Executive Summary

Reef - a graphical front-end to the Vazels platform

We present Reef - a graphical user interface to the Vazels distributed testing system. Vazels was a Masters project by Angel Dzhigarov, and in his own words: '*It is a tool that helps automating experiments with distributed systems. It provides "the infrastructure" to the developers, so that the latter can concentrate on the experiment specification, rather than how to execute the experiment.'* [1]
Vazels is built to run on a Unix-based system, and currently provides only a command-line interface. Reef provides access to basic functionality of Vazels using a more intuitive GUI, provided via a web interface, from any web-enabled platform.

Technical Description

The final deliverable of the project provides an interface through which a user can perform all tasks involved in setting up a basic experiment for running on the Vazels system. The main achievements of Reef are allowing platform agnostic, remote and more user-friendly access to Vazels. Reef is accessible remotely from any operating system without needing physical access to the system running Vazels. Reef reduces the use of the command-line, providing most of Vazels' functionality through the interface. Reef will additionally save configurations so experiments can be repeated without the need to enter the setup stage again.

Reef is delivered by using a Web Interface, with the architecture shown in Fig. 1.

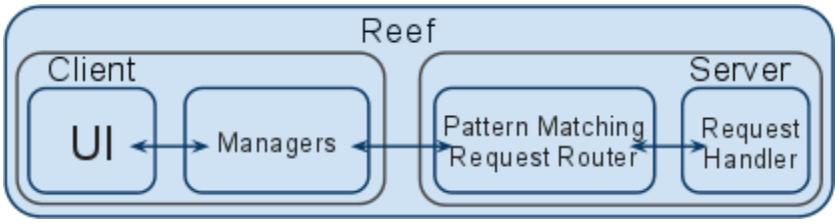

Fig.1 Reef Main Components Diagram

The client communicates user input to the server through AJAX requests to the server's RESTful back-end.

Implementation

- **Google Web Toolkit** - allowed us to write a rich, dynamic JavaScript application for the client-facing section of the project in Java, which was then compiled into browser-optimised ECMA-script.
    - Allowed us to carry out better software design,
    - Delivered the powerful benefits of a staticly typed language in a familiar development environment (Eclipse).
- **RESTlite** - a barebones REST server implementation, licensed under the LGPL, allowed us to quickly and easily implement new features while avoiding writing complex server-side code. The Reef Server is built on this system, relying on it to match requests to the proper handlers.

[1] From the front page of http://vazels.org, the project's homepage

- **Protocol Buffers** - Google's cross-platform serialization tools used by Vazels to store and transmit data
- **Google Charts API** - presented an easy-to-implement way to output data, tightly integrated with Google Web Toolkit.

Software Engineering Issues

Technical challenges:

- Running Vazels on the command-line - an unreliable and difficult to configure interface justified our project, but meant progress was slow while we tried to learn the tool.
- Unfamiliar tools created bottlenecks within the group until all members could function independently
- Migration to Linux for a standardised development system compatible with Vazels slowed progress (using Google Web Toolkit made it difficult to use DoC lab computers).
- Git repository temporarily unavailable - switched repository hosts half way through the project.

Collaboration challenges:

- Differing understandings - group members had differing levels of familiarity with the technologies in use, and differing competencies when it came to learning them. Progress was at vastly differing rates for these reasons, and a failure of the management style to address this problem held us back. Documentation style was aimed at overcoming this problem, but with limited success.
- Differing commitments - some group members had extremely full academic or professional schedules during the Autumn term, whereas others had more time free to work on the project. A minimum workload was agreed, but sometimes not met (total workload was adequate, but balancing of workloads was not consistent).

Management tools:

- Git version control, Trac and GitHub issue tracking
- Doodle and Google Calendars for managing 5 schedules.

Validation & Conclusions

- 80% of key requirements were fulfilled, 2 extensions partially addressed.
- Milestones were almost universally delayed, with work not finishing on the project until 10 Jan (instead of 19 Dec as planned).
- Around 10,000 lines of code produced, around 3,500 in Python (server-side code), and 6,500 in Java (client-side code).
- The conclusion of the project is a usable front-end, delivered through a web browser (Firefox or Opera), providing the bulk of Vazels' basic functionality.
- The project as a whole is documented with the intention that developers could pick up this project in future. Reef provides a reasonable basis for on-going work, with a great deal of time spent focusing on code clarity, as well as correctness.