Reform:
A Domain Specific Language

Dustin Graves

CSCI 342

April 26, 2007

Overview

- Interpreted language
- Monitors and manages data streams
  - Network, File, RS-232, etc
- Reformats and redirects data
- Contains keywords for data source and format definition, creation, and manipulation
- Operates within a distributed environment
  - Remote communication and manipulation
- Authenticates and controls remote access to resources

Interpreter

- Each active interpreter instance acts as a component of a larger distributed network of interpreters
  - An active interpreter instance is referred to as a Reformer
- Contains three key elements
  - Sources – A collection of data sources for receipt and transmission of data
  - Formats – A collection of data formats recognized by the Reformer
  - Channels – A collections of channels for remote Reformer manipulation
- Authenticates remote procedure calls
  - Remote manipulation can be restricted to limit access to data
  - Commands may only be executed by authorized Reformers

Key Elements

- Data Sources receive and transmit data through existing streams
  - Can be created, destroyed, and linked
    - Assigned IN, OUT, or INOUT type
  - Send and receive data to and from a stream
    - Procedures for converting, filtering, and processing data are applied after reading and before writing
  - Have caches with adjustable size
- Data Formats define mappings between records of different types
  - Can be predefined or defined at runtime
  - One or more Data Formats are registered with a data source
    - Specify how data source should decode and encode raw data
    - Characteristics of the raw data are used to determine the specific format type when decoding
  - Individual format types are mapped to other format types
    - Convert data for use with other streams
- Channels communicate with and manipulate remote Reformers
  - Can be opened and closed
  - Transmit language commands for the remote Reformer to process
- Authentication Modules control access from remote Reformers
  - Authentication is performed at the keyword, source, format, and channel levels
  - Custom Authentication modules can be supplied to authenticate requests through different methods
    - Interpreters provide credentials when establishing communication channels

Design

- Each key element is defined as an abstract base class, or interface, designed to be extended
  - The Data Source provides a basic interface for sending and receiving data
    - Extended to support communication with sockets, pipes, COM ports, and files
  - The Channel is a specialized Data Source for persistent two-way communication
    - Requires a Listener to accept connections from other Reformers
  - The Data Format provides a basic interface for mapping binary and ASCII data segments to a structured record
    - Contains a collection of fields with an ID, type, and value to represent data elements
    - Has a unique ID field to distinguish between different objects contained by an individual stream
    - Contains a collection of procedures for decoding raw data, mapping data to other formats, and filtering/processing data
      - Can employ conditional logic, looping, and simple functions for decoding and mapping
  - The Authentication Module provides an interface for testing remote access to Reformer attributes
    - Allows definition of custom authentication methods and layers
- Data Sources are created with a Factory
  - Each Data Source must provide a Creator capable of constructing a Data Source from a variable length list of parameters
  - Data Sources may be loaded dynamically at run time and registered with the Factory
- Monitors asynchronously receive data from Data Sources
  - Apply filters, modification procedures, and perform type conversion before transmitting to linked Data Sources
- The Reformer is a Singleton providing an interface to core language features to be accessed by the Parser
  - Performs synchronization of local and remote procedure calls

Implementation

- A C# Implementation of a basic interpreter capable of managing sources and channels
  - Consists of an Execution module, Parser module, and Core module
- The Execution Module receives language commands and dispatches them to the Parser
  - A delegate is provided to allow remote commands received by the Core to be dispatched to the Parser
- The Parser is generated with the ANTLR Parser Generator
  - Processes language commands and executes the corresponding Core procedures through the Reformer Singleton
- The Core contains the modules responsible for managing the state of the Reformer
  - Implements the Data Source, Format, and Channel interfaces
    - Provides concrete implementations of a File Data Source, Unicast Socket Data Source, and TCP Channel
  - Implements a Generic Factory interface and provides a concrete implementation of a Data Source Factory
  - Implements a Channel Listener for receiving connections from remote Reformers
  - Implements Source and Channel Monitors for receiving, processing, and sending data
  - Implements the Authenticator interface and provides a concrete implementation of a Read Only Authenticator
  - Implements a Reformer to manage core resources at run time
    - Sources are managed with the ‘create’ and ‘destroy’ keywords
    - The ‘->’ is used to link two sources together
    - Listeners are managed with the ‘listen’ and ‘deafen’ keywords
    - Channels are managed with the ‘open’ and ‘close’ keywords
    - Channels, Sources, and Formats are listed with the ‘channels’, ‘sources’, and ‘formats’ keywords
    - Individual Channels, Sources, and Formats are accessed with the ‘channel[]’, ‘source[]’, and ‘format[]’ operators
    - The print keyword will print basic type values and descriptions for channels, sources, and formats.