TKQML: A Scripting Tool for Building Agents

R. Scott Cost, Ian Soboroff, Jeegar Lakhani, Tim Finin, Ethan Miller, and Charles Nicholas

Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, Maryland 21250
{rcost1, ian, jlakha1, finin, elm, nicholas}@cs.umbc.edu

Abstract. Tcl/Tk is an attractive language for the design of intelligent agents because it allows the quick construction of prototypes and user interfaces; new scripts can easily be bound at runtime to respond to events; and execution state is encapsulated by the interpreter, which helps in agent migration. However, a system of intelligent agents must share a common language for communicating requests and knowledge. We have integrated KQML (Knowledge Query Manipulation Language), one such standard language, into Tcl/Tk. The resulting system, called TKQML, provides several benefits to those building intelligent agent systems. First, TKQML allows easy integration of existing tools which have Tcl/Tk interfaces with an agent system by using Tcl to move information between KQML and the application. Second, TKQML is an excellent language with which to build agents, allowing on-the-fly specification of message handlers and construction of graphical interfaces. This paper describes the implementation of TKQML, and discusses its use in our intelligent agent system for information retrieval.

1 Introduction

In the past few years, the explosive growth of the Internet has allowed the construction of "virtual" systems containing hundreds or thousands of individual, relatively inexpensive computers. The agent paradigm is well-suited for this environment because it is based on distributed autonomous computation. Although the definition of a software agent varies widely, some common features are present in most definitions of agents. Agents should be autonomous, operating independently of their creators. Agents should have the ability to move freely about the Internet. Agents should be able to adapt readily to new information and changes in their environment. Finally, agents should be able to communicate at a high level, in order to facilitate coordination and cooperation among groups of agents. These aspects of agency provide a dynamic framework for the design of distributed systems.

Tcl [7] is an ideal language with which to build agents, because scripts written in Tcl may be used on any machine that can run Tcl, and because the Tcl language environment itself is highly portable. Additionally, Tcl/Tk greatly facilitates rapid prototyping and quick development of small applications.

Appeared in the 1997 Conference on Agent Theories and Agent Languages (ATAL97), Newport, RI, July 1997, pages 339–343.

We present TKQML, the integration of an agent communication language, KQML [2] (Knowledge Query Manipulation Language) into Tcl/Tk. TKQML can be used to build KQML-speaking agents that run within a TKQML shell. TKQML can also be used to bind together diverse applications into a distributed framework, using KQML as a communication language. Tcl's embeddable nature allows one to easily add agent communication facilities to existing code. As such, TKQML can be used to enhance the functionality of new or existing systems built using a Tcl framework, by allowing easy integration with agent-based systems.

2 Background

KQML is a language for general agent communication. It was developed as part of the Knowledge Sharing Effort [8], a DARPA project exploring agent communication and knowledge reuse. KQML is a language based on speech acts, such as "tell", "ask", and "deny", which describe the nature of a message without reference to its content. Agents communicate application-specific information embedded in general, higher-level KQML messages. A comprehensive semantics [5,6] for KQML outlines protocols for agent "conversation." Additionally, most implementations provide facilities for message handling, agent naming and resource brokering.

Problems of software agency have not been neglected within the Tcl community. Existing Tcl-based solutions to agent issues, such as AgentTcl [3] and Tacoma [4] have emphasized security and mobility, but fall short with respect to communication. AgentTcl agents, using TCP/IP, exchange bytes strings which have no predefined syntax. In Tacoma, agents must meet in order to communicate. Other projects, such as Tcl-DP [10] provide excellent packages for communication, but lack sufficiently flexible support for higher level languages. TKQML bridges this gap.

3 TKQML

TKQML is an adapter for KATS, a C-based implementation of KQML. It allows Tcl scripts to use existing support for KQML. TKQML usage mirrors that of the KATS API, facilitating easy synchronization with the C version, traditionally the reference implementation.

TKQML is implemented in C as a Tcl application which presents one Tcl command: `kqml`. The application can be thought of as having two components, the KQML library extensions and the message handler.

3.1 Library Calls

In extending the library, our primary goal was to reflect as closely as possible the usage of the KATS API, through the `kqml` command. When the `kqml` command is called, parameters are first translated into an intermediate representation. This allows the representation of complex types at the script level.

Next, a call is made to the appropriate function in KATS with the translated parameters. Finally, the return value and resultant parameters are translated back into string representations and, if appropriate, inserted back into the Tcl interpreter environment, simulating call-by-reference.

In TKQML, complex or large objects are kept in memory, and references to these objects are passed up to the Tcl level. These references are merely string representations of the pointers to memory. Script level routines use the `kqml` commands to access these objects, and are responsible for disposing of them (with a deallocation routine). Since memory pointers have a natural translation to and from string references, no additional memory management is required on the part of TKQML. Note that code could easily be inserted to track allocated memory, facilitating memory management mechanisms.

3.2 Message Handling

An agent will typically deal with each type of incoming message in a different way. KATS provides a pre-emptive solution with the use of a message handler and a router. The router, a separate process, handles the receipt, queuing and transmission of all message traffic. The agent registers message handling routines with a message manager, which is embedded in the agent code. Upon receipt of a message, the router signals the manager, interrupting the current processing. The manager then invokes a routine appropriate for the message.

This process has a natural implementation in Tcl which is much more flexible. Handlers, like the rest of the agent, are written as Tcl scripts, and are registered as strings (or string references to files). Unlike their C counterparts, Tcl handlers do not need to be static objects, compiled and linked before execution. Thus, Tcl handlers can be updated or replaced at any time during execution, making Tcl based agents much more adaptable.

4 CARROT - TKMQL in Action

The CARROT project (Co-operative Agent-based Routing and Retrieval of Text, formerly CAFE) is an ongoing effort at UMBC to develop a distributed architecture for text-based information retrieval [1] that has served as a testbed for TKQML. This project employs a brokered environment of clients and servers. Users make queries through a World-Wide Web-based client, which are routed by a broker agent to an appropriate information source. The broker makes these decisions by gathering information, or metadata, from each source, and deciding which database the query most resembles. A ranked set of results is returned to the client.

A heterogeneous set of text-indexing engines, such as Telltale [9] and mg [11] manage large sets of text data. These engines have been augmented into agents with TKQML. The broker agent communicates transparently with these information servers via KQML. All components consist of C/C++ applications bound to TKQML with a Tcl/Tk shell. One agent, the Agent Control Agent (ACA) is

written entirely as a TKQML script. Our experience with CARROT has shown that TKQML can facilitate quick prototyping and rapid development of agents and their GUIs, reducing the time necessary to build large agent-based systems.

4.1 The Agent Control Agent

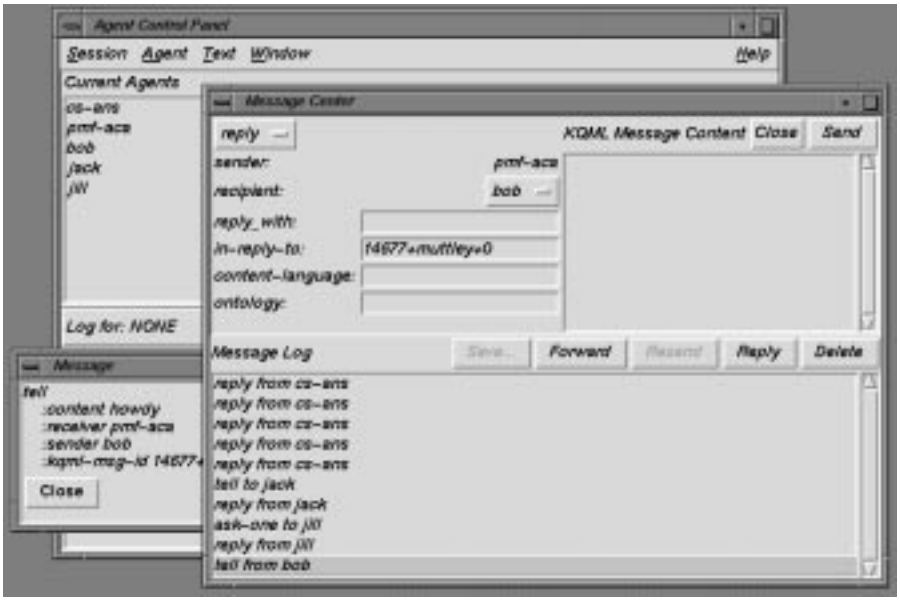

Fig. 1. Agent Control Agent

The Agent Control Agent (ACA) was designed to help manage and coordinate the activities of several agents in the CARROT project. It is built entirely from a TKQML shell, and presents a Tk interface panel for controlling agents (see Figure 1).

The ACA provides many facilities for directing agents. It can start or stop agents, and connect to agents which are already running. Arbitrary KQML messages can be sent to agents from the ACA panel. The ACA can monitor the log files of individual agents. A list of currently-known agents is displayed, and appropriate actions apply to the currently highlighted agent in the list.

The ACA, as indicated by its name, is itself a KQML-speaking agent. Almost all control functions are accomplished through KQML messages sent and received from other agents. This makes the ACA independent of individual agent architectures.

The Agent Control Agent has proved a valuable asset to orchestrating groups of agents. It is helpful for organizing demos, organizing and displaying agent

activity, and debugging agent sessions. Moreover, the flexibility of Tk has allowed us to easily prototype and integrate new functionality on demand.

5 Summary and Conclusions

Both Tcl and KQML are powerful tools in the development of agent-based systems. TKQML combines the two, making it possible to benefit from both the light weight and portability of Tcl scripts and the high-level communication support of KQML with one package. We feel that its power, simplicity and potential for future development make it an ideal platform for the development of agent-based systems.

Resources for TKQML v1.0 can be found at: *http://kqml.org/tkqml/*.

References

1. Grace Crowder and Charles Nicholas. Resource selection in CAFE: An architecture for network information retrieval. In *Proceedings of the Network Information Retrieval Workshop, SIGIR 96*, August 1996.
2. Tim Finin, Yannis Labrou, and James Mayfield. *Software Agents*, chapter KQML as an agent communication language. MIT Press, 1997.
3. Robert Gray. Agent Tcl: A flexible and secure mobile-agent system. In *The Fourth Annual Tcl/Tk Workshop Proceedings*. The USENIX Association, 1996.
4. Dag Johansen, Robbert van Renesse, and Fred B. Schneider. An introduction to the TACOMA distributed system. Technical report, University of Tromso, June 1995.
5. Yannis Labrou. *Semantics for an Agent Communication Language*. PhD thesis, University of Maryland Baltimore County, 1996.
6. Yannis Labrou and Tim Finin. Semantics and conversations for an agent communication language. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*. Morgan Kaufman, August 1997.
7. John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.
8. Ramesh S. Patil, Richard E. Fikes, Peter F. Patel-Schneider, Don McKay, Tim Finin, Thomas Gruber, and Robert Neches. The DARPA knowledge sharing effort: Progress report. In Bernhard Nebeld, Charles Rich, and William Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR92)*. Morgan Kaufman, 1992.
9. Claudia Pearce and Charles Nicholas. TELLTALE: Experiments in a dynamic hypertext environment for degraded and multilingual data. *Journal of the American Society for Information Science*, June 1994.
10. Brian C. Smith, Lawrence A. Rowe, and Stephen C. Yen. Tcl distributed programming. In *Proceedings of the 1993 Tcl/Tk Workshop*. The USENIX Association, June 1993.
11. Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, 1994.