Pioneering Technologies for a Better Internet

Cs3, Inc.

5777 W. Century Blvd.
Suite 1185
Los Angeles, CA 90045-5600

Phone: 310-337-3013
Fax: 310-337-3012
Email: info@cs3-inc.com

FLEA OVERVIEW

FLEA: Formal Language for Expressing Assumptions
Language Description

INTRODUCTION

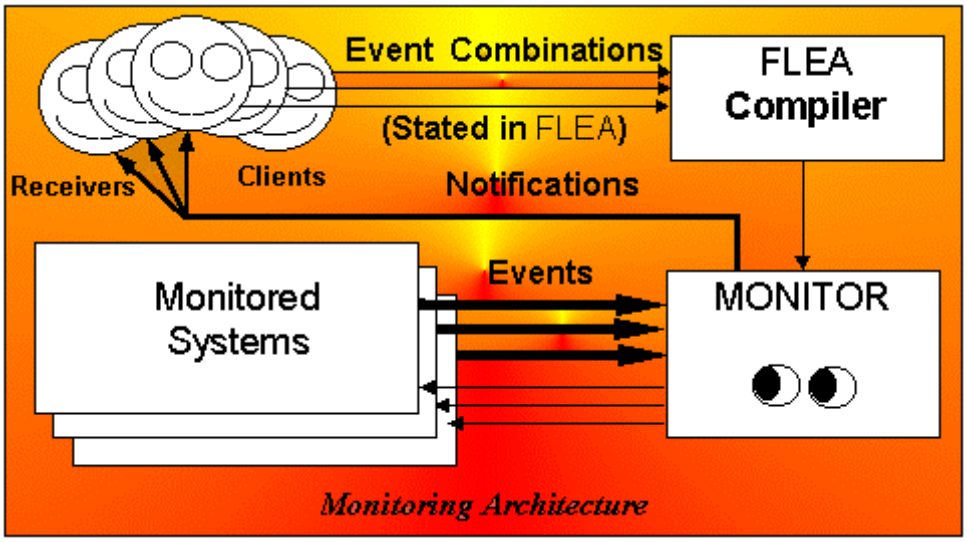

Figure1 : Monitoring Architecture

The overall architecture for monitoring is shown in *Figure 1*. The Monitored System, in the lower left of the figure, is the source of events. The user, sketched as a face in the upper left, is a human (or another computer system) with an interest in being notified of certain combinations of events of the monitored system. In principle, the user could observe the monitored system's events directly, and perform all the reasoning to deduce occurrencies of the combinations of events in which he/she/it is interested. The whole purpose of the FLEA compiler and monitor is to take over the burden of this reasoning. FLEA is a small language particularly suited to the expression of event combinations, from which the run-time code to monitor those assumptions is automatically generated.

> *FLEA stands for a Formal Language for Expressing Assumptions. Our original motivation was to use monitoring to notify a system's users / administrators / designers whenever assumptions underlying use of that system are violated [Fickas & Feather 1995]. There can be a multitude of uses for such information, for example, alerting users when they are using a system in a manner for which it is not intended, alerting administrators of changes in typical usage patters (to which they might wish to respond by reconfiguring the system and/or its environment), or alerting designers of the need/opportunity to extend their systems in new ways which they had not necessarily predicted. We have come to realize that FLEA can be used to monitor for event-based conditions, whether or not they represent assumptions of expected system usage.*

The varying thicknesses of the arrows in the figure indicate our crude expectation of the relative frequences - users provide few declarations of event combinations to monitor for, the monitored system yields many instances of events, and the monitor generates moderately many notifications. This is not a requirement - for example, the user could define event combinations that never arise, and so never generate any notifications. The efficiency of the monitoring software is roughly in line with these expectations - when the user sends the monitor a new event combination, compilation is relatively slow, whereas when the monitor receives an event from the monitored system, handling it is relatively fast. There can be multiple sources of observed events, multiple users expressing event combinations of interest, and multiple destinations to which notifications are sent.

We can roughly divide flea into three parts. The main focus of this document is the most well developed part of flea, which is the language for defining interesting data in terms of the primitive data. The other parts deal with input to the monitor and output from the monitor, i.e., what the primitive events will look like, how they can be observed, what form notifications will take and how they should be delivered. In general, the monitor process has little control over the processes with which it is to communicate. Typically these will have to be started by some mechanism independent of the monitor, and flea will be used only to describe to the monitor this environment into which it must fit. The input and output parts of the language are really only collections of utilities that cover the cases we have so far encountered. End users can add to these collections of utilities, but only by writing new programs. We will describe some of the utilities that we have or plan to have.

The following constructs are part of FLEA. For details and a complete reference manual send e-mail to Cs3 or call (310) 337-3013:

- Data model
- Declaring events
- Declaring relations
- Logical formulae
- Temporal expressions
- Computational conveniences
- Transition events
- Additional arguments to event and relation declarations
- Input to the monitor
- Output from the monitor
- Appendix I - syntax
- Appendix II - predefined relations