The following paper was originally published in the
Proceedings of the Fifth Annual Tcl/Tk Workshop
Boston, Massachusetts, July 1997

Building a Notification Infrastructure using Tcl, Zephyr, and Linda

W. Scott Meeks
The Open Group Research Institute
Cambridge, MA

Building a Notification Infrastructure Using Tcl, Zephyr, and Linda

W. Scott Meeks

The Open Group Research Institute, 11 Cambridge Center, Cambridge MA 02142, USA

Poster Abstract

This poster describes the design and implementation of a notification system that can be integrated with other Web tools and applications. A Tcl-based toolkit using the MIT Zephyr Notification System was built to allow applications to notify each other or a user of relevant changes in information. A blackboard service was built using the notification toolkit and a server for the Linda language. This service is used to selectively store notifications for later use by applications. The use and effectiveness of the toolkit and service were demonstrated in an application for conducting a distributed guided tour of individual Web pages, and by integrating notifications from our Mediator service into our HistoryGraph application.

We tested our technology in two sample applications. The first was the GroupWalk program. It was written from scratch in Tcl to help in providing a "guided tour" of individual Web pages. This particular application provides a good example of the sort of cooperation and coordination we want to be able to support using the notification system.

When run, the program can take on one of two roles. In the first role, it watches the tour leader's browser and sends out a notification whenever the current URL changes. This information is also saved in the blackboard. In the second role, it receives these notifications and tells a tour member's browser to fetch the correct URL. Should a tour member arrive late for the tour, they can "catch up" by automatically retrieving (using information stored in the blackboard) all pages up to and including the current page in the order accessed (thus enabling sensible usage of the browser's *Back* button).

The second application was to integrate our Mediator [Mediator] service with our HistoryGraph [Hirsch97] application. The Mediator service provides controlled access to a set of structured documents for a team of users. It runs as an HTML forms-based CGI application. The HistoryGraph application tracks and displays browser navigation history in an easily manipulable form. It is written in Tcl/Tk. The main HistoryGraph display consists of nodes representing visited URLs and links representing the order in which URLs were visited.

Our primary goal was to provide a simple and extensible system for sending and receiving asynchronous event notifications between Web tools as well as users. We wanted to support dynamically changing groups of users and tools in a scaleable manner, and also wanted users and tools to only receive notifications of interest to them, so we chose a system that supports multicasting in something of a publish and subscribe fashion.

However, we also wanted to provide for persistent storage of certain notification information than can be easily shared among our tools. In essence, we wanted something like a blackboard that could serve both as a message repository and as a rendezvous service. Based on the mobility of both users and services (such as the blackboard) running on the network we desired that notifications be delivered independent of the actual location of the entity r eceiving it.

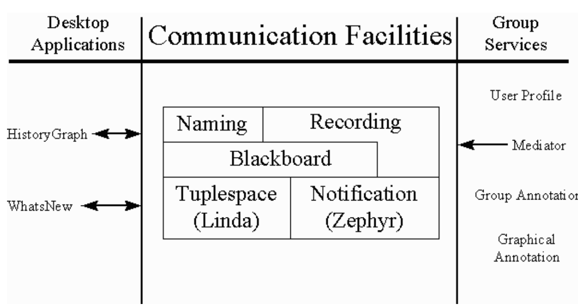

Figure 1: The notification software architecture and its use.

Figure 1 shows the relationship and structure of the notification software and our other applications and services. For the infrastructure of the notification system, we chose the MIT Zephyr Notification System [DelaFera87] and for the persistent store a server implementing the Linda[Schoenfeldinger95] language. We considered implementing our own infrastructure, perhaps by using Tcl-DP, but it made sense to use Zephyr since it provided a large body of existing code including the servers, host managers, zwrite program and the library API for accessing all of this. Likewise, we could have implemented the blackboard directly in Tcl, perhaps using ndbm, but since we already had the server which provided the persistent store as well as the Linda synchronization semantics, it made sense to use that.

In the Zephyr system, "multiple, redundant Zephyr services provide basic routing, queuing, and dispatching services to clients that communicate with the Zephyr client library."[DelaFera87] Essentially, Zephyr provides a location independent publish and subscribe model with a three-level notice classification scheme and optional authentication. Since these are all features we wanted to exploit and the Zephyr software is freely available, it made sense for us to incorporate it rather than reproduce it.

Linda is a small set of functions that can be added to a language to make a parallel dialect of that language. The functions are used to access a *tuplespace* which contains tuples of arbitrary information. We could have chosen other database solutions, but the Linda server was already written, was freely available, provided parallel programming and synchronization semantics that we hoped eventually to exploit, and was already being used in other Web applications.[Schoenfeldinger95]

There were a number of reasons we wanted to build a toolkit in Tcl, but primarily we liked the rapid prototyping capabilities and it provided compatibility with a number of our other applications and tools. However, we were particularly attracted by some of the new features in Tcl7.5.

Because of the different needs of our applications we wound up using at least half a dozen different versions of Tcl/Tk (including versions like Tcl-DP and IncrTcl as well as our own custom shells) amongst the various components of our software. The complexity involved made interoperability and maintenance difficult. Between the load command and packages, sockets, and the clock function in Tcl7.5, we were able to eliminate all but one specialized shell for the HistoryGraph application. After our experience with building the Zephyr package, we're fairly certain we could eventually eliminate that one.

The notification toolkit, blackboard, Mediator, and HistoryGraph are available at http://www.osf.org/www/waiba/webwarev2_1/. The GroupWalk associate was a very early prototype and a version for external release has not been pr oduced.

Acknowledgments

This research was supported in part by the Defense Advanced Research Projects Agency (DARPA) under the contract number F19628-95-C-0042. The views and conclusion contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

My thanks go to Charles Brooks and Frederick Hirsch for critical readings of the original paper and this abstract.

References

[DelaFera87] C. A. DelaFera, *The Zephyr Notification System*, MIT Project Athena documents, Massachusetts Institute of Technology, Cambridge, MA, USA, 1987.

[Hirsch97] F. Hirsch, W. S. Meeks, C. Brooks. *Creating Custom Graphical Web Views Based on User Browsing History*. http://www.osf.org/www/waiba/papers/www6/hg.html

[Mediator] *Distributed Authoring with the Mediator*. http://www.osf.org/RI/PubProjPgs/med-one.html

[Ousterhout94] J. K. Ousterhout, Tcl and the Tk Toolkit, *Addison-Wesley*, 1994.

[Schoenfeldinger95] W. Schoenfeldinger, WWW Meets Linda: Linda for Global WWW-Based Transaction Processing Systems. World Wide Web Journal, Issue 1: Conference Proceedings, Fourth International World Wide Web Conference, O'Reilly and Associates, December 1995. http://www.w3.org/pub/WWW/Journal/1/schoen.174/paper/174.html

[Welch95] B. B. Welch, Practical Programming in Tcl and Tk, *Prentice Hall*, 1995.