayList<LabProgress>
+getLabProgressByStudentId(studentId : String) : ArrayList<LabProgress>
+getLabProgressByLabIdAndStudentId(labId : String, studentId : String) : LabProgress
+getStageProgressById(id : String) : StageProgress
+getStageProgressByName(name : String) : StageProgress
+updateStageProgress(sp : StageProgress) : void
+getStageProgressByLabProgressId(lpId : String) : ArrayList<StageProgress>
+getBookingById(id : String) : Booking
+getBookingByName(name : String) : Booking
+updateBooking(booking : Booking) : void
+getBookingsByStudentId(studentId : String) : ArrayList<Booking>
+getLabStageById(id : String) : LabStage
+getLabStageByName(name : String) : LabStage
+updateLabStage(ls : LabStage) : void
+getLabStagesByLabId(labId : String) : ArrayList<LabStage>
+getResourceById(id : String) : Resource
+getResourceByName(name : String) : Resource
+updateResource(res : Resource) : void
+getResourcesByLabStageId(lsId : String) : ArrayList<Resource>
+getBaseVMForStageById(id : String) : BaseVMForStage
+getBaseVMForStageByName(name : String) : BaseVMForStage
+updateBaseVMForStage(bvm : BaseVMForStage) : void
+getStudentAllocatedVMById(id : String) : StudentAllocatedVM
+getStudentAllocatedVMByName(name : String) : StudentAllocatedVM
+updateStudentAllocatedVM(dcvm : StudentAllocatedVM) : void
+getVMForStudentIdAndLabStageId(studentId : String, lsId : String) : StudentAllocatedVM
+getStudentAllocatedVMByStudent(studentId : String) : ArrayList<StudentAllocatedVM>
+getNoBookingsByDate() : LinkedHashMap<Date, Integer>
+delete(delTarget : WLabObject) : void
+reload(o : WLabObject) : Object
+genericGetAll(clazz : Class) : ArrayList
<<use>>
<<use>>
<<use>>
<<use>>
org.paulneve.wlab.webui
org.paulneve.wlab.datamodel
org.paulneve.wlab.virtualisation
VirtualisationAccessVLabImpl
VirtualisationAccessVMWareImpl
<<Interface>>
VirtualisationAccess
+startVM(name : String)
+suspendVM(name : String)
+deleteVM(name : String)
+cloneVM(name : String) : String
+listVMs() : String []
+getVMStatus(name : String) : String
+getVMIP(name : String) : String
+deleteSnapshot(name : String, index : int)
+getNumberOfCurrentlyRunningVMs() : int
+isAlive(name : String) : boolean
+numberOfSnapshots(name : String) : int
+revertToSnapshot(name : String, index : int)
+snapshotVm(name : String)
+turnOffVM(name : String)
org.paulneve.wlab.authentication
<<Interface>>
AuthenticationAccess
+isValidUser(username : String, password : String, firstLogin : boolean) : String
AuthenticationAccessLDAPImpl
AuthenticationAccessDBImpl

Figure 2

To use any alternative implementations of these interfaces, you will need to modify the Spring configuration accordingly. If, for example, you provide a new implementation of **VirtualisationAccess**, e.g. **VirtualisationAccessVMWareImpl**, to make use of it the **virtualisationAccess** bean in **application-properties.xml** will need to be modified to refer to your new class. Any parameters your new class expects to be injected will also need to be included in the bean definition.

Further information about these interfaces can be found in the Javadoc, available at http://www.paulneve.com/wlab/javadoc.

Documentation for Students

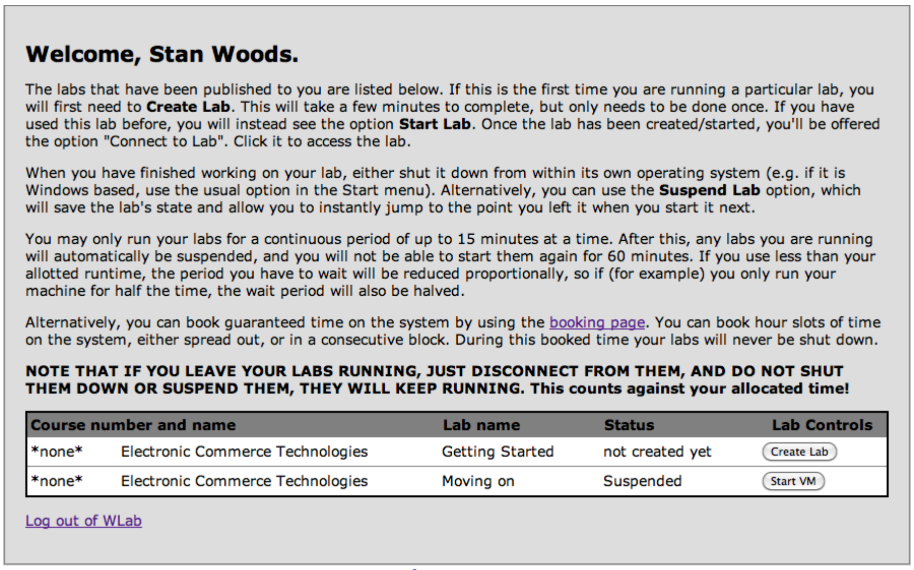
Introduction

The WLab system allows your tutor to design and publish practical lab exercises using a computer, and for you to undertake these exercises via any web browser. You can work either on- or off-campus, and WLab requires no additional software to be set up on your computer as long as your web browser can run Java applets[12].

Getting into the system

Your tutor should have given you a URL where you can access the WLab system. Load a web browser, enter the URL and log in with the appropriate username and password. If your tutor has not advised you otherwise, this will usually be the same username and password that you use to access your standard institutional IT facilities.

After a successful login, you should see a screen similar to the following:



Welcome, Stan Woods.

The labs that have been published to you are listed below. If this is the first time you are running a particular lab, you will first need to **Create Lab**. This will take a few minutes to complete, but only needs to be done once. If you have used this lab before, you will instead see the option **Start Lab**. Once the lab has been created/started, you'll be offered the option "Connect to Lab". Click it to access the lab.

When you have finished working on your lab, either shut it down from within its own operating system (e.g. if it is Windows based, use the usual option in the Start menu). Alternatively, you can use the **Suspend Lab** option, which will save the lab's state and allow you to instantly jump to the point you left it when you start it next.

You may only run your labs for a continuous period of up to 15 minutes at a time. After this, any labs you are running will automatically be suspended, and you will not be able to start them again for 60 minutes. If you use less than your allotted runtime, the period you have to wait will be reduced proportionally, so if (for example) you only run your machine for half the time, the wait period will also be halved.

Alternatively, you can book guaranteed time on the system by using the booking page. You can book hour slots of time on the system, either spread out, or in a consecutive block. During this booked time your labs will never be shut down.

NOTE THAT IF YOU LEAVE YOUR LABS RUNNING, JUST DISCONNECT FROM THEM, AND DO NOT SHUT THEM DOWN OR SUSPEND THEM, THEY WILL KEEP RUNNING. This counts against your allocated time!

Course number and name		Lab name	Status	Lab Controls
none	Electronic Commerce Technologies	Getting Started	not created yet	Create Lab
none	Electronic Commerce Technologies	Moving on	Suspended	Start VM

Log out of WLab

Figure 89

At the bottom of the screen, the lab exercises that have been published to you by your tutor will be listed.

Using a lab for the first time

The first time you use a lab, it will need to be created by you. You can see that this is required if, in the **Status** column against a particular lab, it shows the text **not created yet**. You will see that the corresponding **Lab Controls** will show the single option, **Create Lab**. Click it; the screen will change and you will see the text

```
Creating a new lab. You will not be able to change any other lab
states until this has finished. Please wait.
```

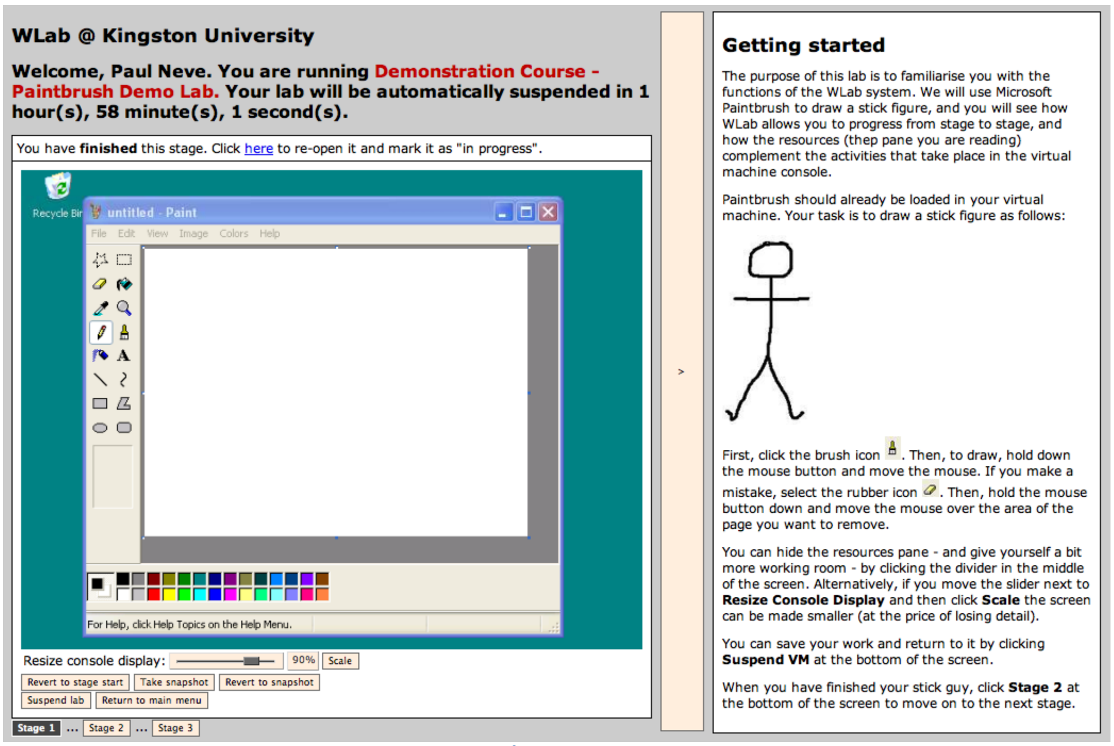
It will take a few minutes for the lab to be set up for you. When complete the **Lab Controls** will then show the option **Connect to lab**. Select that to access your lab session.

On subsequent accesses, you will not need to create the lab. Instead, you will see the option **Start Lab**, and once this is used, you'll be able to immediately **Connect to lab**.

[12] If you are not sure whether or not this is the case, visit the URL http://www.java.com/en/download/help/testvm.xml. You should see the text **Your Java is working**.

The lab environment

The WLab lab environment looks like the below:



WLab @ Kingston University

Welcome, Paul Neve. You are running Demonstration Course - Paintbrush Demo Lab. Your lab will be automatically suspended in 1 hour(s), 58 minute(s), 1 second(s).

You have **finished** this stage. Click here to re-open it and mark it as "in progress".

Getting started

The purpose of this lab is to familiarise you with the functions of the WLab system. We will use Microsoft Paintbrush to draw a stick figure, and you will see how WLab allows you to progress from stage to stage, and how the resources (thep pane you are reading) complement the activities that take place in the virtual machine console.

Paintbrush should already be loaded in your virtual machine. Your task is to draw a stick figure as follows:

First, click the brush icon . Then, to draw, hold down the mouse button and move the mouse. If you make a mistake, select the rubber icon . Then, hold the mouse button down and move the mouse over the area of the page you want to remove.

You can hide the resources pane - and give yourself a bit more working room - by clicking the divider in the middle of the screen. Alternatively, if you move the slider next to **Resize Console Display** and then click **Scale** the screen can be made smaller (at the price of losing detail).

You can save your work and return to it by clicking **Suspend VM** at the bottom of the screen.

When you have finished your stick guy, click **Stage 2** at the bottom of the screen to move on to the next stage.

Figure 90

At the top left of the screen, a count down is displayed. Each student is allocated a maximum amount of lab run time that they can use in a single sitting. When you exceed this period, any labs you have running will automatically be suspended, and you will be prevented from starting labs for a period of time specified by your tutor (this is to prevent students from "hogging" the server).

Underneath the countdown, you will see either the text

`This stage is` **`in progress`**`. Click` <u>`here`</u> `to mark it as finished.`

or

`You have` **`finished`** `the stage. Click` <u>`here`</u> `to re-open it and mark it as "in progress".`

The use of these options will be discussed shortly.

The two most important aspects of the lab screen are the console, and the resources pane.

The console is on the left hand pane, and is where you will work on the exercise itself. Think of the console area as like a window on a lab PC – clicking on the appropriate components in this window should run programs, select menu items, and so on. Your tutor should have provided you with everything you need in the console to undertake the activities of the lab exercise.

If the tutor has chosen to publish any resources for your lab, they will be shown in the right hand pane. Resources are static materials that are designed to help you with the activities of the lab – think of them like an electronic version of the handouts you'd usually receive to accompany a practical exercise. A resource might explain step-by-step what you need to do to achieve the lab activity, or simply pose questions and try to stretch your knowledge during the exercise – this will depend on your tutor! If there is more than one resource, links to navigate between them will be shown at the bottom of the resource pane (you may need to scroll down).

Using the *Resize Console Display* facility to see more of the console

You can hide the resource pane – and thus give yourself more room to work in the console – by clicking the divider bar that separates them. This is useful if some of the console window is cut off in order to show the resource pane. However, this might not be sufficient for some lab exercises, where the desktop might be so large it simply doesn't fit into a web browser window.

In this event, you can use the **Resize Console Display** facility to show more of the console window. You can make the individual components shown within the console window smaller, which means more of the console window can be shown within the same space in your browser. Slide the bar next to the text **Resize Console Display** and choose a value. 50% will make the components within the console half as wide and half as high as normal (so they will be a quarter of their normal size, meaning you can see 4 times as much of the console in the original space). When you have chosen a value, click **Scale** and the console will re-size itself accordingly.

Note that the price you pay for this additional screen real estate is a loss of detail – text and graphics will not as as clear as when the Scale setting is 100%.

Marking labs when they are finished

It is very important that, when you have finished a lab stage, you mark it as such by using the link above the console pane. Your tutor will use this information to evaluate your progress and WLab has no other way of "knowing" that you have finished this part of the lab exercise. If you mark a lab stage as finished and later decide this is not the case, you can use the same function to re-open it and again set it to an "in-progress" state.

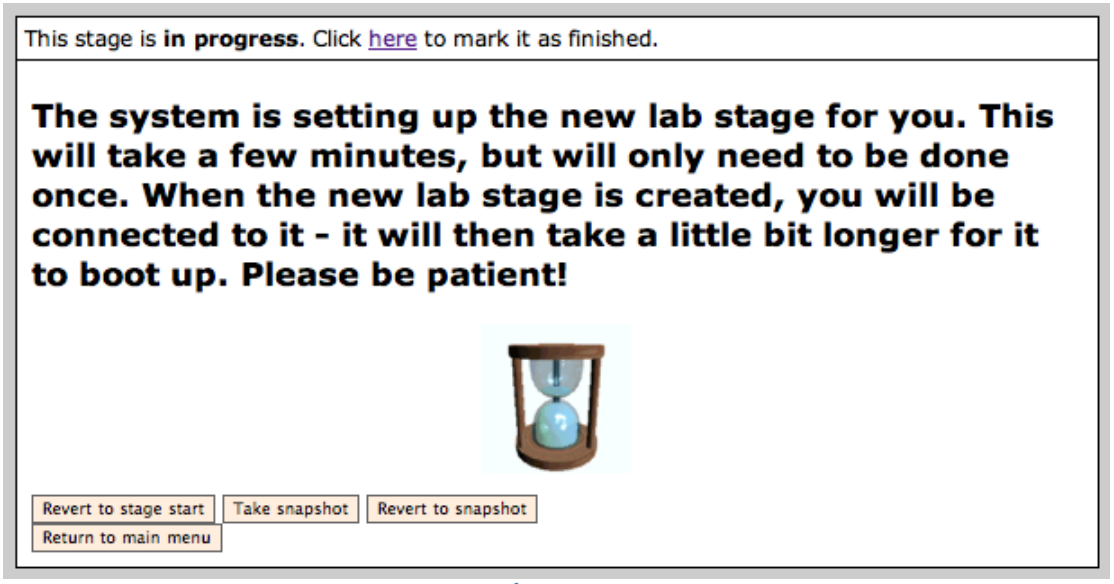
Navigating lab stages

Most labs will be divided into stages. Each stage will have its own set of activities, and in most cases subsequent stages will build on the previous. So, for example, the end point of stage 1 will usually be the start point of stage 2.

When you navigate to a new stage, the console will automatically change its state to bring in everything you need for this stage. This means that the data in the console will change – so if you were stuck on the previous stage, don't worry, because moving to the new stage will automatically give you everything you need to embark upon the next part of the lab.

To navigate between lab stages, there are buttons underneath the console pane, one for each stage (how many there are will obviously depend on the lab itself). The current stage will be highlighted; to move to the next (or indeed any other stage) simply click the appropriate button.

The first time you go to a lab stage, it will need to set itself up for you. You will see a message like the following:


This stage is **in progress**. Click here to mark it as finished.

The system is setting up the new lab stage for you. This will take a few minutes, but will only need to be done once. When the new lab stage is created, you will be connected to it - it will then take a little bit longer for it to boot up. Please be patient!

Figure 91

As was the case when the initial lab was created, this process will take a few minutes to complete. Half way through the process, the screen will change, and instead you will see the message:

```
Waiting for your session to come on line. Please wait.
```

30 seconds or so after that, the console should re-appear, with the starting point of the next stage ready to go.

This creation process only has to happen once per lab stage. Once the lab stage has been set up for you, re-visiting it is more or less immediate. If you move backwards and go to a previous stage, the previous stage will load in exactly the state you left it in.

Using the snapshot facilities

WLab allows you to take a "snapshot" of a lab stage state. Think of this as a way of "saving" your entire lab stage. If you take a snapshot before you embark upon something new, you can revert back to the snapshot if it doesn't work out as you hope.

To take a snapshot, click **Take snapshot**. You will see the console reload itself and after a moment or two it should return you back to your lab exercise. However, once a snapshot has been taken a new option is available, **Revert to snapshot**. If you select this, the lab stage will lose any changes since the snapshot was taken, and it will revert to exactly the state as was then.

The system automatically takes a snapshot when the lab starts for the first time. If you really make a mess of a lab stage, you might wish to use the option **Revert to stage start**. This will take you all the way back to the state where you first started the lab stage – i.e. where the tutor expected you to start that lab stage from. **NOTE THIS WILL LOSE ANY AND ALL WORK YOU'VE DONE IN THIS LAB STAGE, SO YOU SHOULD BE VERY CAREFUL BEFORE USING IT!**

Closing down your lab

When you have finished working, you should *always* use the **Suspend lab** button. This will close your lab down, and return you to the list of labs. In contrast, **Return to main menu** will do just that and return you to the main menu, *but it will leave your lab running on the server*. The only time you should ever use the **Return to main menu** option is if your tutor has told you to do so (if, for example, you are going to run several labs at the same time).

Booking guaranteed time on the system

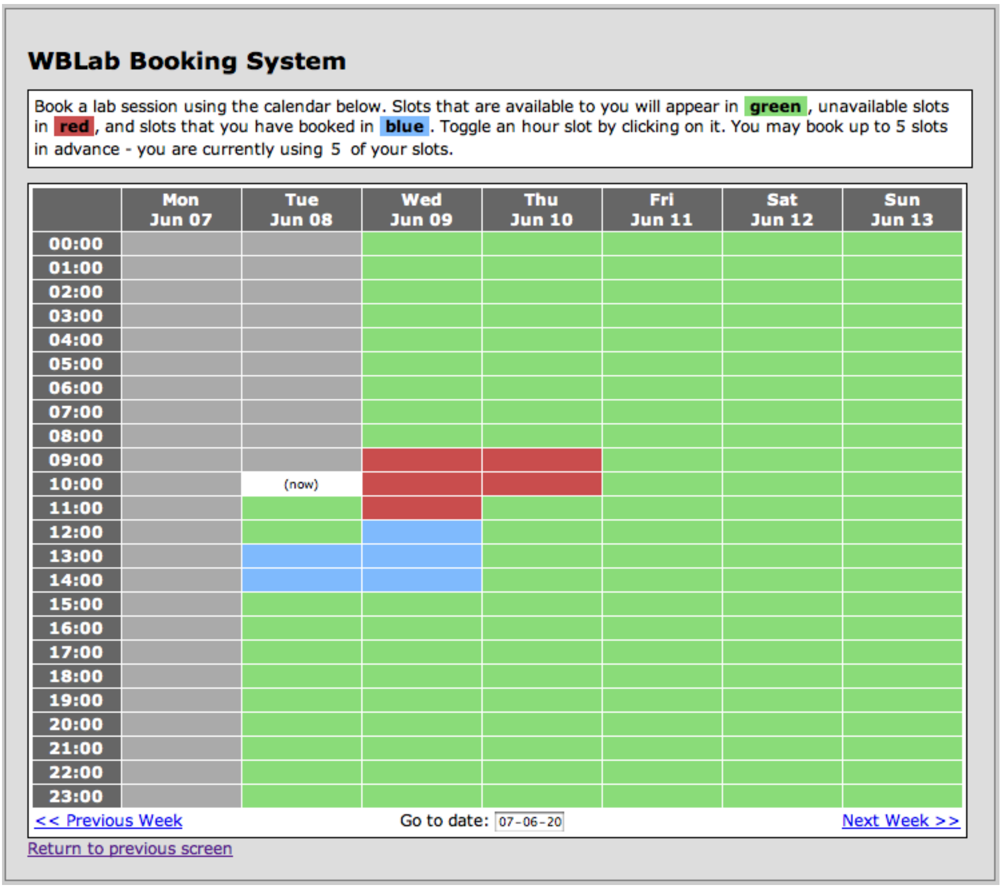
The WLab system restricts students to a limited time window during which they can run labs, so as to ensure that everybody gets a fair share of system resources. The amount of time available to you is displayed on the initial page after login. If you run your labs for the entire allocated time, they will be automatically suspended and you will be prevented from starting them again for a defined "interregnum period" – this is also shown on the same page.

If you run your labs for a period of time shorter than the maximum, you are still subjected to the interregnum period, but this is reduced proportionally depending on how long you used. The table below illustrates this:

Student's maximum runtime	Maximum interregnum period	Student's actual runtime	Actual interregnum period
2 hours	30 minutes	2 hours	30 minutes
2 hours	30 minutes	1 hour	15 minutes
2 hours	30 minutes	15 minutes	3 minutes, 45 seconds

You may also be restricted from starting labs if other students are already using the system to its maximum capacity.

In order to allow you the ability to choose when you work, a booking system is available and you can access it by selecting the link to the **Booking Page** from the initial post-login screen:



WBLab Booking System

Book a lab session using the calendar below. Slots that are available to you will appear in **green**, unavailable slots in **red**, and slots that you have booked in **blue**. Toggle an hour slot by clicking on it. You may book up to 5 slots in advance - you are currently using 5 of your slots.

	Mon Jun 07	Tue Jun 08	Wed Jun 09	Thu Jun 10	Fri Jun 11	Sat Jun 12	Sun Jun 13
00:00	▨	▨	🟩	🟩	🟩	🟩	🟩
01:00	▨	▨	🟩	🟩	🟩	🟩	🟩
02:00	▨	▨	🟩	🟩	🟩	🟩	🟩
03:00	▨	▨	🟩	🟩	🟩	🟩	🟩
04:00	▨	▨	🟩	🟩	🟩	🟩	🟩
05:00	▨	▨	🟩	🟩	🟩	🟩	🟩
06:00	▨	▨	🟩	🟩	🟩	🟩	🟩
07:00	▨	▨	🟩	🟩	🟩	🟩	🟩
08:00	▨	▨	🟩	🟩	🟩	🟩	🟩
09:00	▨	▨	🟥	🟥	🟩	🟩	🟩
10:00	▨	(now)	🟥	🟥	🟩	🟩	🟩
11:00	▨	🟩	🟥	🟩	🟩	🟩	🟩
12:00	▨	🟩	🟦	🟩	🟩	🟩	🟩
13:00	▨	🟦	🟦	🟩	🟩	🟩	🟩
14:00	▨	🟦	🟦	🟩	🟩	🟩	🟩
15:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
16:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
17:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
18:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
19:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
20:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
21:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
22:00	▨	🟩	🟩	🟩	🟩	🟩	🟩
23:00	▨	🟩	🟩	🟩	🟩	🟩	🟩

<< Previous Week   Go to date: 07-06-20   Next Week >>

Return to previous screen

Figure 92

Bookings are available in hour-long slots. Click a green slot to book it – it will change to blue. Red-coloured slots indicate slots that are already booked to maximum capacity. Use the links at the bottom left and bottom right to move between weeks, or click the date to select a date from a calendar.

You will only be able to make a limited amount of future bookings. It is up to you how you use these – you might want to place them one after another and give yourself the longest possible guaranteed slot, or instead, you might want to stagger them and make sure you have regular lab access albeit for smaller amounts of time.

You cannot change your bookings if, at the current moment in time, you have a booked session in progress.