Quokka: Visualising Interactions of Enterprise Software Environment Emulators

Cameron Hine, Jean-Guy Schneider, Jun Han
Swinburne University of Technology
Faculty of ICT
Melbourne, Australia
{chine,jschneider,jhan}@swin.edu.au

Steve Versteeg
CA Labs
Melbourne, Australia
steve.versteeg@ca.com

ABSTRACT

Enterprise software systems operate in large-scale, heterogeneous, distributed environments which makes assessment of non-functional properties, such as scalability and robustness, of those systems particularly challenging. Enterprise environment emulators can provide test-beds representative of real environments using only a few physical hosts thereby allowing assessment of the non-functional properties of enterprise software systems. To date, analysing outcomes of these tests has been an ad hoc and somewhat tedious affair; largely based on manual and/or script-assisted inspection of interaction logs. Quokka visualises emulations significantly aiding analysis and comprehension. Emulated interactions can be viewed live (in real-time) as well as be replayed at a later stage, furthermore, basic charts are used to aggregate and summarise emulations, helping to identify performance and scalability issues.

Categories and Subject Descriptors
D.2 [Software]: Software Engineering; D.2.5 [Software Engineering]: Testing and Debugging—testing tools

General Terms
Measurement, Performance, Verification

Keywords
Enterprise Software Emulation, Service Virtualization, Scalability, Visualisation

1. INTRODUCTION

Enterprise software systems operate in large-scale, heterogeneous, distributed environments. These environments typically contain dozens of different types of endpoint systems hosted by an eclectic range of infrastructure: legacy mainframes, clusters, servers, PCs, right through to virtual infrastructure provisioned in a cloud. This diversity is reflected in the range of (application-layer) protocols used to communicate between endpoints. Magnifying the complexities inherent in distributed environments with such heterogeneity is the scale enterprise software systems encounter in production deployments: tens-of-thousands of endpoints can be operating at a given point in time.

Developing software systems which adequately handle the heterogeneity and scale encountered in enterprise environments is a significant technical challenge. Assessing non-functional properties, such as scalability and robustness, is particularly troublesome for quality assurance (QA) teams. Test-bed environments used for such assessments should reflect (even exceed) the heterogeneity and scale expected in industrial deployments.

Enterprise software environment emulators, such as Kaluta [7], provide test-bed environments approximating interaction behaviour and characteristics of real enterprise environments. Kaluta’s behaviour is based on execution of lightweight models specifying an approximation of the interaction behaviour of a real (basis) endpoint. Resource overhead for model execution is nominal so that a single machine can emulate thousands of endpoints simultaneously. QA teams can use Kaluta to produce suites of test-bed environments in which the run-time properties of various enterprise software systems and components may be assessed.

To date, our work in this area has focused on manipulating communications between networked machines so that one will appear and act as if it is an enterprise environment from the perspective of the other. Presenting these underlying communications to human users in appropriate forms is essential for comprehension and diagnosis of potential issues in software systems under test.

In this work we introduce Quokka: a tool which visualises interactions of Kaluta emulations. It is intended to aid comprehension and analysis of emulations in general. Quokka provides a high-level dynamic view of a live emulation allowing users to see at-a-glance what interaction is currently happening and on which endpoints these interactions are occurring. Interactions occurring in previous emulations are also presented in the dynamic visualisation and can be stepped backward and forward to identify points of unexpected interaction behaviour. This ability to immediately see the interaction behaviour of an emulator and a system under investigation, at a comfortable pace, provides great benefit to a QA team attempting to comprehend system be-
haviour. Additionally, Quokka provides a few basic charts which aggregate and summarise the interaction characteristics of an emulation. These charts allow QA personnel to more easily identify scalability and performance issues.

This paper is organised as follows: Section 2 provides a brief description of Kaluta, the enterprise software environment emulator which Quokka visualises. This provides the necessary technical context for the discussion of Quokka appearing in Section 3. We discuss related work in Section 4 and wrap up with conclusions and future work in Section 5.

2. KALUTA

Kaluta is a prototype enterprise software environment emulator capable of representing large-scale, heterogeneous environments using a single physical host. Models capable of expressing interaction behaviour of endpoints allow users to specify behaviour of particular endpoint types, such as LDAP directory servers, SOAP based Web Services, and BitTorrent peers. Configuration is achieved by itemising the required emulated endpoints in terms of these specifications along with any parameter values required for their instantiation. Kaluta can execute many thousands of these interaction models concurrently thereby appearing to be, and behaving similarly to, a real large-scale enterprise software environment.

Kaluta’s architecture, depicted in Figure 1, consists of two main components: the engine and the network interface. The engine’s responsibility is concurrent execution of as many endpoint model instances as possible. The network interface’s responsibility is communication between external software systems and the appropriate interaction models being executed by the engine. Apache Thrift [1] services facilitate inter-component communication and also provide an interface for components to monitor the activity of a running emulation. Everything is tied together by a main Kaluta class which handles configuration and initiation of the components, as well as enabling the monitoring facilities utilised by Quokka for visualisation of live emulations. We refer the reader to [7], and http://quoll.ict.swin.edu.au/kaluta.html for a more thorough treatment of Kaluta’s design and implementation.

Kaluta improves upon our earlier emulator prototypes Reacto and Reac2o [8] by allowing much greater flexibility. Emulated endpoints can be specified using different concrete model stacks and the engine itself can be implemented using whichever general purpose language is most suitable. Additionally, Kaluta’s monitoring interface allows not only the visualisations illustrated by Quokka, but also enables a range of future enhancements based on live emulation events.

3. QUOKKA

Quokka provides dynamic and static visualisations of live and previously recorded Kaluta emulations. In this section we briefly describe its features, implementation, as well as the main benefits gained through its use.

The dynamic visualisation depicts emulated endpoints and their interactions with external software systems as emulation progresses. External systems are presented in the centre of the dynamic view as circles. Emulated endpoints are depicted as rectangles and are arranged in a series of concentric rings encircling the inner external systems. Open communication channels are represented as circles appearing within the bounds of the rectangle of the corresponding endpoint. Endpoints may have numerous channels open simultaneously which is depicted by having numerous such inner circles: one for each open channel. As channels are closed the corresponding circles disappear.

Message transmission is presented in two ways. Firstly, message direction is depicted as an equilateral triangle drawn within the associated channel. An upward pointing triangle represents a message transmission, such as a response to an earlier request, and downward for messages coming in to the channel, such as a request arriving from an external system. The second presentation of message transmission is presented as diamonds which travel (transmission) along an arrow connecting the corresponding emulated endpoints and external systems.
Figure 2: Quokka Dynamic View of an Emulation involving 48 endpoints and a single external system. Six endpoints contain open channels with seven channels in total. Three channels have recently sent messages and two have recently received them. There are two open channels without any recent activity and one message in transit towards the external system.

Control over the dynamic visualisation is provided in the form common to media players: buttons for play/pause, step forward and backward as well as sliders for progress as well as playback speed. When visualising a live emulation, these controls act similarly to TiVo and other similar broadcast TV recording systems. Pausing dynamic visualisation of a live emulation does not pause the emulation itself, rather it pauses the playback of the live emulation and buffers events occurring in the interim. Once synchronicity between the visualisation and live emulation is broken all features of the replay based dynamic view become available, i.e. both representations of message transmission are presented rather than just direction.

Oftentimes Kaluta will be emulating many thousands of endpoints simultaneously. Visualising environments of this scale is an issue. There is simply not enough screen real estate to display such environments at the level of detail shown in Figure 2. Quokka provides two features to help handle large scale environments in the dynamic view. Firstly, zooming and panning facilities allow the user to focus attention on particular regions of the visualisation, zooming in to provide more detail, and out to provide less. Secondly, the endpoints which are visualised can be filtered by IP addresses and subnets. A filter of “10.0.1.*” for instance will only visualise emulated endpoints with IP addresses beginning with “10.0.1.”.

Complementing dynamic visualisation, Quokka provides a few rudimentary charts summarising emulation interactions. These include bar, line and pie charts outlining emulations aspects such as: distribution of message transmission and receptions among endpoints, distribution of message types (shapes), as well as performance data such as latency and throughput of endpoints.

Figure 3: Pie chart illustrating distribution of input and output messages amongst ten emulated endpoints.

Quokka can be initialised by either providing a log file describing the interactions of a previous emulation, or by connecting to a live Kaluta emulation through the provided monitoring service. In either case the result is a series of calls to Quokka’s event function which takes an interaction event as a parameter and triggers the corresponding dynamic visualisation. In replay mode, Quokka parses a log file producing a series of events which are fed into the event function at the rate controlled by the playback speed slider. In live mode, on the other hand, the event function is invoked directly by Kaluta which has been registered through the monitoring interface, Quokka’s dynamic visualisation simply tries to maintain this pace. To implement the TiVo like controls of live emulation visualisation events are recorded in a log which is used as the basis of the dynamic visualisation once moving backwards through time.

Three libraries are used by Quokka to facilitate visualisation and inter-component communication. The dynamic visualisation of events is achieved using the yFiles [3] for Java graph diagramming library developed by yWorks. JFreeChart [2] provides the charting capabilities and, as mentioned earlier, Apache Thrift [1] is used as the monitoring interface allowing communication between Quokka and Kaluta.

There are many benefits to having emulations visualised by Quokka. In live mode dynamic visualisation provides a real-time holistic view of running emulations. In this view it is clear which endpoints are being interacted with frequently and which are not and also gives an indication of how many endpoints an external system tends to interact with simultaneously. In replay mode dynamic visualisation provides a way to step through interactions allowing close inspection of unexpected sequences helping to identify issues in the external systems behaviour as well as potential issues in the emulated endpoint specifications themselves. This visual representation of interactions is preferable to sifting through text logs as all the important concepts: channel, message,
endpoint, are presented together and retain/highlight their conceptual relationships to one another. Finally, the various charting options provides a valuable aggregate perspective on distribution on interactions and the performance of external systems and the emulator itself.

4. RELATED WORK

Quokka provides elementary dynamic and static visualisations of time-dependent, event based data describing the interactions of enterprise software environment emulators. Müller and Schumann [10] provide an exceptional overview and taxonomy of visualisation methods for this time-series and in particular event-driven data. Questions regarding unknown temporal data such as those stated by MacEachren[9] are also relevant to our visualisation efforts, for example and phrased in the terminology of our own context: when do messages exist, how long are they present until they are consumed, what if any cyclic behaviour occurs in the messages themselves and emulated endpoints, do certain interaction patterns affect performance?

There have been significant efforts in the area of software performance engineering visualising potential architectures of large, complex software systems and using these visualisations to help present expected performance. Grundy and Hosking investigated the potential for static and dynamic visualisation of software architectures[6] eventually leading to the SoftArch/MTE [5] and also to integration with open source CASE tools in Argo/MTE [4]. A major difference between their work and our own is the former’s focus on architecture and design of a prospective system. Our work on emulation assumes that these deployment environments, although variable, are fixed and mostly beyond the control of the developer. Nevertheless there is opportunity to incorporate relevant visualisations from the performance engineering field into our own work.

5. CONCLUSIONS AND FUTURE WORK

Quokka is a visualisation tool for enterprise software environment emulators which aids comprehension of running, and previously run, emulations as well as providing better facilities to analyse its behaviour and results. Previously, Kaluta’s sole means to investigate and analyse the behaviour of an emulation was through manual inspection of its interaction logs. This made it difficult to form a holistic picture of what was occurring during a live emulation, and in general, made analysis of the results of an emulation quite tedious. The dynamic mode of Quokka delivers a visual, holistic view of emulation interaction behaviour both while it is running and, also, allows stepping and previewing of previous emulations allowing users to examine specific subsequences of interactions of interest. Furthermore, the charts provided by Quokka aggregate and summarise emulation interactions allowing analysis of the scalability and performance of external systems, as well as the emulator itself.

There is potential to extend these visualisations further in future work. We are particularly interested in forming a relationship between the interaction modelled executed by Kaluta and the visual representation of those models. Success in this area could lead to the visualisation growing to encompass configuration and specification aspects, allowing emulated endpoints to be defined in a visual manner. Additionally, there are much more advanced ways to visualise time-dependent data such as that collected by an emulator. In the future we would like to explore axis-based visualisations such as TimeWheels and MultiCombs [11] for representing the endpoint interactions over the course of an emulation.

6. ACKNOWLEDGEMENTS

This work is part of the ARC Linkage Project LP100100-622, supported by the Australian Research Council and CA Technologies. We would like to thank CA Labs for their ongoing support and contributions to this work. We would also like to thank Thai Nguyen for his considerable contributions to the Quokka development effort.

The most recent release of Quokka and Kaluta are available for download from the project web site: http://quoll.ict.swin.edu.au/releases/kaluta.tgz.

7. REFERENCES