MONITORING APPLICATIONS ONCE THEY ARE RELEASED INTO THE USER COMMUNITY

Sev Binello

BROOKHAVEN NATIONAL LABORATORY

2009 ICALEPCS

Imagine…

- You are using an application and it…
  - crashes
  - has bugs
  - does not do what you need it to do

And on the other hand…

- You developed the application and need to learn…
  - that it crashed
  - that is has bugs
  - that it does not work as the user needs it to work

Wouldn’t It Be Nice If…

- Users could instantly contact developers right when they were using the application to:
  - report a bug?
  - request a modification?

- Developers were:
  - notified right away when an application crashed?
  - given information that enabled them to debug the application?
  - able to determine where other faulty instances were running?

Our Approach

- Three systems were developed at RHIC to address these issues:
  - Crash Utility
    - gathers crash information, stores core file and notifies developers
  - Send FeedBack
    - gathers information from users, stores it in our "Action Please" trouble-tracking system and notifies developers
  - Application History
    - gathers information about application usage and reliability and stores it in a database

All 3 Systems Designed With The Following In Mind …

- Affect the application as little as possible
  - most of the work done in a separate process
  - avoid complex communication mechanisms

- Do setup work up front, before the application does what it is designed to do

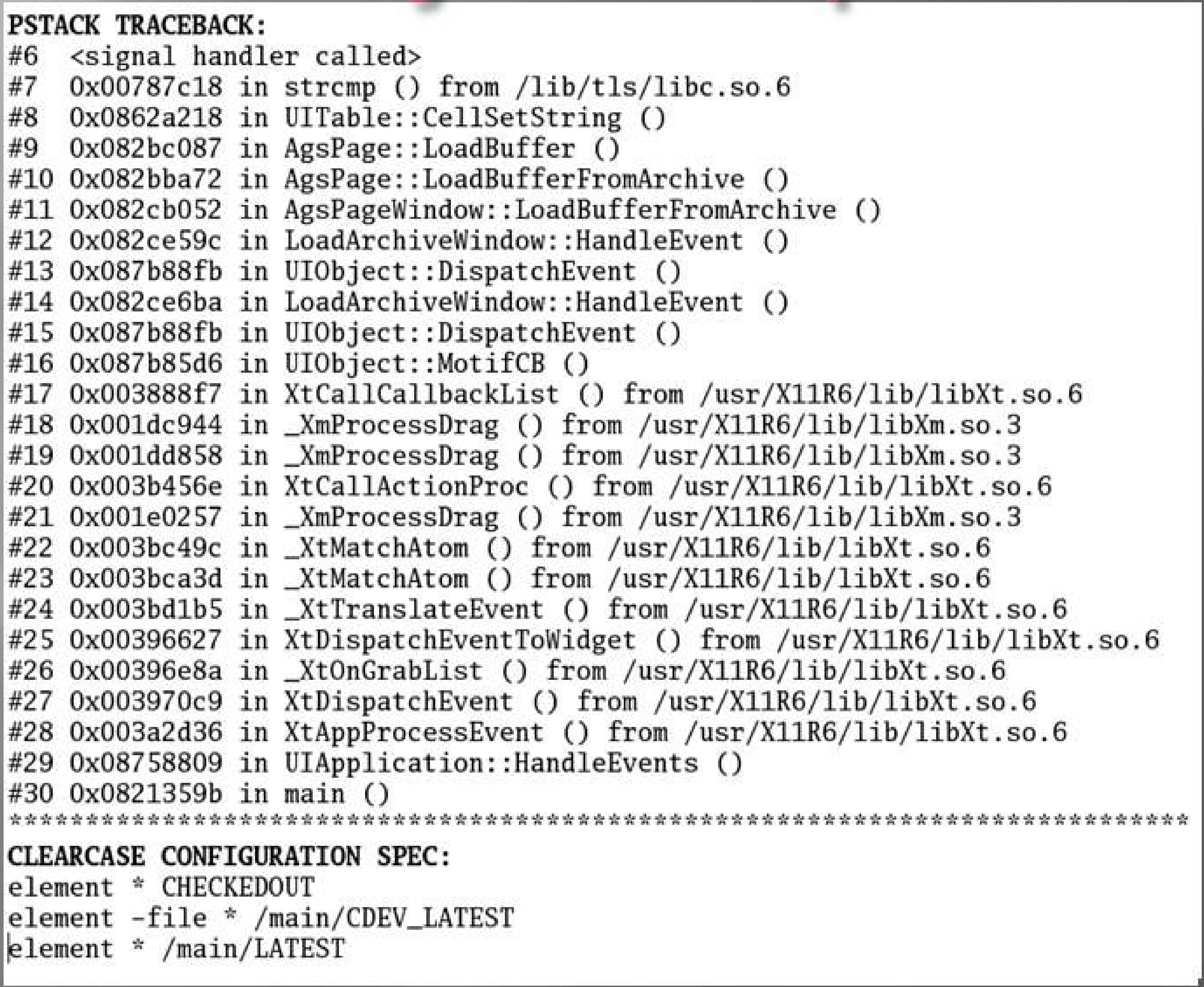
- Make it easy for developers to include in their applications

Crash Utility System
What Do We Get From It ?

- Immediate notification that an application has crashed
- Information about the process
- User comments
- Stack trace and core file
- Source version information

Crash Report – Example email

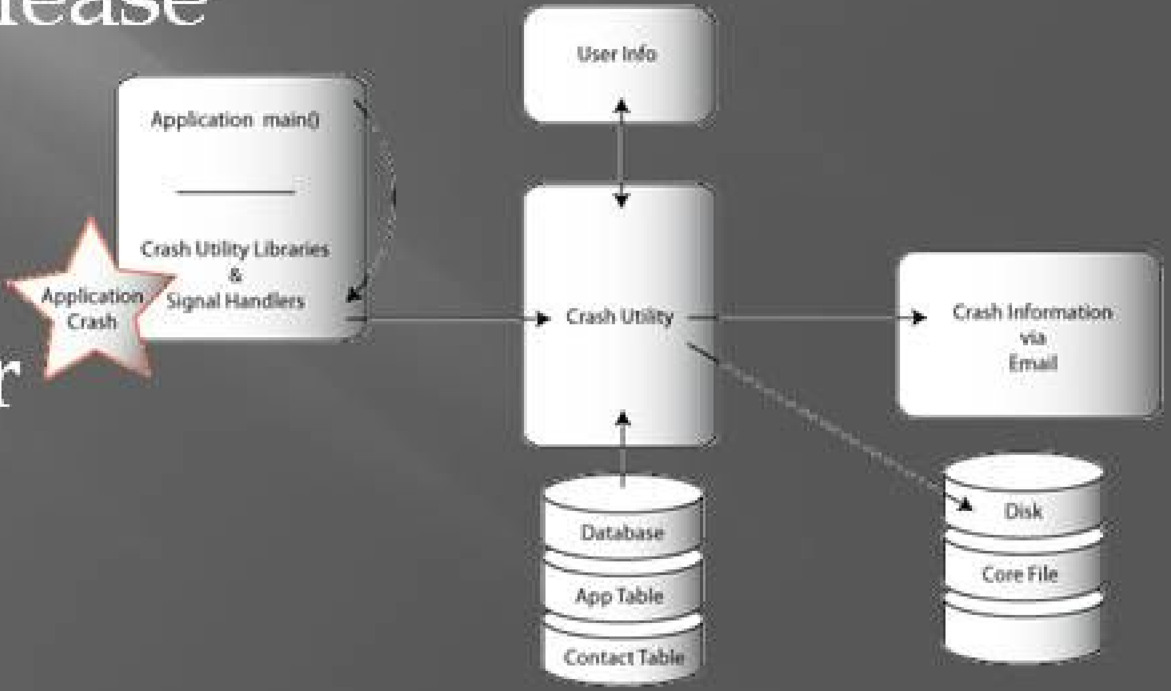
Crash Utility – Example email



```
PSTACK TRACEBACK:
#6  <signal handler called>
#7  0x00787c18 in strcmp () from /lib/tls/libc.so.6
#8  0x0862a218 in UITable::CellSetString ()
#9  0x082bc087 in AgsPage::LoadBuffer ()
#10 0x082bba72 in AgsPage::LoadBufferFromArchive ()
#11 0x082cb052 in AgsPageWindow::LoadBufferFromArchive ()
#12 0x082ce59c in LoadArchiveWindow::HandleEvent ()
#13 0x087b88fb in UIObject::DispatchEvent ()
#14 0x082ce6ba in LoadArchiveWindow::HandleEvent ()
#15 0x087b88fb in UIObject::DispatchEvent ()
#16 0x087b85d6 in UIObject::MotifCB ()
#17 0x003888f7 in XtCallCallbackList () from /usr/X11R6/lib/libXt.so.6
#18 0x001dc944 in _XmProcessDrag () from /usr/X11R6/lib/libXm.so.3
#19 0x001dd858 in _XmProcessDrag () from /usr/X11R6/lib/libXm.so.3
#20 0x003b456e in XtCallActionProc () from /usr/X11R6/lib/libXt.so.6
#21 0x001e0257 in _XmProcessDrag () from /usr/X11R6/lib/libXm.so.3
#22 0x003bc49c in _XtMatchAtom () from /usr/X11R6/lib/libXt.so.6
#23 0x003bca3d in _XtMatchAtom () from /usr/X11R6/lib/libXt.so.6
#24 0x003bd1b5 in _XtTranslateEvent () from /usr/X11R6/lib/libXt.so.6
#25 0x00396627 in XtDispatchEventToWidget () from /usr/X11R6/lib/libXt.so.6
#26 0x00396e8a in _XtOnGrabList () from /usr/X11R6/lib/libXt.so.6
#27 0x003970c9 in XtDispatchEvent () from /usr/X11R6/lib/libXt.so.6
#28 0x003a2d36 in XtAppProcessEvent () from /usr/X11R6/lib/libXt.so.6
#29 0x08758809 in UIApplication::HandleEvents ()
#30 0x0821359b in main ()
*****************************************************************************
CLEARCASE CONFIGURATION SPEC:
element * CHECKEDOUT
element -file * /main/CDEV_LATEST
element * /main/LATEST
```

Crash Utility Components

- C library linked into C, C++ applications (not JAVA)
- Predefined development environment
- Web-based application release procedure
- Database containing application and developer information
- Crash Utility process



Crash Utility Library

- Initialization code
  - sets up "Signal Handlers"
  - prepares static information and commands (e.g. time, machine, pid)
- Signal Handlers
  - catch the signal and instantiate the Crash Utility process

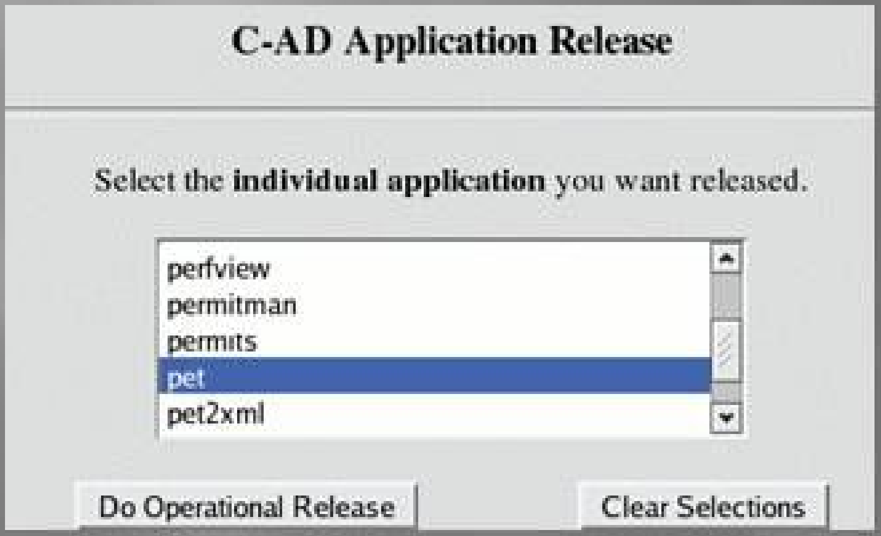
Development Environment

- Facilitates debugging
  - information embedded in executable
    (e.g. compiler, OS and application source versions)
  - applications built with debugger option turned on
    (i.e. g++ -g)
  - symbolic information left in executable

Advantage: Debugging core files now same as debugging a running application

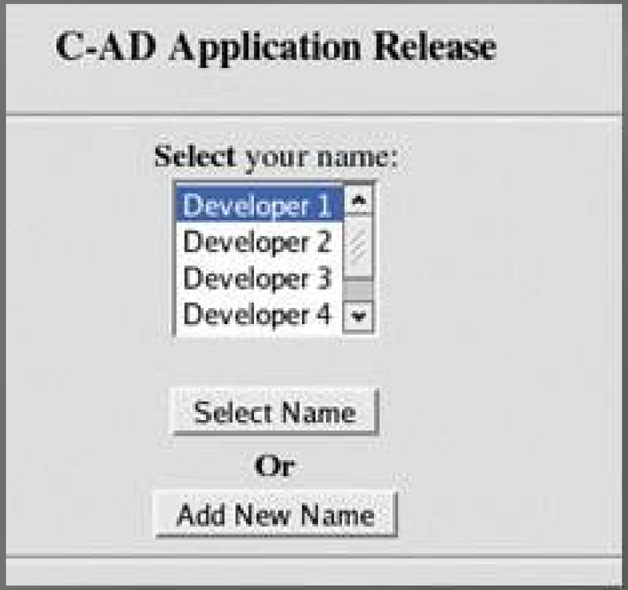
Web-Based Release Procedure

- Records *who, why,* and *when* an application is released
- Maintains information about applications and developers in a database



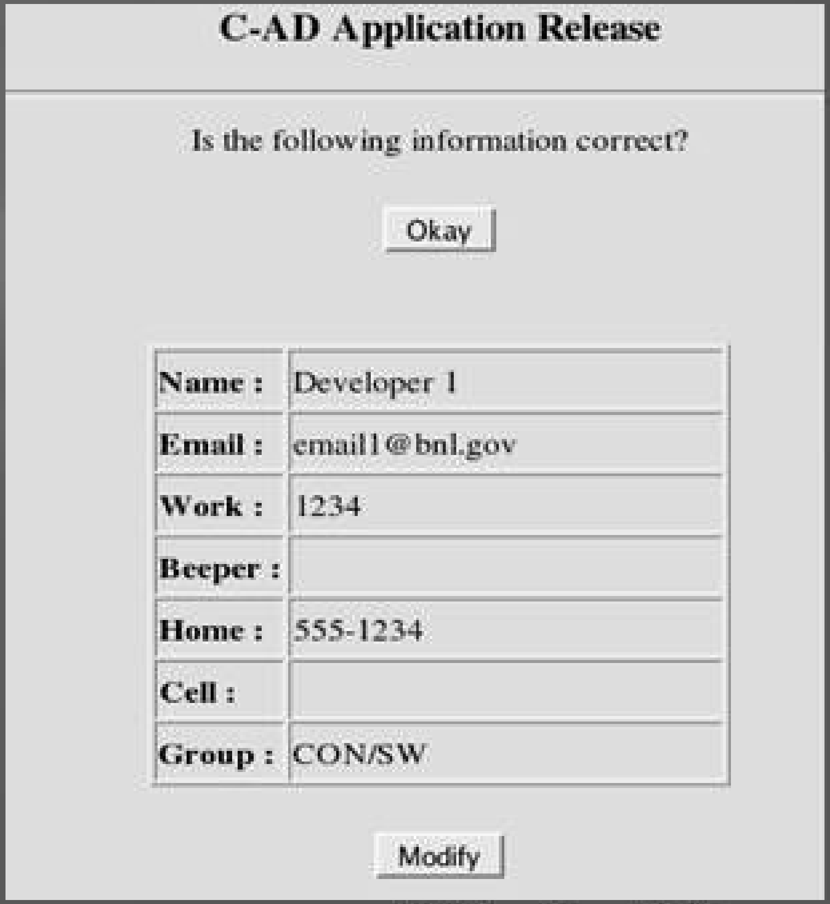
C-AD Application Release

Select the **individual application** you want released.



C-AD Application Release

Select your name:



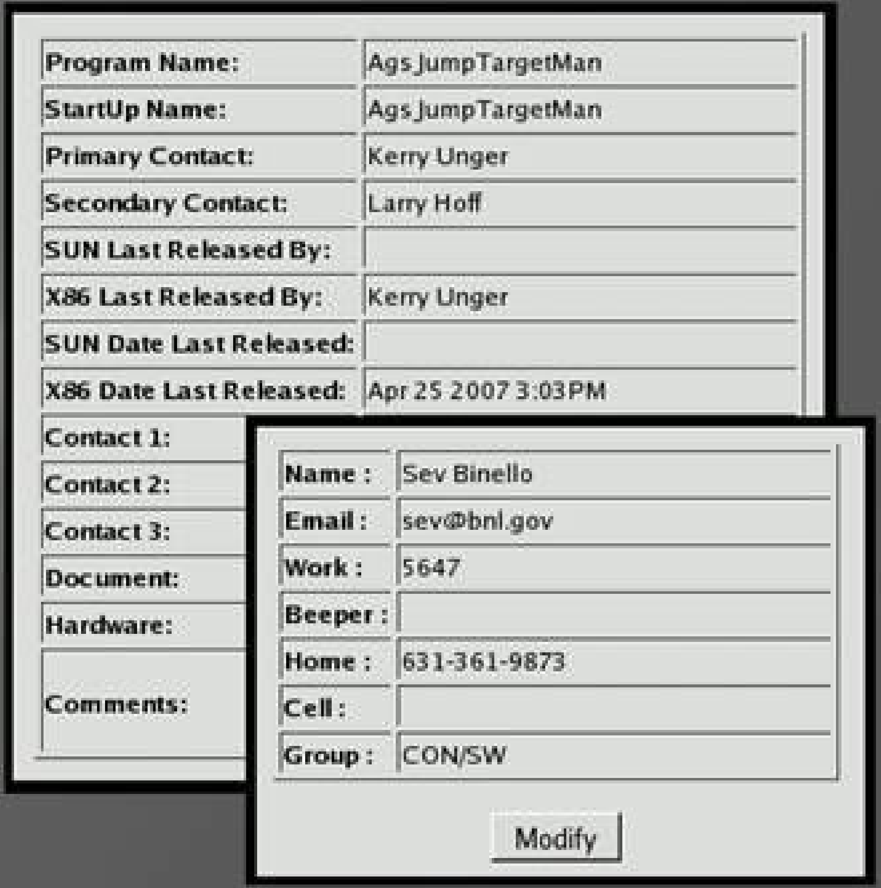
C-AD Application Release

Is the following information correct?

Name :	Developer 1
Email :	email1@bnl.gov
Work :	1234
Beeper :	
Home :	555-1234
Cell :	
Group :	CON/SW

Database

- Contains information linking applications to developers
  - application name
  - contacts
  - developer that last released the application
  - location of documentation
  - brief description
  - relevant hardware



Program Name:	Ags JumpTargetMan
StartUp Name:	Ags JumpTargetMan
Primary Contact:	Kerry Unger
Secondary Contact:	Larry Hoff
SUN Last Released By:	
X86 Last Released By:	Kerry Unger
SUN Date Last Released:	
X86 Date Last Released:	Apr 25 2007 3:03PM
Contact 1:	
Contact 2:	
Contact 3:	
Document:	
Hardware:	
Comments:	

Name :	Sev Binello
Email :	sev@bnl.gov
Work :	5647
Beeper :	
Home :	631-361-9873
Cell :	
Group :	CON/SW

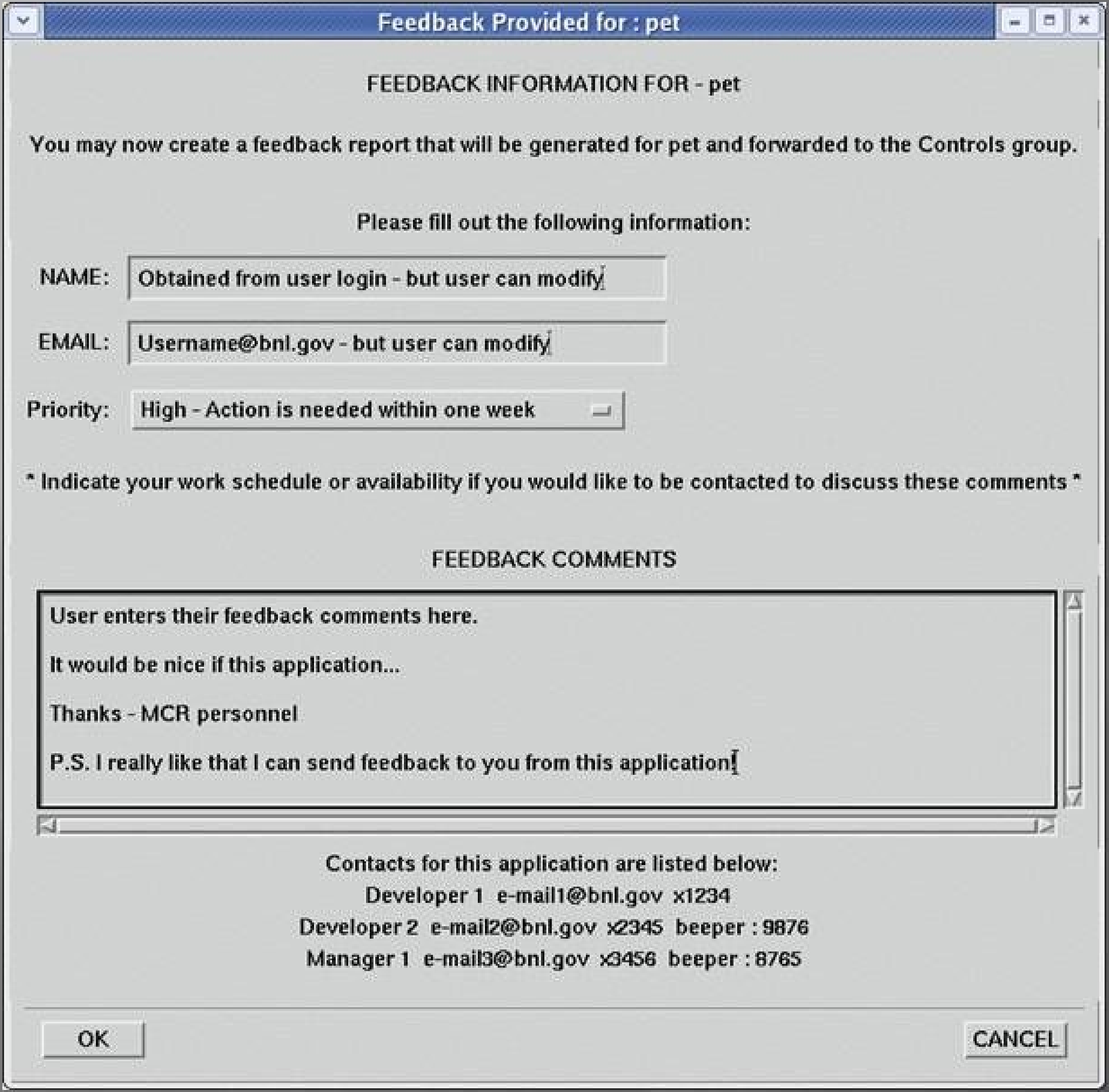
Crash Utility Process

- Does most of the processing
  - compresses and stores the core file
  - prompts the user for additional information
  - determines the developer(s) to notify
  - packages the crash information and sends email

Send Feedback System
What Do We Get From It?

- Instant user feedback
  - what bothers and/or frustrates a user as the application is being used
    - bugs
    - deficiencies
  - what new functionality a user may desire
- Permanent record instantly added to our "Action Please" trouble-tracking system

Send Feedback



Feedback Provided for : pet

FEEDBACK INFORMATION FOR - pet

You may now create a feedback report that will be generated for pet and forwarded to the Controls group.

Please fill out the following information:

NAME:	Obtained from user login - but user can modify
EMAIL:	Username@bnl.gov - but user can modify
Priority:	High - Action is needed within one week

* Indicate your work schedule or availability if you would like to be contacted to discuss these comments *

FEEDBACK COMMENTS

User enters their feedback comments here.

It would be nice if this application...

Thanks - MCR personnel

P.S. I really like that I can send feedback to you from this application!

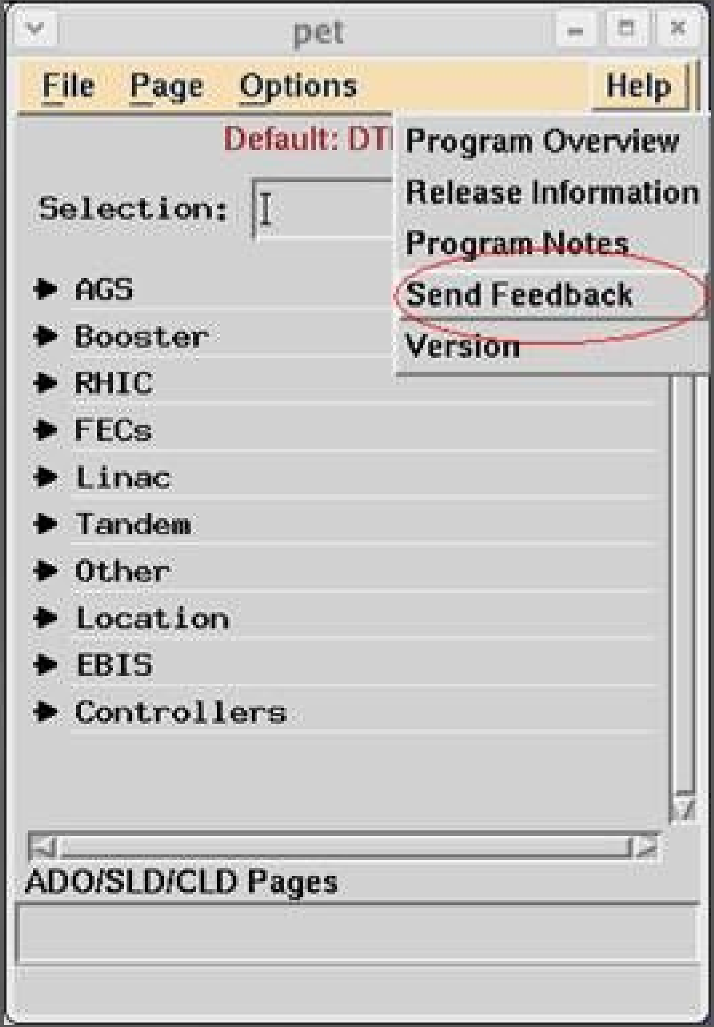
Contacts for this application are listed below:

Developer 1 e-mail1@bnl.gov x1234

Developer 2 e-mail2@bnl.gov x2345 beeper : 9876

Manager 1 e-mail3@bnl.gov x3456 beeper : 8765

OK CANCEL



Application History System What Do We Get From It?

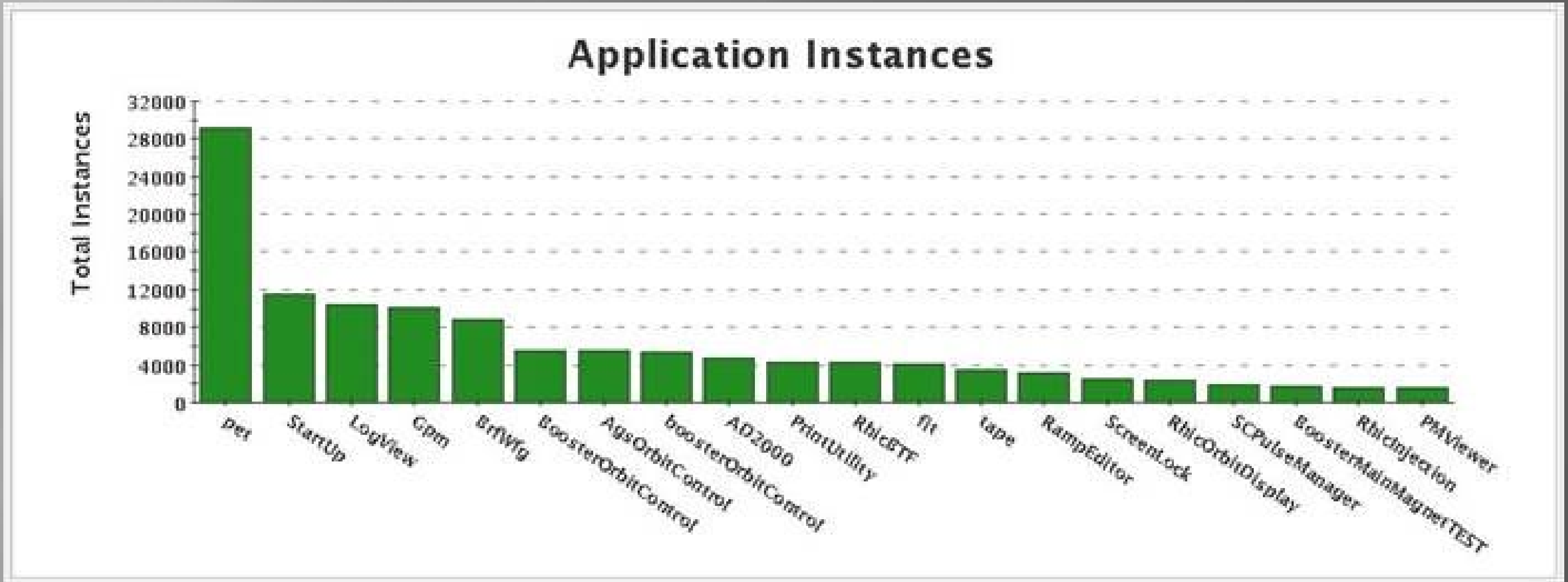
- Information about where an application is running and who is running it
- Discover usage patterns
  - how often is an application used?
  - who are the power users?
- Reliability statistics

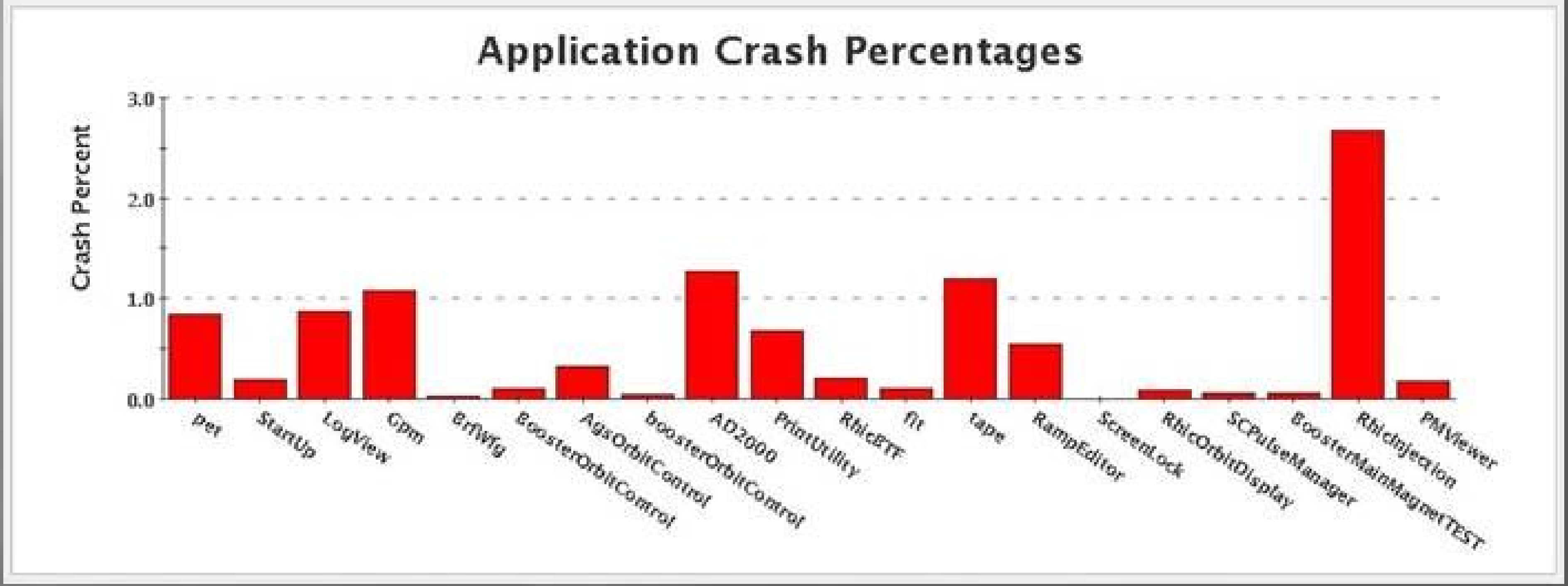
Application History Components

- Application library
  - writes information to an NFS mounted directory (e.g. start/stop times, machine, user name, exit status)
- Application History Server
  - polls, looking for messages
  - reads messages
  - stores information in database
  - deletes message

Advantage: Application is independent of server

Application History





Benefits We Have Seen

- Crash Utility System
  - Developers have found that
    - saving core files with debugger information included is extremely useful
    - real-time delivery of crash information very helpful
- Send Feedback
  - Used by operators to report problems and make requests
- Application History
  - Mostly used to locate applications
  - Work ongoing to further mine available information

QUESTIONS ?