Progress and Revisions

1. Progress and problems

As of the time of writing, there has been one iteration - which was scheduled for completion on Monday 01/11/10, but was pushed back to the Friday (05/11/10) - presented with a selection of functionality to the client, albeit in a fragmented form. The first iteration did not meet all the acceptance criteria we had set ourselves, due to some of the problems identified in our first report (See Appendix 1, Excerpt 1, and also reference the original report). Our next iteration is planned for Monday 15/11/10, but our client will not review progress until the Friday 19/11/10 as he is unavailable until that time.

Features implemented so far

For useful discussion on the features implemented so far, a basic knowledge of the technical aspects of the project is required. The reader should be familiar with the introduction given to these concepts in Project Report 1, which for convenience is enclosed in Appendix 2 below.

- A running server is able to deliver a web front end, progressively exposing features to the user, as and when they are implemented.
- Changes made by the user on the web front-end (referred to hence-forth as “client-side”) are stored in a persistent way on the back-end (sometimes referred to as “server-side”).
- User sessions are authenticated using a password - currently all users share an account. A single set of login credentials is likely to remain, as only one instance of the Vazels control centre can run on a machine at a time.
- Changes that the user makes are sent to the server asynchronously meaning that the user can continue working on the client-side whilst the server processes said changes.
- Groups (see Appendix 1, Excerpt 5) can be created from the client-side and stored on the back-end.
- The server can start and stop the Vazels control centre.
- The server is able to accept workloads specifications from the client, but cannot yet assign them to groups. This is due for completion today.
- The server can accept archives containing actors, and assign them to any dependent workloads.
- Server can store persistent data to guard against failure and a need to be restart the back-end.

Problems

- The toolkits used were novel to the group (e.g. Git and the Google Web Toolkit), and the learning curve was steep. Some members of the group coped with this differently.

    - Mitigation: All members committed to completing online tutorials provided for each toolset. This was completed as of 08/11/10, a little later than ideal. Group members are still having some problems getting used to using Git, but we feel that from a personal development point of view this is worthwhile.
- Some members of the group were not able to run the Vazels system, and had limitations with Git. This was because they were running Microsoft's Windows operating system, where Vazels (despite being written primarily in Java) does not run, and whilst Git will run, certain technical limitations of Windows impair parts of the functionality.
    - Mitigation: Simon has installed Ubuntu on his laptop alongside Windows; Chris is virtualising a working Lubuntu operating system (a linux variant based on Ubuntu), and Andreea has taken steps to work around limitations with Git, while running Vazels directly from the Department of Computing Lab computers.
- Communication remains difficult with varied timetables and academic commitments (See Appendix 1, Excerpt 1)
    - Certain steps mentioned in the first report have been effective, for instance we are having less frequent meetings, organised to fit around all our schedules at a point when we have enough spare time to cover all the issues we need to go through. (See Appendix Excerpt 2). Technologies like Doodle[1], and Google Calendar have been essential to scheduling for five group members each with very different availability. Using larger work packages has also been effective for us, providing each group member with plenty to do in between meetings.
    - **Further mitigation:** All members are now actively uploading tickets to the issue tracker whenever they discover a new issue or task, so that it can be recorded without direct contact with other group members. All members are now taking responsibility for taking tickets from Trac if they have finished their own work. The benefits of this are two-fold: a group member no longer feels compelled to fix every problem as they come to it (they simply record it for later and move on) - meaning they can continue focusing on the key requirements of their existing task; and group members with spare time can pick up issues in time for the next meeting - when all major tickets should get assigned.
    - Modern infrastructure (Skype, mobile phones and instant messaging) makes communication easier when it is essential, even without face-to-face contact. This means that group members do not always have to meet in person to discuss every issue - a serious concern when an issue is blocking but a group member is not available to meet (e.g. On Wednesday we conducted a two-hour meeting with one of the participants on Skype - otherwise he would have missed the meeting).
- Department of Computing networks assigned unusual hostnames to guest computers (e.g. our laptops). This held up progress with running the Vazels program at the beginning of the project.
    - Mitigation: Have discussed with CSG how the network is set up. Consequently, we will now only run Vazels on a lab machine, rather than trying to use our personal laptops in the labs.

[1] http://www.doodle.com/

2. Revisions to Project Report 1: Project Inception

Key requirements

These remain unchanged since the Project Inception, with one exception. The other requirements are now fixed unless there is significant justification. Key requirements have been discussed where relevant with our supervisor, Prof. Wolf, and only the following major change was made:

- **Key Requirement Change** (See Appendix 1, Excerpt 4): The requirement to be able to "Run a [specific] workload associated with a group" is not achievable, due to limitations in the Vazels platform, uncovered during the "Pause for revision..." that was part of Milestone 2 (See Appendix 3 for our original draft schedule). The Experiment as a whole is started with a single command, and workload specifications (uploaded by the user during setup phase) are responsible for telling the system when their different sections will run. In light of this, part of the original requirements was both impossible and redundant (although the display of host status remains within the project scope).

Feature extensions

These also remain largely unchanged, and are still subject to producing a fully functional prototype fulfilling all key requirements. The order of priority for our extensions is as follows (Please see the first report for a full explanation of planned extensions):

- **Display of collected data:** We have now reached a stage that the system is able to provide the user with raw dumps of data so we are able to begin to consider finding improved ways of presenting this to the user. A good test of this may be to begin to design the pre-fabricated tests that we are able to predict the output format of (See Appendix, Excerpt 3). It is hoped that by taking a brief inspection of the user's own tests we will be able to predict the best way of presenting it, e.g. a series of numerical values would call for a graph, whilst a series of text values would maybe be better displayed on a timeline, and individual values are probably best just printed
- **Pre-fabricated tests:** Whilst pre-fabricated tests themselves are not a high priority for us at the moment, they may be implemented in the process of implementing the first extension listed here.
- **Saving test configuration:** Since many of the configuration settings are stored in persistent storage on the server-side in our existing implementation, it would be nice to be able to communicate that back to the client in a way that allows them to "save their work" and come back to it later. We would probably do that in the form of a single archive file, which we would encourage the user to think of in the same way as a "project" file in any well-known IDE.
- **New possible extension:** To deliver a system usable by a person with no prior experience of the Vazels system. This was not one of our original key requirements, and may not fall within the scope of this project, given time constraints, but would certainly be a very worthwhile extension.
- **"Point-and-click" interfaces for configuration:** This is now a low priority. It would be a

worthwhile usability feature, but is unlikely to fall within the scope of this project.

Measuring progress

Progress measures remain centred around the acceptance and completion of Trac tickets. As expected, different tickets require different levels of workload before being deemed completed so it is difficult to quantify the progress with completed tickets alone. We are still able to measure overall project progress using the planned milestones outlined in Project Report One. To track individual progress, rather than the progress of the group as a whole, during each group meeting we ensure we keep track of individual progress (and difficulties) with tasks and if necessary reconsider task allocation (we discuss the more in Section 3). We try to assign all existing tickets at the end of each milestone (for the start of the next).

Major risks:

While no new major risks are foreseen, time pressures are making it difficult for all members of the group to achieve their deadlines.

- **Mitigation**: We are taking active steps to ensure that if a group member thinks they may be unable to complete their work within the given time, the issue is raised sooner rather than later, ensuring that others can help them achieve their tasks.
- **Further Mitigation**: since this is in part a people management issue, we have discussed this further in Section 3.

Schedule revisions

Please see Appendix 2 for a copy of our original draft schedule.

The Schedule So Far

- Research and concept exploration were completed within the time constraints.
- Project requirements were finalised as expected within the time constraints expected.
- The first iteration was not completed in time for 01/11/10, as planned in the original report. The reasons have been discussed above in the “Problems” section, and in the introductory paragraph, but a very basic prototype with limited functionality was demonstrated on Friday 05/11/10.
- The “User Stories” task has been revised, and moved to the later deadline of 15/11/10, with completion expected to be ahead of the new schedule. The task is now less client-focused, due to the client’s limited experience with the existing system (subject to the extensions mentioned above, a basic level of knowledge of the Vazels system is assumed of the user). The project remains focused and driven, because - as Computer Scientists taking Distributed Systems opens - at least three members of the group are likely to be end users of this project if it is successfully implemented.
- Work towards starting the control centre with the user’s configuration of Groups, Actors, and Workloads is on schedule to meet requirements for the milestone on 15/11/10 - our second iteration.

Remaining tasks

- Although test data is a requirement to displaying output, the functional aspect of doing

so does not require us to generate new test data through our system; we have made the decision to start work on displaying output, using data acquired from tests run via the command line. This has meant that work originally planned to begin for Milestone 5 (22/11/10) has begun before the 2nd iteration is complete (Scheduled for Milestone 4 on 15/11/10). This will allow us to spend more time exploring different ways displaying final test data to the user, which in the future may deliver a compelling motivation for using our UI.

- For Milestone 5, we planned to spend a large amount of time on displaying feedback to the user, and assessing feasibility of our requirements. The previous paragraph outlines that more progress than expected has been made already in terms of displaying feedback. We plan to devote effort to exploring the feasibility of "Prefabricated tests," and have a view to de-scoping some of the other extensions in the next week after a little more investigation.
- We plan to take the opportunity leading up to Milestone 5 to catch up on a small backlog of tasks in Trac which have been put to one side with the aim of satisfying basic functional requirements. This should give us a little more breathing space, and allow us to ensure we're happy with our 2nd iteration. This means that if we have to make compromises in order to deliver our 2nd iteration on time, we will have time to address issues (as well as client feedback) before starting what was planned to be our 3rd iteration
- **Functional key requirements finished by Milestone 6 (29/11/10)**: we feel it is worth re-iterating that this ambitious goal is still a part of our schedule. It is also worth noting that part of Milestone 7 (due 6/12/10) is to finish implementing any unfinished features due in Milestone 6, giving us room to prioritise if we do run out of time.
- The remainder of our original draft schedule remains unchanged.

3. People Management

- Group meetings have worked well. They have provided evidence of how much each person has produced, and although everyone has provided different levels of input, there has been input from everybody. The lectures on agile development and the scrum have helped encourage us to each take some time during each meeting to talk about the recent progress we have made and any problems we have been having with it. Specifically, we are making sure we update each other during the time between meetings on the progress we are making.
- An issue that has arose is that Andy has spent longer on the project than most others, and produced much more useful output. It would be unfair to ensure everybody did as much work, but nonetheless, this leads to an uneven, and unfair distribution of work. We do not see it as necessary to provide a maximum time commitment, but wish to avoid any resentment that may arise from a feeling of an unbalanced commitment. As such, agreeing on a universal minimum ensures that even the hardest-working members of the group have agreed on an acceptable level of work for the others to be doing.
- On a related issue, that of group members having different levels and areas of skill, we have had some issues. Naturally, this has led to some people being more productive,

having already been familiar with the technologies and concepts involved. We gave the tasks suited to particular members to those people, for example Andy had more input on the server side as he has experience in this. We have tried to ensure that each member of the group recognises the time spent on tasks by other group members. This is helped by a discussion at the meetings not only of what has been achieved but also of the obstacles that have been encountered, which leads to an understanding of why a task may not have been achieved and prevents dissension.

4. Ethical and Environmental impact

This project is an interface to an existing set of tools (Vazels), so it is our belief the ethical and environmental risks associated with the *use* of our work are primarily based upon our work making existing risks more accessible (if our project is successful, it may attract new users to the Vazels system). The cloud to this silver lining is that we must address concerns revolving around basing our work so heavily upon work that has already been done.

Risks posed by our project (environmental and ethical)

- **Potential misuse:** Vazels allows a user to automate the function of a large group of computers all at the same time. However, this capability is dependant on an already existing infrastructure of machines, and relies on a central control centre, so it does not provide an ideal platform for e.g. DDOS[2] attacks. In order to use the tools, the user must already have the ability to run code on the machines in the testbed, so no new risks are introduced (except, perhaps, for convenience).
- **Access to the Vazels Control Centre:** Our server provides a web interface to the Vazels control centre. Potentially, this allows unauthorized users to use the Vazels Control Centre, which previously required users to log in via SSH. However, to exploit its capabilities, the user needs privileges to run code on remote machines. To help avoid the risk of unauthorized users sending commands to the control centre, we have leveraged the existing password authentication mechanism in the Restlite platform (on which our back-end is built).
- **Transfer of private SSH keys by Vazels:** Vazels sends private SSH keys for over the network to remote machines. This is a potential security risk, but an issue in the existing system, and so will not be addressed within the scope of this project.
- **Environmental impact:** As we discussed in the summary paragraph above, no *novel* environmental risks are introduced by our project, which simply presents a user-friendly interface to an existing system, however, we are improving the usability of a system designed to run on potentially many machines. If a user of Vazels runs an experiment comprising 400 computers, each with a modest 250W power supply, all of which would be switched off otherwise, for 5 hours to simulate a series of traffic spikes, then that would draw 500KWh charge; equivalent to approximately 1/20th of the average American household's yearly power usage[3]. If our system were to empower 10 extra

[2] http://en.wikipedia.org/wiki/Denial-of-service_attack

[3] http://wiki.answers.com/Q/How_much_energy_does_the_average_household_use

users to run this kind of experiment on Vazels for every year of the project's lifetime, this could have a real - but not unacceptable - impact upon the environment. We do not intend to take steps to mitigate this risk, as we do not see it as a significant one, and our contribution to this risk is limited to making an existing system more usable. We should note, when considering the impact of this risk, that the system is designed to be used in conjunction with existing cloud services (such as PlanetLab[4]), and if this is made easier, users may choose to switch to these services rather than using their own machines. In effect, our work could reduce the amount of hardware that is used globally, by consolidating fragmented solutions.

- One of the foremost uses of distributed systems (and thus potentially, test suites designed to test distributed systems) is environmental modelling. Our project, if completed to a high enough standard, could aid the development of such systems, and thus potentially have a net positive effect on the environment and our knowledge about it.

Ethical issues on the topic of basing our work heavily on the work of others

- **Vazels:** Our project is providing a user-friendly front-end to Vazels, and we feel it is important that any legal restrictions applying to our work are in-keeping with the spirit of openness and collaboration. The Vazels project is released online under the GPLv3[5], so we plan to release our code under this or a compatible license. Following concerns within the group over code ownership (we were unsure whether, for instance, Imperial College would own our code), we dispatched an email to Michael Huth to enquire about this issue. Following that exchange, we concluded it was reasonable for us to pursue a course of action that would lead to our code being made open-source. Aside from meaning that we can release our code in a spirit in-keeping with the project we are building on top of (which was an ethical concern), from a practical point of view this also means that we can make use of solutions like GitHub[6] and Lighthouse[7] (well-known collaboration platforms) which normally reserve their free services to open-source projects (a practical concern). We do not intend to take any action on this until after we have finished development, as we have no wish to interrupt our work-flow, but have every intention of publishing our code in the future, with certain group members keen to continue work on this project in their own time after the assignment has concluded.
- **Other toolkits:** We are making extensive use of existing tools and frameworks. Aside from licensing concerns with Vazels, discussed above, we also had to inspect licenses of the other toolkits we have been using:
    - **Google Web Toolkit**: Released under the Apache License, v2.0[8]. Primarily, the license speaks to the modification and re-distribution of the toolkit itself, along

[4] http://www.planet-lab.org/
[8] http://www.apache.org/licenses/LICENSE-2.0.html
[5] http://www.gnu.org/licenses/gpl-3.0.html
[6] https://github.com/
[7] http://lighthouseapp.com/

with derivative works. The Google Web Toolkit licenses do not make any claim to the code and content generated by the toolkit, only to the toolkit itself. (Details on the role the Google Web Toolkit plays in our project are outside the scope of this report, but details of the toolkit itself can be found on the project's website[9])

- **Restlite:** Our project makes extensive use of the source code of the Restlite project[10] of which our Restful back-end is a derived work. This is released under the LGPL[11] and whilst not being as restrictive as the GPL, this license does pose certain restrictions on the release of our software. It is, however, compatible with the GPLv3 (if not the GPLv2). We are required to display appropriate notices in the source of our project. Since our intention is to make this code open-source, the use of the LGPL in this toolkit should not pose a problem. We will have to be careful to distinguish between our original work and the code that was part of the Restlite project, in order to avoid issues surrounding plagiarism.
- **Git**: Released under the GPLv2[12], which makes no claim to any of the files organised by the tool, only its own code. This is not a concern for us as we only use the tool; we do not re-use its source code.
- **Eclipse**: Released under the permissive Eclipse Public License[13]. The use of the industry-standard Eclipse IDE to produce our code is not going to cause us any licensing concerns.

Appendix 1

Below are sections from our previous report referred to in the body of the text, listed for convenience. Please see original report to see the excerpts in better context.

Excerpt 1 (page 6 - "First Iteration Plan, Potential Risks")

- Failure to meet these short time deadlines. Realistically, we will be hard-pushed as a team to meet all these goals by Monday. To mitigate this, group members with smaller workloads (or who finish early) will have to help others to get their work done on time.
- Failure to achieve expected results with the GWT platform which we are using to produce the client-side interface. Using Google's GWT platform, we are hoping to harness the power of statically-typed Java, which in our experience is a much more productive environment. As always though, adopting a new toolkit has a set of risks and unexpected problems.
- At this stage, we have yet to write any tests. We are aware of this as an issue, and the potential for code regressions. Our review process should help with this, and our current code-base is small enough that the risk should not be great, but as we move forward we hope to achieve much better coverage.

---

[9] http://code.google.com/webtoolkit/

[1] http://code.google.com/p/restlite/

[1] http://www.gnu.org/licenses/lgpl.html

[1] http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

[1] http://www.eclipse.org/legal/epl-v10.html

Excerpt 2 (page 3 - “Software Development Model”)

> We aren’t having the daily meetings recommended by many Scrum manuals, instead preferring to have slightly larger work packages to allow individual flexibility and personal responsibility for time-management.

Excerpt 3 (page 3 - “Extensions”)

> Prefabricated tests – currently all tests are user-written. We think it would be useful to have some pre-made tests that a user can “mix in” with their existing tests and workloads. These tests might provide data such as open network connections, or current CPU usage. Since we already know the type of data returned by these tests, we should be able to provide the results in a compelling user interface.

Excerpt 4 (page 2 - “Key requirements - running phase”)

> Display which hosts are online or offline at any given time, categorised by Group, with the ability to have a Group of Vazels run a workload that has been assigned to it.

Excerpt 5 (page 1 - “Introduction”)

> We, intuitively, call a group of hosts which are configured identically a Group. A Group consists of a number of remote hosts, and specifies what the hosts it contains can do. It does so by means of a Workload, which tells the system about snippets of code which will be stored on the hosts of the group and named Actors.

Appendix 2 - Project Introduction

Vazels was a Masters Project undertaken by Angel Dzhigarov last year. He developed a solution for managing comprehensive test suites over a distributed network. The running system has at its head a Control Centre, which administers several groups of remote hosts. From the Control Centre, a user can start or stop the experiment, configure a set of tests for remote hosts to carry out, and collect data from the running hosts in the network.

The user can also perform certain administrative tasks, such as configuring which tests should run on which hosts and how many of each type of host there should be on the network. We, intuitively, call a group of hosts which are configured identically a Group. A Group consists of a number of remote hosts, and specifies what the hosts it contains can do. It does so by means of a Workload, which tells the system about snippets of code which will be stored on the hosts of the group and named Actors.

Broadly speaking, the hosts fall into two categories: Vazel machines, and SUE machines. Vazel machines hold user-created tests, which are invoked at runtime and will report back information. SUE stands for “System Under Experiment” – it is the system that the user is trying to test out using the distributed network. It is worth noting that a host can be both a SUE machine and a Vazel machine – that is, it is both running the System Under Experiment, and running a set of

user-created automated tests.
At the moment, a user must utilise the command line to start the Control Centre, specifying the names and sizes of all the groups, as well as other configuration options (for security, and regarding the mechanisms that Vazels uses for internal communication). They then have to use a separate command line client to specify the workload for the groups, and they must then copy files into a newly created directory structure, matching up their chosen group names to the right folders. The conclusion is that the user is forced to spend time engaged in setup tasks, which can scare users away from trying out a very capable and helpful system.
The difficulty in setting up projects and visualising the concepts involved provides a large part of the motivation for this GUI project. We aim to provide the user with an intuitive interface to set up an experiment, without having to type on the command line, or deal with setting up tools such as Siena or an RMI Registry on remote machines (both of which are currently involved in the setup process). We also aim to provide a useful way for the user to gather results about their running experiment, and monitor its status in real time.

Appendix 3 - Draft Schedule from Project Report 1

The time constraints on this project make it worth us setting out weekly goals at this early stage. We will break down the workload into weekly milestones, hoping to deliver a final iteration on 18/12/10. The description with each milestone details what work will be undertaken in the week leading up to it. To summarise:

- We plan 5 iterations in total over a 7 week period,
- We aim to complete functional requirements by 13/12/10 (and to have realised our key requirements by 6/12/10).
- Major requirements should be finalised by 08/11/10, with major changes permitted after that only with considerable justification.
- No new requirements will be considered within the scope of the project after 13/12/10. By producing a number of iterations before this point, and getting regular feedback from the client, we hope that there should not be any such requirements by this point.

25/10/10 – Research & Concept Exploration

- Preliminary research of the Vazels project, and a good understanding of the concepts involved by all members of the group. A good understanding of the motivation and some of the requirements for a GUI to the system.

01/11/10 – Key Requirements

- Key project requirements finalised with the project supervisor.
- **1st iteration complete.** A basic “setup” solution, to help the user administer groups and workloads. The user will still have to manually provide the system with required files (such as Actors, and the SUE).
- A pause for revision and reflection on issues so far.

08/11/10 – User stories, and server-side management of files

- Key user stories completed, and discussed with client in some detail.
- Major changes to requirements will be considered only with adequate justification.
- Implement a system to allow the user to fulfil the security requirements of the Vazels system easily (or removing the need from the user, and automatically generating a new SSH key on a per-project basis).
- Server can now build an abstract model of Groups, Actors, and Hosts. The user can provide the necessary files via the GUI, and no longer has to manually configure server-side file locations, etc. When the user wishes to start an experiment, the Web interface will manage the back-end tasks for them.

15/11/10 – Setup complete: leaving the command line behind

- **2nd iteration complete.** Functional requirements of the setup phase are now met (changes to requirements notwithstanding).
- Crucially, the user no longer has to touch a command line until they are ready to deploy their test onto the machines taking part in the test.

22/11/10 – Feedback and data display

- The user can now retrieve test data from the server (via AJAX request, meaning that they do not have to refresh the web page). A basic display of this data should be implemented. If reasonable within the time-frame, data will be presented on either a static graph, or a time-line. *[Extension: Depending on how difficult this actually proves to be to implement, we will decide on the feasibility of a more rich data display at this point, as well as a time-frame for putting it into practise]*
- Take this opportunity to explore the feasibility of some of our other extension tasks. Specifically, investigate the performance overheads (and implementation details) for the "Prefabricated tests" mentioned previously (e.g. monitoring CPU usage).

29/11/10 – Finish key requirements

- **3rd iteration ready.** Server security model fully implemented. Functional requirements of running phase ready. User can run tests and display data collected from them. All key requirements should now be a part of the project. **Feature Freeze.** No new features will be started after this point until after the next milestone.

6/12/10 – Client feedback and requirement revision

- Finish implementing incomplete tasks from 3rd iteration. Revise project requirements, and decide on feasibility of extensions.
- If finishing features from 3rd iteration and bug testing goes better than expected, begin to investigate possibilities for implementing extensions.
- Client-centred testing. Ensure the client is satisfied with work so far, and address any concerns.

13/12/10 – Document, test, and finalize

- **4th iteration complete.** All functional requirements now realised. **Re-freeze features.**

- Ensure good test coverage. In cases where it is difficult to write tests (e.g. UI elements), find other ways to validate code.
- Ensure good documentation coverage.
- Ensure the Wiki forms a complete and useful picture of the development process – and of the product as it currently stands. Ensure information is accurate and up-to-date.
- Ideally, a new team of developers should be able to continue our work after we finish – given access to our code repositories, our documentation, and our Wiki.

18/12/10 – Final Iteration

- **5th iteration complete.** The quick turnaround between this iteration and the previous one should be possible by this point, where all code changes are bug-fixes.
- No remaining “Blocking” or “Major” bugs.