Developing Signal-Pattern-Recognition Programs

Software system aids development of application programs that analyze signals.

Lyndon B. Johnson Space Center, Houston, Texas

Pattern Interpretation and Recognition Application Toolkit Environment (PIRATE) is a block-oriented software system that aids the development of application programs that analyze signals in real time in order to recognize signal patterns that are indicative of conditions or events of interest. PIRATE was originally intended for use in writing application programs to recognize patterns in space-shuttle telemetry signals received at Johnson Space Center's Mission Control Center: application programs were sought to (1) monitor electric currents on shuttle ac power busses to recognize activations of specific power-consuming devices, (2) monitor various pressures and infer the states of affected systems by applying a Kalman filter to the pressure signals, (3) determine fuel-leak rates from sensor data, (4) detect faults in gyroscopes through analysis of system measurements in the frequency domain, and (5) determine drift rates in inertial measurement units by regressing measurements against time. PIRATE can also be used to develop signal-pattern-recognition software for different purposes — for example, to monitor and control manufacturing processes.

PIRATE was preceded by a custom stripchart-analysis program that took a long time to develop and offered little opportunity for reuse. Also available prior to the development of PIRATE were commercial block-oriented development software systems that were useful for prototyping but exhibited significant limitations: for example, they could not be used to produce real-time application programs, could not be used to develop software compatible with the hardware and the other software of the Mission Control Center, could not be used to develop application programs that could function in the face of the communication difficulties (especially, intermittency and errors) inherent in monitoring remote equipment, and could not provide pattern-recognition capabilities. PIRATE overcomes these deficiencies to a large extent, and goes beyond that by including a C-language interface that provides unprecedented flexibility.

PIRATE includes the following components:

- *The PIRATE data-flow language.* An application program is specified by use of the PIRATE data-flow language. An application-program specification defines which data processing modules will be used in the program and establishes the data flowing among the modules. Similarly, for building a module, one specifies the flow of data into and out of a module by use of the PIRATE language.
- *The PIRATE predefined modules.* PIRATE contains several predefined modules, including ones for data communication, signal processing, and data filtering. Among these are software tools to filter out the highly non-Gaussian errors that are typical of the communication process while leaving the nonerroneous data intact. (Most other signal-processing software filters that can remove non-Gaussian errors also undesirably modify the underlying signals.) Also among the predefined modules are a Bayesian classifier and other software tools for interpreting the contents of signals.
- *The PIRATE code generator.* The PIRATE code generator translates an application-program-specification file and the associated module-configuration files into a standard C-language file. This file contains the main routine for the application program. A C compiler can then compile and link this file to produce an efficient real-time pattern-recognition application program.
- *The PIRATE object library.* The generated code makes calls to several PIRATE infrastructure routines. The PIRATE object library contains the object code for these infrastructure routines and for the predefined module routines.
- *The PIRATE "imake" facility.* The imake is a programming software tool that was developed to address issues of portability pertaining to the X Window System and to provide a high-level view of the software-building process. However, the standard imake suite explicitly targets the construction of X Window System software. The PIRATE imake facility takes advantage of the standard imake suite where practical, but targets the construction of PIRATE and PIRATE application programs rather than X Window System software.
- *An architecture for the development of modules by the user.* PIRATE is intentionally an open-ended software tool. While simple application programs can be constructed by use of the predefined modules, it is likely that a useful pattern-recognition application programs could not. Instead, domain-specific logic can be expected to be necessary. PIRATE enables the implementation of domain-specific knowledge in the widely used C programming language. The architecture for user-developed modules specifies how such domain knowledge can be used in a PIRATE application program.

PIRATE is used both in building a pattern-recognition application program and in the real-time execution of that program. To build an application program, one constructs an application-specification file, application-specific modules, and application imake file. The imake file identifies the components that form the executable application program and directs construction of these components from the source files developed by the user.

During execution of the pattern-recognition application program, the source module of the program acquires the incoming data and uses the PIRATE infrastructure to feed the data to downstream modules. The infrastructure sends the appropriate data to the appropriate modules, which operate on the data. Each module uses the PIRATE infrastructure to send its output to modules downstream of it.

In PIRATE, transmission of data between data-processing modules is always performed by calls to C functions that are parts of an executable module. In other software systems, data are often transmitted between modules via operating systems; the computational overhead of doing so is often several orders of magnitude greater than that of PIRATE. In PIRATE, the processing of data within a module is performed by compiled functions. In other systems, the processing of data may be performed by interpreters, which, again entail computational overhead much greater than that of PIRATE.

This program was written by Robert O. Shelton of **Johnson Space Center** *and David Hammen of LinCom. For further information, contact the Johnson Technology Transfer Office at (281) 483-3809.*
MSC-22944