Device Oriented Class Library

Ralph Lange
BESSY II
EPICS Collaboration Meeting @ SLAC May 1999

Sequential procedures are an integral part of the regular operation of an accelerator: reading out and restoring snapshots, cycling magnets, ramping or degaussing parts of the installation are repetitive operation procedures.

Concept

Some complex devices (e.g. RF transmitters) are highly complex and most difficult to handle. Errors in the operation procedures may lead to time-consuming delays or even to equipment damage. Quite often there are lists or sets of devices that have to be processed in a certain order or other devices that have to be waited for before continuing with the next process task.

So the major goals for a generic sequential procedure tool are:

1. Hide the device complexity from the procedure designer, i.e. allow designing procedures without presuming intimate device knowledge
2. Make designing procedures easy and fast with short turn-around times.

The BESSY I Solution

Within the old BESSY I control system, all procedures were hidden in fixed application programs (written in Fortran). Even minor changes needed recompilation of the complete application. Procedure structures and device specific code were mixed in the source files, which made the code hard to change and maintain.

When the new BESSY I control system was introduced, a new C++ application was implemented that took care of all the procedural tasks:

A simple “home-made” interpreter reads macro files that are pre-processed by CPP to allow `#define` and `#include` directi ves. Commands in these macro files instantiate objects from a device class library and lists of these devices. Once instantiated, the devices’ methods (including read-from-file, write-to-file, read-from-hardware, write-

to-hardware, drive-to-setpoint etc.) are accessible from within the macro files. The data file manager organizes writing and reading device data in a proprietary ASCII format. An error log handler writes error messages to an attached terminal and/or file.

This application is still extensively used at BESSY I for saving and restoring snapshots, cycling and ramping the ring, injection control and switching between the operational states of the machine (injection, up, standby, standby-over-weekend, shutdown, off).

The EPICS Approach

To create a similar application to be used at BESSY II operation, the following changes will have to be made:

- The class library will be adapted to the BESSY II devices, i.e. it should be stripped down to the abstract base classes and rebuilt for the new device classes
- The device inteface will be changed to use CA or CDEV (or both)
- The data file interface will be changed to use an EPICS compatible format (i.e. BURT or SDDS file format)
- The logging interface will be changed to use the CMLOG logging facility.
- The old buggy interpreter will be replaced by tcl/tk and perl bindings

SWIG – Interface Generator

SWIG is a public domain tool that takes a C, C++ or Objective C library (plus header files) and automatically creates bindings and interfaces for a couple of well-known scripting languages, including perl, tcl and python. See: `http://www.swig.org`

This tool will be used to create the bindings for the device class library to make it accessible from background scripts (perl) as well as from small OPI applications (tcl/tk based).

Device Oriented Class Library

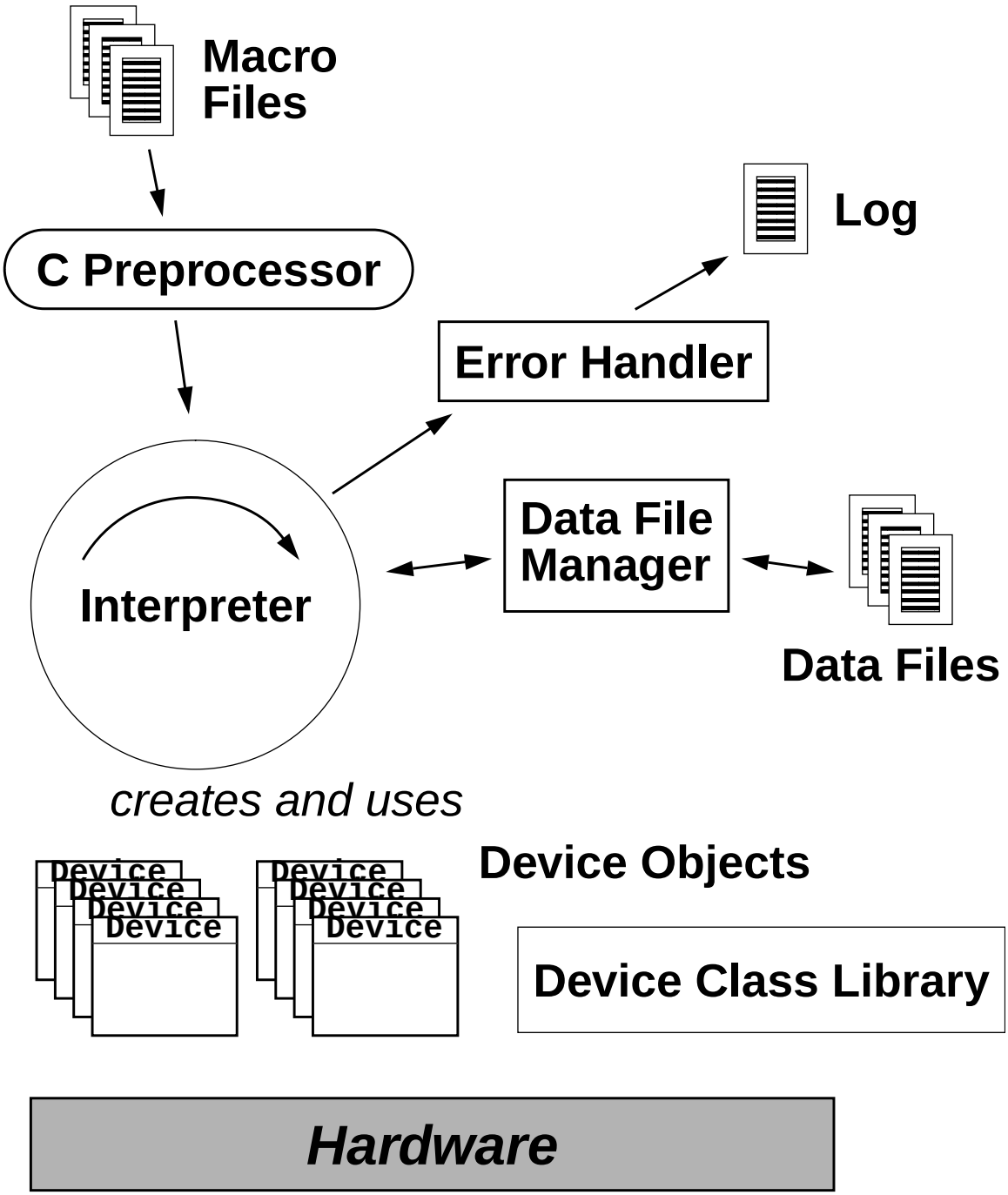
- Concept
- Historical Background (BESSY I Solution)
- New Design (EPICS Approach)
- SWIG – Interface Generator Tool

Concept

- Sequential Procedures are an Integral Part of Regular Operation
- Complex Devices and Lists or Sets thereof are Difficult to Handle
- Goals:

  1. Hide Device Complexity from the Procedure Designer (Physicist)
  2. Make Designing Procedures Easy and Fast (Use Interpreter)

The BESSY I Solution



The EPICS Approach

- Strip Down and Clean Up Class Library
- New Device Interface Library
  => to use CA or CDEV
- New Data File Interface
  => to use BURT or SDDS File Format
- New Log Interface
  => to use CMLOG
- Get Rid of the Old Interpreter
  => to be used Directly from Perl and Tcl/Tk

SWIG – Interface Generator

- PD Code Development Tool (University of Utah)
- Takes C, C++ or Objective-C Code (Library plus Header File is Sufficient)
- Automatically Generates Bindings and Interfaces for a Selection of Well-Known Scripting Languages (like Perl, Tcl and Python) and Produces Generic Interface Documentation (LaTeX, HTML, ASCII)