Java/C++ eExceptions and Message Logging

Language-independent exception definition/handling and general message logging.

(Adventures in Middleware)

Ron MacKenzie, Greg White

Why is this important?

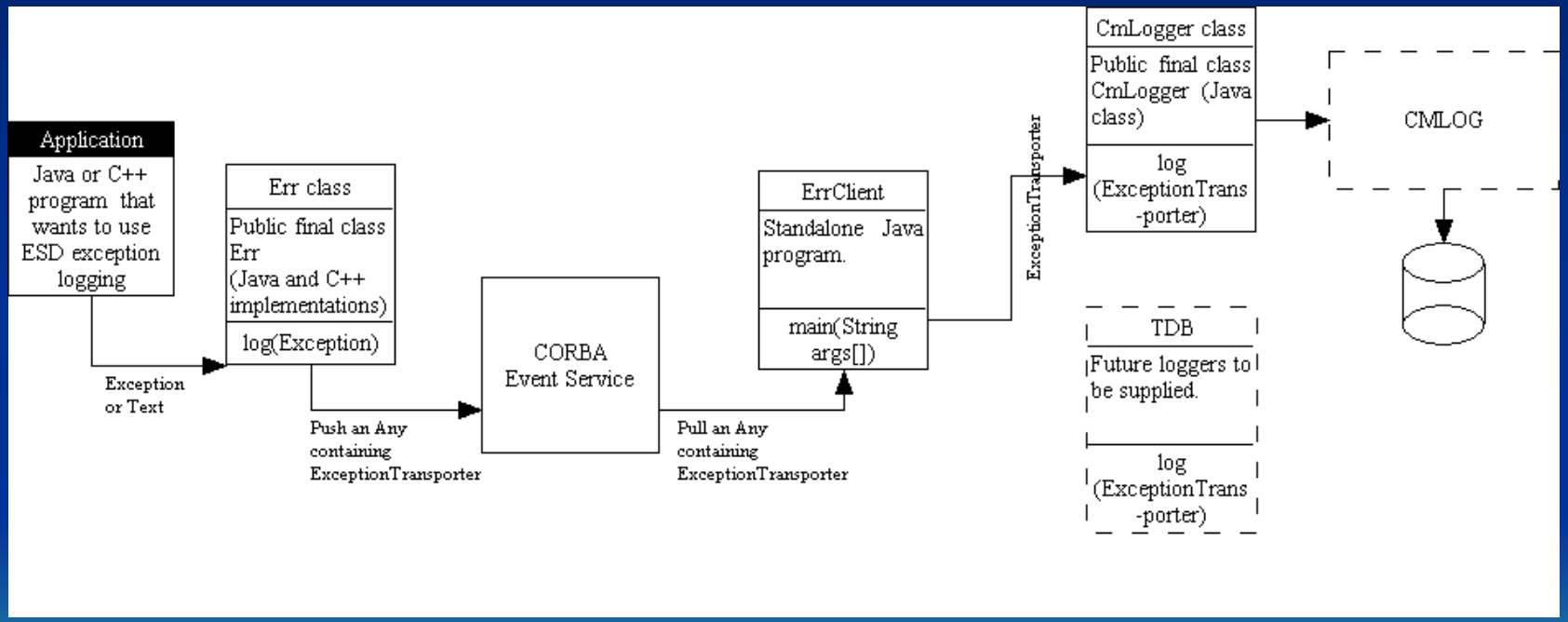
- Highly distributed control system with thousands of instances of programs.
- Coherent and Reliable message logging is essential for keeping the system running.
- It is necessary to convey the context in which an exception was thrown in distributed multitiered environment.
  - Call Stack often lost (by serialization)
  - Layered application context needed.
  - Solution: Exception Chaining provides context.

Distributed Network

Features

- Language Independence (Java and C++)
- Define the exceptions that C++ and Java apps throw all in one place.
  - ✓ Standard High level set of Exceptions (like DataNotFound).
  - ✓ Corba IDL is used for exception definition
- Encapsulate details when exceptions occur.
- More details can be successively added while logging.
- Informational and Slow Speed Data logging are other uses in addition to exception logging.
- Corba Event Service is used for transport.
- IONA Orbacus is the ORB (distributed Objects++).

Data Flow


Application
Java or C++ program that wants to use ESD exception logging
Exception or Text
Err class
Public final class Err (Java and C++ implementations)
log(Exception)
Push an Any containing ExceptionTransporter
CORBA Event Service
Pull an Any containing ExceptionTransporter
ErrClient
Standalone Java program.
main(String args[])
ExceptionTransporter
CmLogger class
Public final class CmLogger (Java class)
log (ExceptionTrans -porter)
TDB
Future loggers to be supplied.
log (ExceptionTrans -porter)
CMLOG

Features (cont)

- Message logging will never block the application.
- Err.log is resilient.
  - If Err.log can’t connect to Event Service, message will simply be logged to stderr.
  - If Event Service is bounced, Err.log will reconnect on next log call.

ErrClient and Loggers

- ErrClient is a Corba Pull Consumer
- ErrClient and CmLogger are pure java.
- It is easy to plug in other logger classes.

ErrClient and Logger auto-reconnect.

- If the event service is down, ErrClient process reconnects automatically when the event service is brought back up.
- When the message destination (e.g. CMLOG) is down, the CmLogger process reconnects automatically when it is brought back up.

IDL Defined Structure

- Exception Transporter is a simple structure.
- Send it across the Event Service.

```
#ifndef EXTRANS_IDL
#define EXTRANS_IDL
/**
 * IDL for exTrans.  This is the Error Transporter object (structure).
 *
 * @version 10-May-2003, Ronm.
 */

#pragma prefix "slac.stanford.edu"

module err {

  struct ExceptionTransporter {
    string destination;        // Which logger this msg is going to.
    string name;
    string suppl;
    long time;                 // IDL long is java int.
    string facility;
    string host;
  };
};
#endif
```

Part II

Application Programmer's Perspective

- Err and Except Classes

Except Class (definitions)

- All exceptions defined in one file
  - For example: except.idl
  - To add new exceptions
    - Cvs checkout, add the exception, cvs commit
    - Gmake
    - Presto! Java and C++ include files created.
  - IDL Examples
    - exception UnableToGetDataException {};
    - exception NameLookupException {};
    - exception MonitorStartupException {};

Using the Err class

- Include/Import
  - JAVA
  - import {java incl path}.except.*; // exceptions
  - import {java incl path}.err.*; // Err logger

  - C++
  - #include "except.h"
  - #include "err.h"

Using the Err class (cont)

- Instantiate
  - JAVA
  - Err err("AIDA Magnet Server");
    - Instantiate with facility name
  - C++
  - Err err("AIDA Magnet Server");
- *Err is singleton design pattern

The Err class and exception chaining

- A logged exception consists of **three strings** which are concatenated to form the logged message
  - EXCEPTION_NAME + REASON STRING + SUPPLEMENTAL STRING
- new NoDataSourceException("for PB60:LUMVAL");
- err.log(ex, "while acquiring the Luminosity Value");

- Logged Message:
  - "NoDataSourceException for PB60:LUMVAL while acquiring the Luminosity Value“
- A form of Exception Chaining!

Using the Err class (cont)

- These exceptions are sub-classed from the Java Throwable class, so they are thrown and caught like other Java exceptions.

```
try {  ...
    throw new      NoDataSourceException("for PB60:LUMVAL");
}
catch (Exception ex) {
   err.log(ex, "when attempting to acquire the Luminosity Value" );
}
```

Logged Message:

"NoDataSourceException for PB60:LUMVAL when attempting to acquire the Luminosity Value"

Using the Err class (cont)

- Err.log returns an exception, so you can log and throw an exception in one statement like this:

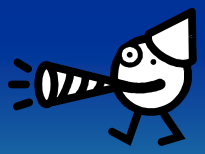
```
throw err.log(new NoDataSourceException("for
    PB60:LUMVAL"), " when starting ca service");
```

Deliverables

- Requirements/Design document
- Programmer's guide
- System Administrator's guide
- Javadoc
- Test and Example programs
- All tied together on web page
- http://www.slac.stanford.edu/grp/cd/soft/err/

Project Status

- Delivered and in production!
- AIDA Application (PEPII) currently using Err.
- Planning for LCLS applications to use Err.
- Email me to obtain the source (ronm@slac)



An outline the use of “Any”

- Exception Transporter struct defined in IDL
  - ExceptionTransporter et =
    new ExceptionTransporter (*fill fields here*);
- Push the Corba Any
  - Any any = orb.create_any();
  - ExceptionTransporterHelper.insert(any, et);
  - pushMessage(any);
- Pull the Corba Any
  - ExceptionTransporter et = new ExceptionTransporter();
  - Any any = null;
  - any = supplier.pull();
  - et = ExceptionTransporterHelper.extract(any);

Development Tools

- Formal internal review with action items tracked.
- Emacs,
- IDL
- Gmake (GNU) / Unix Dev Environment(UDE)
- Gdb (C++ debug)
- CVS, CVS-WEB
- UDE Process management (shell scripts / ssh).