SAND2008-0883P

Nightly Software Testing in Acro with EXACT and FAST

William E Hart
Stef Chakerian
Lee Ann Riesen
Jon Berry
Bob Heaphy
Cindy Phillips

Discrete Math and Complex Systems Dept.
Sandia National Laboratories
wehart@sandia.gov
www.cs.sandia.gov/~wehart

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

- Overview of Acro
- Managing software quality within Acro
- Testing with EXACT
- Distributed testing with FAST
- Ongoing/Future activities

Overview of Acro

The Acro Software Repository

Goal: Integrate management of Sandia optimization libraries

- Consolidation of configuration management
- Easier build management
- Consolidated testing
- Management of third-party solvers

acro - From Greek akros, at the point, end, or top.

acro - A Common Repository for Optimizers

acro - A generic term for warblers of the genus Acrocephalus, usually referring to the sedge and/or reed warblers. It spends summers in the UK and winters in Africa, south of the Sahara Desert.

Why Acro? Some history…

- Bill H was supporting configuration systems for a variety of loosely coupled optimization libraries (Coliny, PICO, UTILIB)
  - Required multiple checkouts
  - Required explicit installation of third-party software
  - Testing was problematic (and thus not done much)

- Why not integrate libraries into DAKOTA?
  - Developers focused on libraries
  - Developers only wanted to checkout the libraries they needed
    - Coliny vs PICO
  - Developers not as worried about portability (esp. in initial phases of development)

Acro Organization

Packages – distinct software libraries

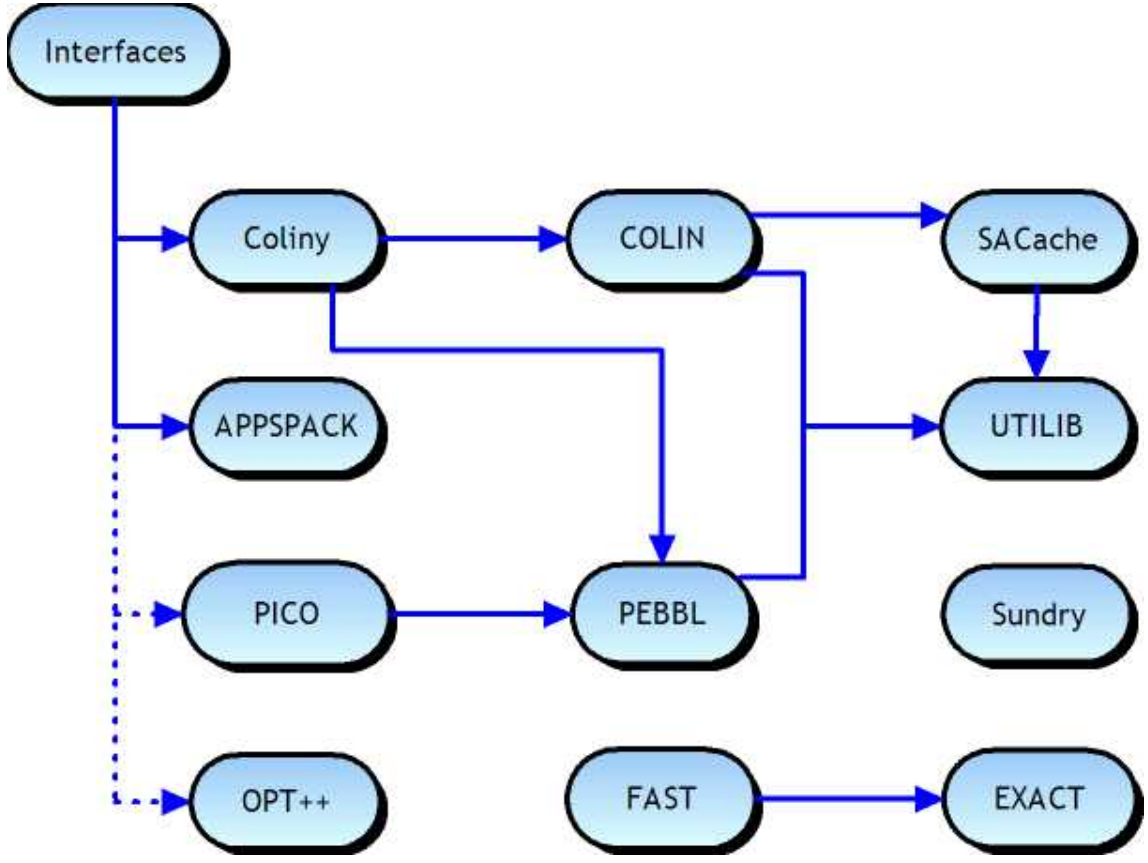
- SNL optimization libraries whose development is managed by the Acro subversion repository
- Snapshots of third-party libraries that integrated into the Acro subversion repository

Projects – ensembles of packages

- Acro manages package integration
- Projects can be checked out separately from the subversion repository
- Includes necessary third-party software
  - E.g. A user can checkout, build and run acro-pico without installing other third-party software

Acro Packages

Note: additional packages are under limited release, which is not stable enough for distribution right now



Overview of Acro Packages

APPSPACK
- Parallel pattern search
- Tammy Kolda

COLIN
- Optimization interface layer that supports optimization hybridization
- Bill Hart, John Siirola

Coliny
- Global optimization heuristics, derivative-free local search
- Bill Hart

EXACT
- Framework for software testing and experimental analysis
- Jonathan Berry, Bill Hart

FAST
- A framework for distributed test management
- Bill Hart (and Stef Chakerian)

Interfaces
- The ‘acro’ command-line interface, and wrappers for third-party solvers
- Bill Hart

Overview of Acro Projects (cont'd)

OPT++

- Nonlinear programming methods
- Patty Hough

PEBBL

- Parallel branch and bound library (scalable to 10,000s processors)
- Jonathan Eckstein

PICO

- PEBBL-based MILP solver
- Cindy Phillips

SACache

- Annotated caching tool
- Jean-Paul Watson, Bill Hart

Sundry

- Miscellaneous solvers
- Jon Berry

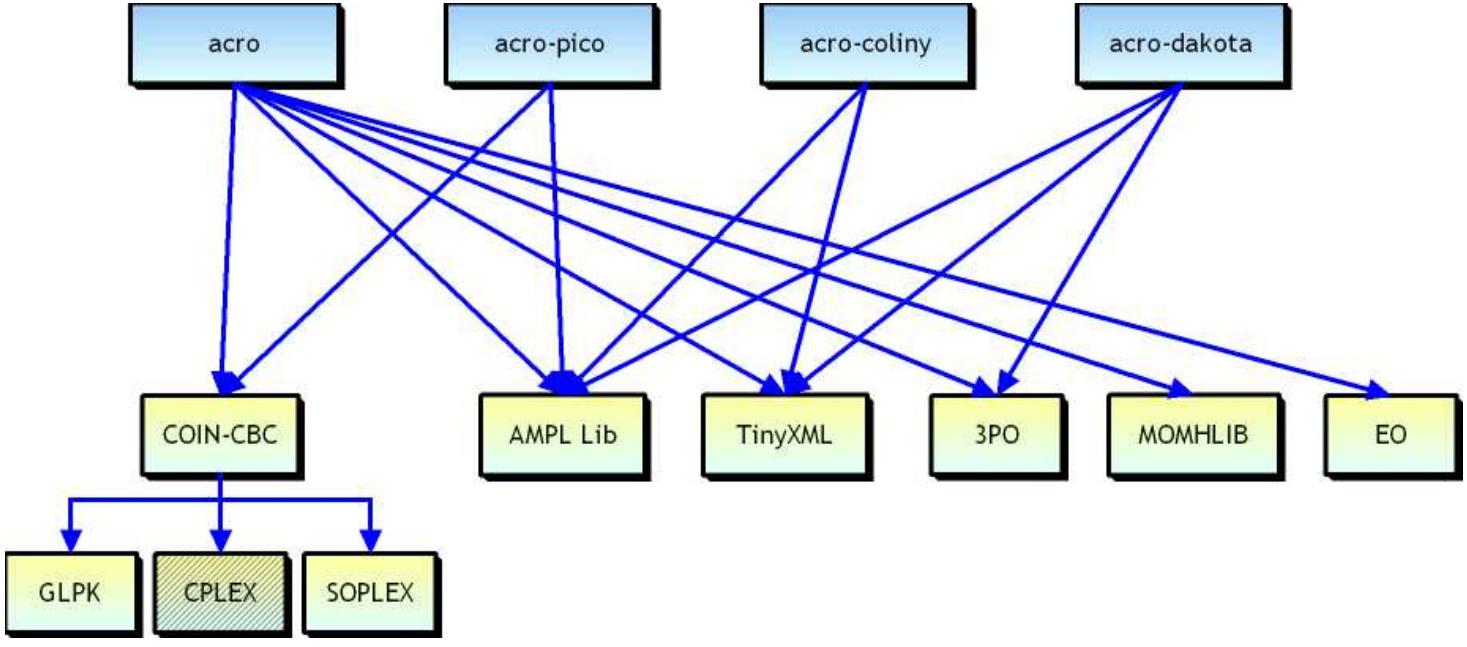
UTILIB

- C++ utility library
- Bill Hart (and many others)

Managing Third Party Libraries

Goal: integrate third party software into a common software repository

Motivation: need to integrate a diverse range of solvers for sophisticated optimization solvers



Managing Third Party Libraries (cont'd)

Acro Impact

- Management of build process
  - Each TPL has its own build process
  - Acro integration enables seamless builds
- TPL version control
  - Acro TPL integration enables consistent dependency on TPLs for Acro developers
  - Developers can try out new TPL releases without sacrificing stable builds
  - Acro Versioning of TPLs enables more facile integration
- TPL integration has promoted more direct comparisons of different sub-solvers (e.g. linear-programming solvers)

Acro Impact: Software Development

Acro integration has encouraged fine-grain software decomposition

- Example: PICO -> PEBBL + PICO
  - PEBBL branch and bound library was always generic, but couldn't easily be used in other applications
- New applications: Lipshitzian global optimizer, quadratic semi-assignment solver, GNLP solver

Acro integration has enabled hybridization of Acro solvers

- Example: GNLP solver integrates PEBBL, COIN (w/PICO LP management), and Coliny
- Example: Coliny includes COLIN solver wrappers for PEBBL Lipshitzian solver and APPSPACK solver

Acro Impact: Applications

- EPA Water Security
  - PICO deployed to optimize MILP sensor placement formulations
  - Release to water utilities planned in FY08

- DAKOTA
  - Coliny and APPSPACK software integrated to support engineering design
  - Open-source release with broad user base

- Acro
  - Command-line optimization tool that recognizes structured problem instances and supports problem transformations
  - Prototype being used within various research efforts

Acro Software Quality Management

Acro SQA Overview

Support for SQA activities has proven critical to ensuring that Acro's software is stable

- Software stability is necessary for application impact
- Software stability is necessary to support code integration

SQA Activities

- Subversion revision control
- Bugzilla bug tracking
- Nightly testing
- Analysis of code metrics
- Release checklist

Why SQA? Some history…

- 1/9/2004: Acro CVS repository created
  - *Initial motivation: integration of CVS repositories*
- 04/13/2004: Acro Bugzilla repository goes online
  - *One of the first bugs submitted is still open…* ☹
- 05/05/2004: The Acro-Regression email list is setup to archive Dakota-Acro nightly integration tests
- 02/15/2005: Nightly emails updated to summarize Acro tests in a more Acro-centric format
- 04/26/2006: Bill refuses to fix bugs that are not in Bugzilla
  - *"If a bug isn't submitted via bugzilla, don't expect me to act on it. If you send me an email, I'll kindly ask you to submit a bug report ... and then promptly delete your email."*

Acro SQA? More history…

- 5/2006: Release checklist created for the Acro 1.0 release
- 11/2006: Nightly testing results integrate EXACT software tests
  - *Used weblinks for test results stored on static web pages*
  - *Includes 'code checks' that summarize software metrics*
- 1/2007: Bill and Lee Ann converted Acro to subversion
  - *…, and there was much rejoicing!*
- 1/2008: Stef Chakerian sets up database to archive nightly test results
  - *… and then promptly quits!* ☹

SQA Lessons Learned

What SQA practices have we adapted from other code projects?

What new SQA practices have we developed for Acro?

What other capabilities do we wish we had?

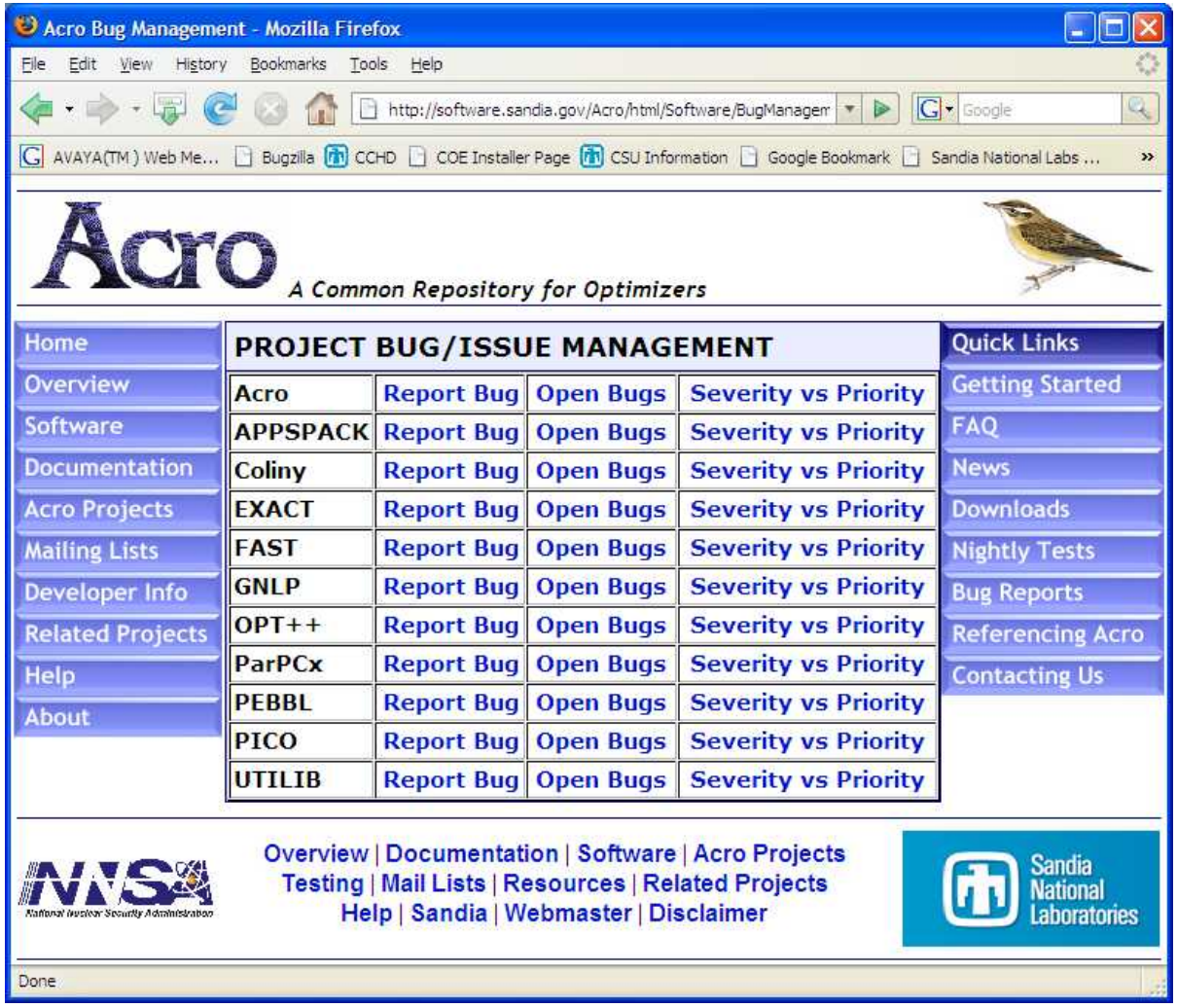
Version Control

LL: Use subversion to integrate multiple software projects

- The physical location of software is not really an issue when using this type of version control tool

Bug Tracking

LL: Make it easy for users to create and view bugs



Sandia National Laboratories

Nightly Testing

LL: Provide active reporting mechanisms like email, RSS, etc

LL: Enable quick review of testing results

LL: Make it easy to drill down to get more details

From: acro-regression-bounces@software.sandia.gov on behalf of Software Quality Engineering account [sqe@software.sandia.gov] Sent: Mon 1/21/2008 8:01 AM

To: acro-regression@software.sandia.gov

Cc:

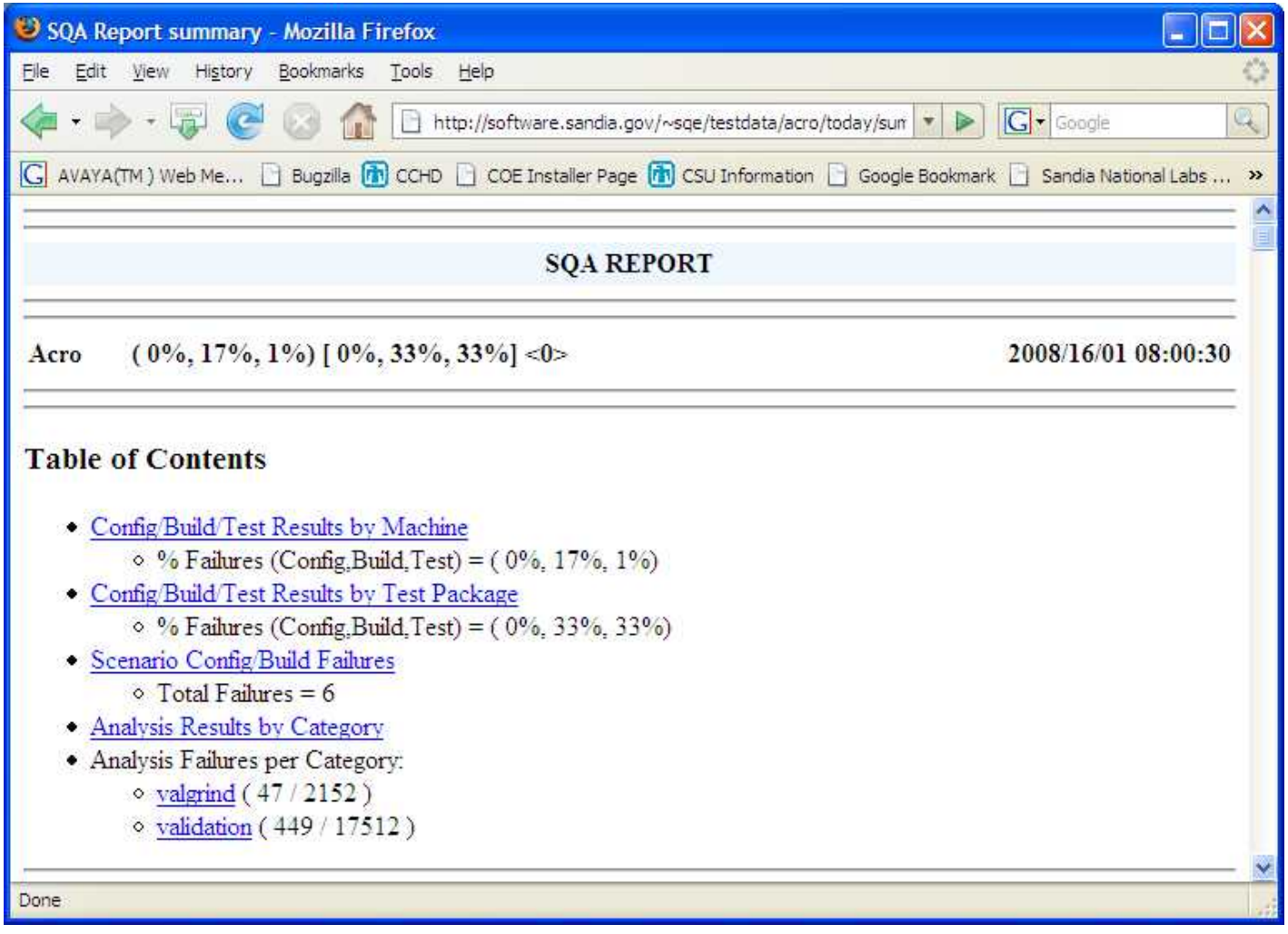
Subject: [Acro-Regression] Acro SQA Report: (0%, 20%, 1%) [0%, 35%,29%] <0>

```
**************************** SQA REPORT LINKS ****************************
**************************************************************************
 Acro (  0%,  20%,   1%)  [  0%,  35%,  29%] <0>        2008/21/01 08:01:23
**************************************************************************
**************************************************************************

 Web Link:  http://software.sandia.gov/
~sqe/testdata/acro/20080121/summary.html

 Config/Build/Test Results by Machine
      % Failures (Config,Build,Test) = (  0%,  20%,   1%)

 Config/Build/Test Results by Test Package
      % Failures (Config,Build,Test) = (  0%,  35%,  29%)

 Scenario Config/Build Failures
      Total Failures = 6

 Analysis Results by Category

 Analysis Failures per Category:
            valgrind    ( 47 / 1664 )
          validation    ( 358 / 13210 )
```

Nightly Testing

Acro tests are summarized in nightly web pages

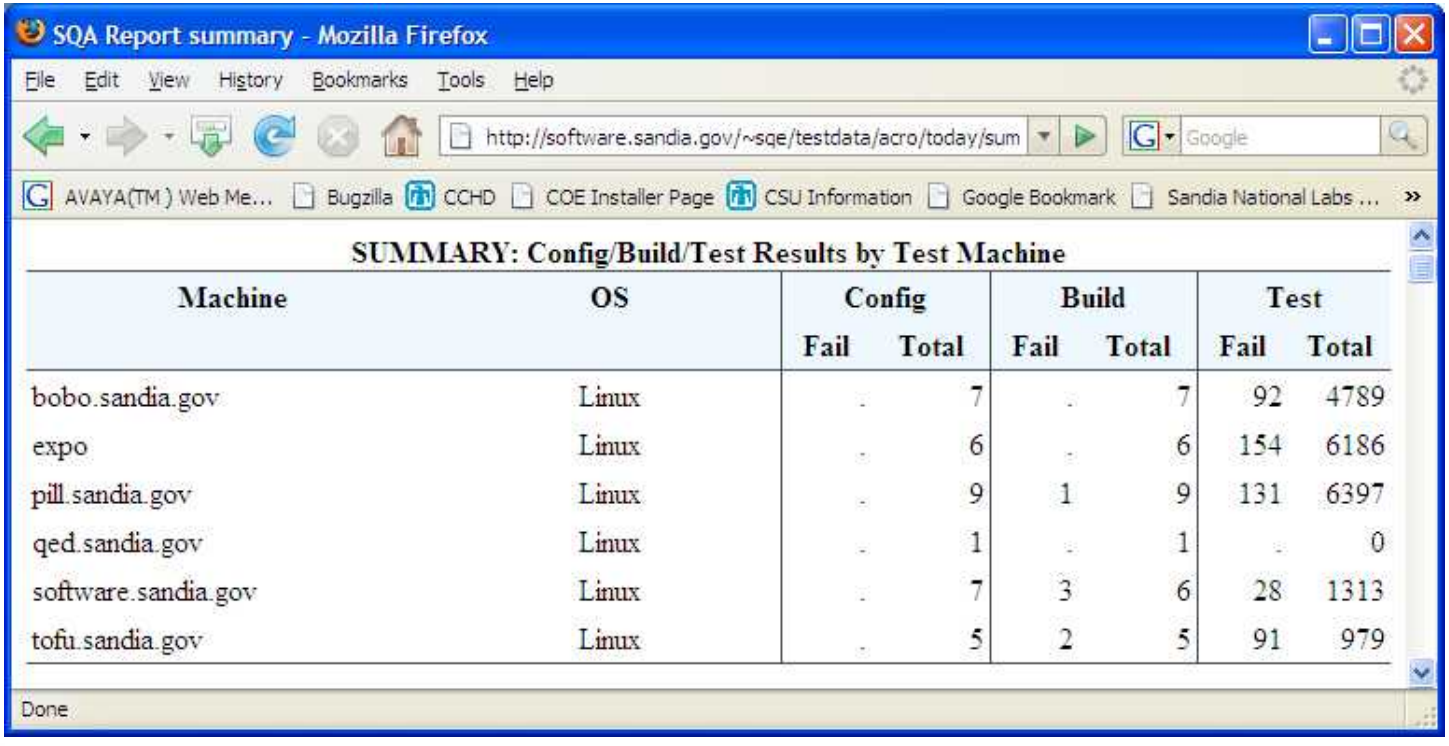
LL: archive test data in database, and use dynamic web pages



Sandia National Laboratories

Nightly Testing – Test Data

Acro test summary by machine



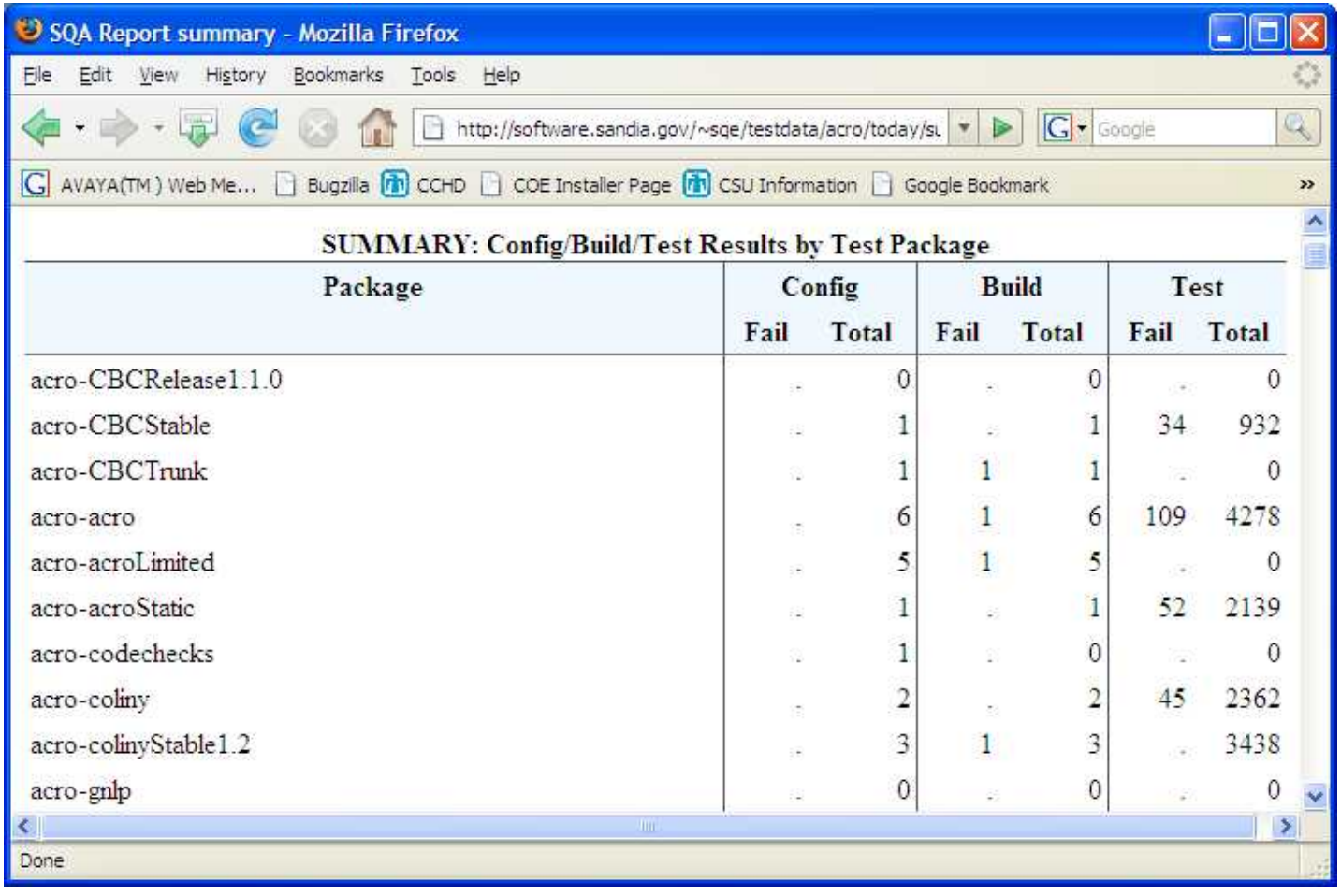
Machine	OS	Config		Build		Test	
		Fail	Total	Fail	Total	Fail	Total
bobo.sandia.gov	Linux	.	7	.	7	92	4789
expo	Linux	.	6	.	6	154	6186
pill.sandia.gov	Linux	.	9	1	9	131	6397
qed.sandia.gov	Linux	.	1	.	1	.	0
software.sandia.gov	Linux	.	7	3	6	28	1313
tofu.sandia.gov	Linux	.	5	2	5	91	979

Sandia National Laboratories

Nightly Testing - Test Data

Acro test summary by test target (using target groups)

LL: Support rich target naming conventions



SUMMARY: Config/Build/Test Results by Test Package

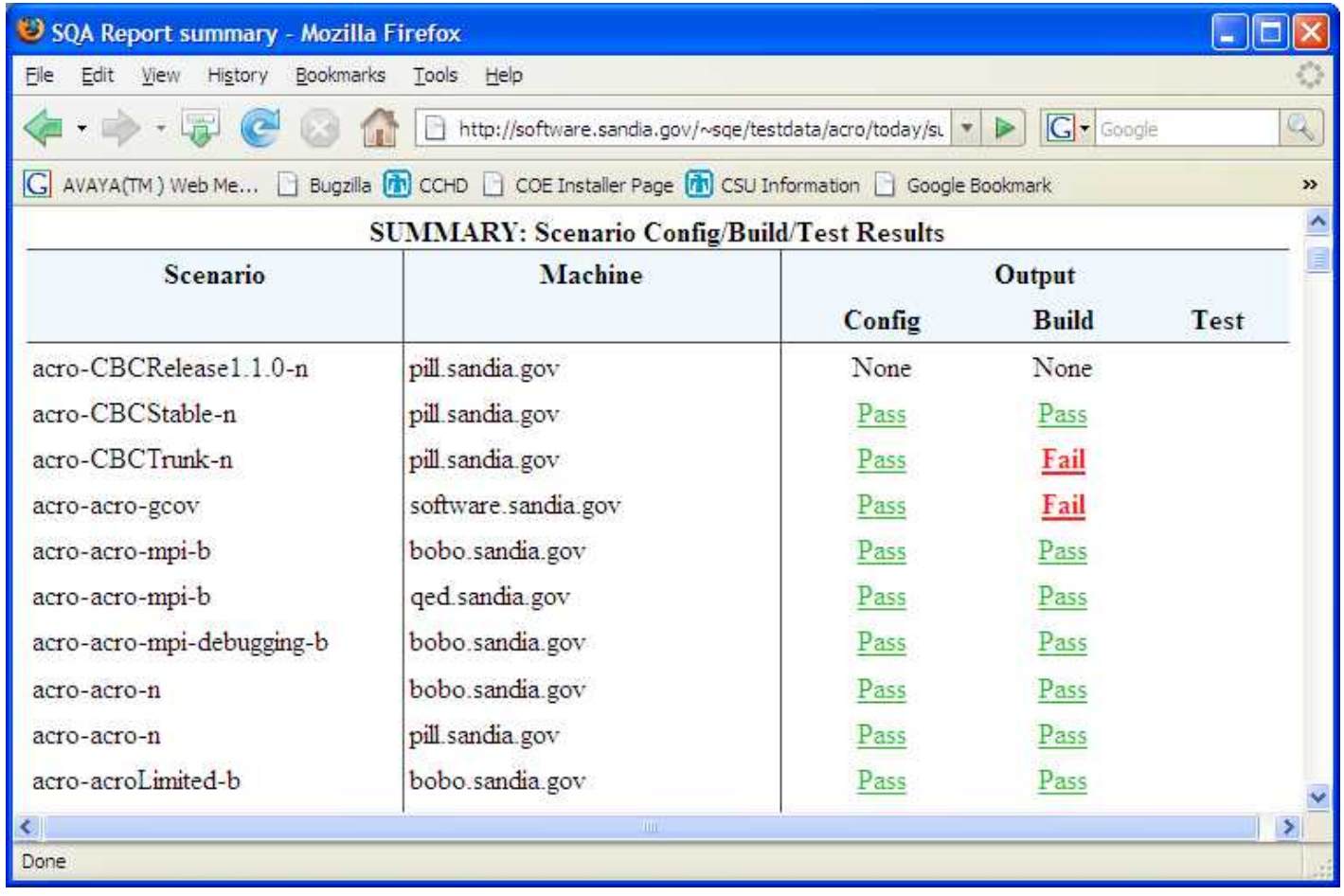
Package	Config		Build		Test	
	Fail	Total	Fail	Total	Fail	Total
acro-CBCRelease1.1.0	.	0	.	0	.	0
acro-CBCStable	.	1	.	1	34	932
acro-CBCTrunk	.	1	1	1	.	0
acro-acro	.	6	1	6	109	4278
acro-acroLimited	.	5	1	5	.	0
acro-acroStatic	.	1	.	1	52	2139
acro-codechecks	.	1	.	0	.	0
acro-coliny	.	2	.	2	45	2362
acro-colinyStable1.2	.	3	1	3	.	3438
acro-gnlp	.	0	.	0	.	0

Sandia National Laboratories

Nightly Testing - Test Data

Acro build/test details for all targets

LL: Enable quick review of testing results



SUMMARY: Scenario Config/Build/Test Results

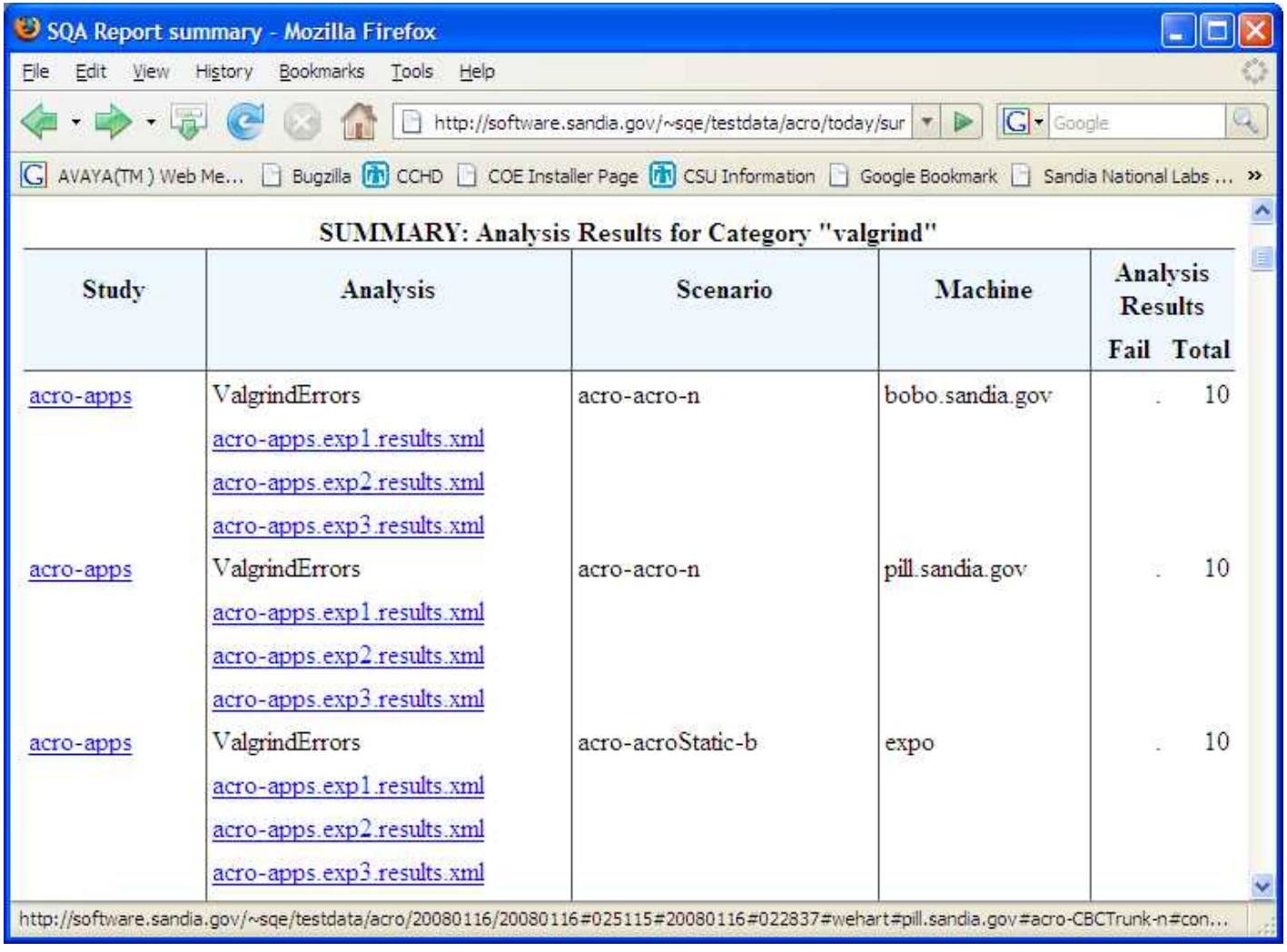
Scenario	Machine	Output		
		Config	Build	Test
acro-CBCRelease1.1.0-n	pill.sandia.gov	None	None	
acro-CBCStable-n	pill.sandia.gov	Pass	Pass	
acro-CBCTrunk-n	pill.sandia.gov	Pass	Fail	
acro-acro-gcov	software.sandia.gov	Pass	Fail	
acro-acro-mpi-b	bobo.sandia.gov	Pass	Pass	
acro-acro-mpi-b	qed.sandia.gov	Pass	Pass	
acro-acro-mpi-debugging-b	bobo.sandia.gov	Pass	Pass	
acro-acro-n	bobo.sandia.gov	Pass	Pass	
acro-acro-n	pill.sandia.gov	Pass	Pass	
acro-acroLimited-b	bobo.sandia.gov	Pass	Pass	

Sandia National Laboratories

Nightly Testing - Test Data

Acro test details by category

LL: Enable quick review of testing results



Sandia National Laboratories

Nightly Testing - Other

LL: Do not perform tests if the code has not changed

LL: Maintain artifacts from previous test results

LL: Support “expected failure” of tests

LL: Analyze code performance over time

LL: Extended computational experiments for large-scale benchmarking

Acro Code Tests

Smoke tests

- Fast tests of functionality

Functional tests

- General functionality tests

Performance tests

- Compare runtime statistics (not well supported)

Memory tests

- Run valgrind and report errors

Integration tests

- Build tests with other codes: Dackota, PDock
- Generic interpretation of config/build logs

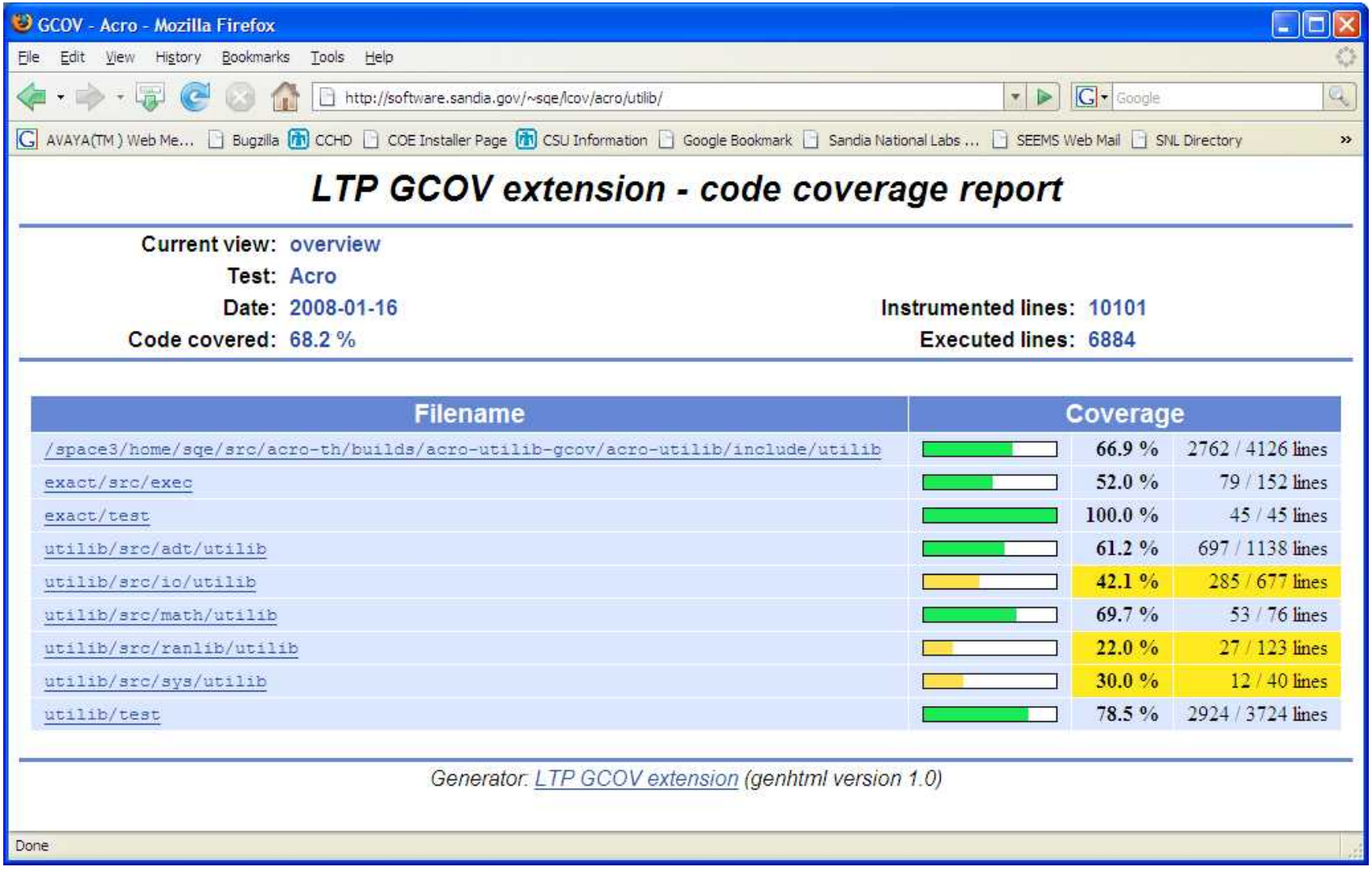
Portability tests

- Lightweight distributed build framework (weak support on PCs)

Code coverage tests

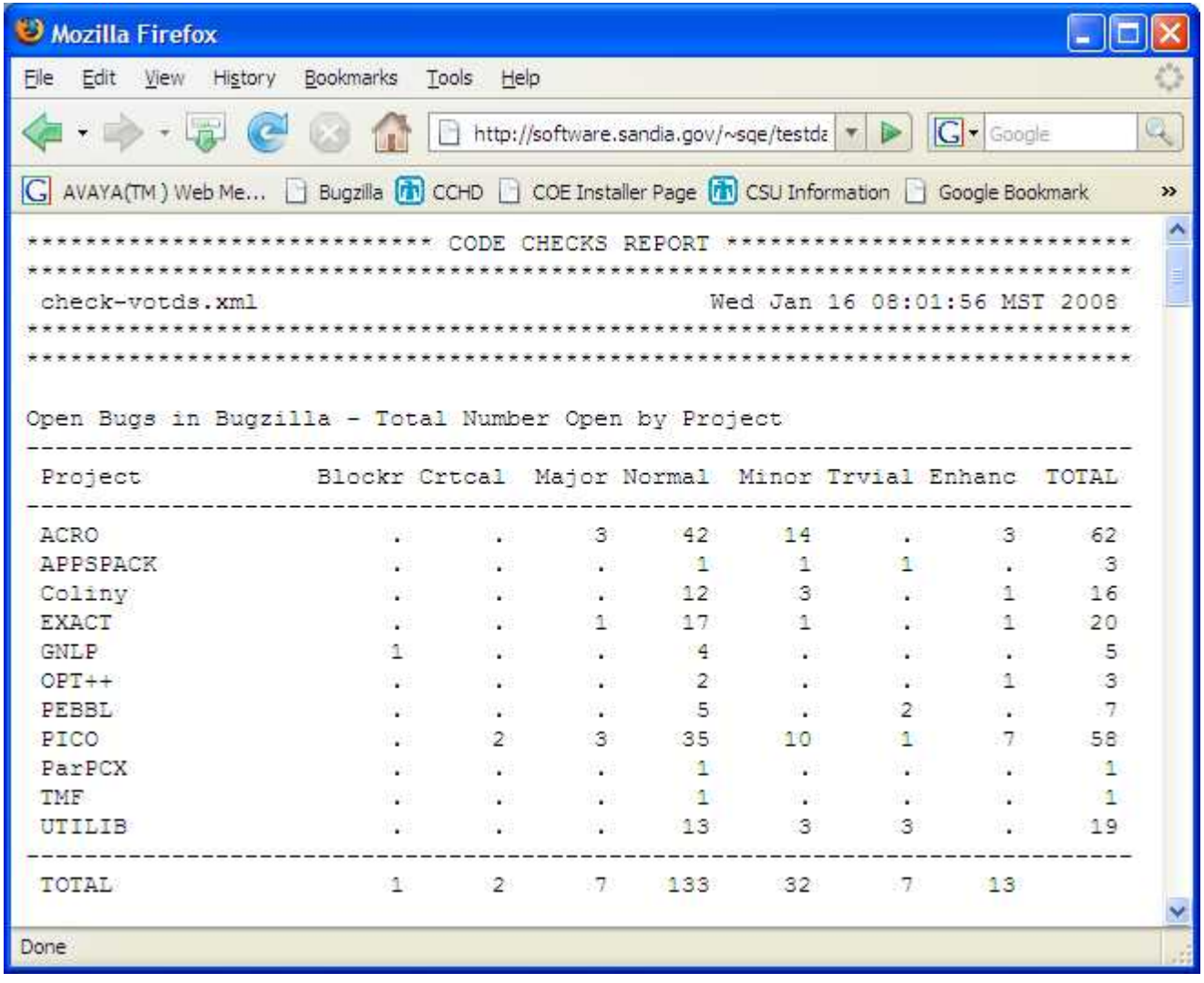
- Coverage statistics generated with gcov and lcov

Code Coverage Statistics



Code Metrics - Bugs by Project

LL: summarize bug statistics with active web pages

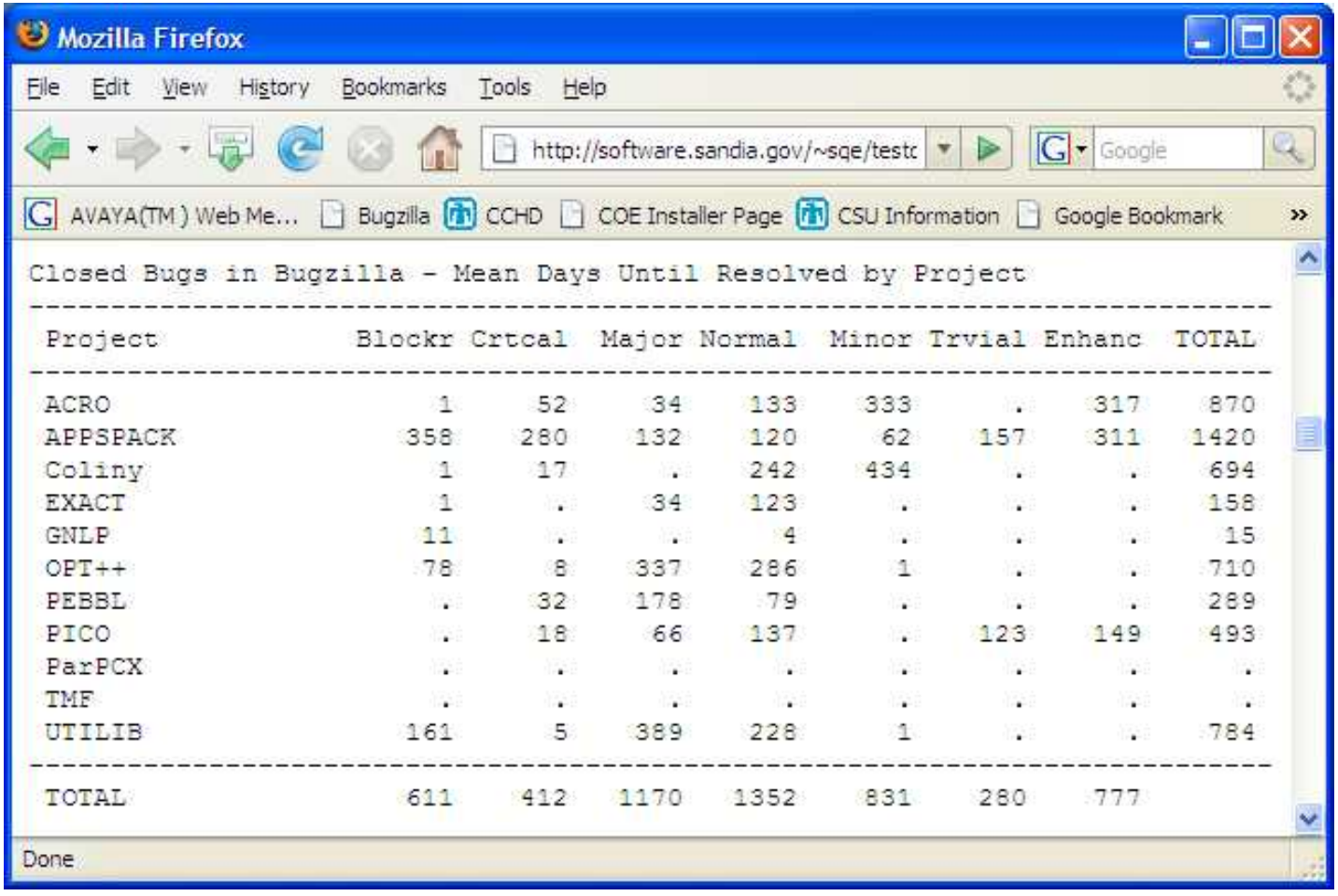


Project	Blockr	Crtcal	Major	Normal	Minor	Trvial	Enhanc	TOTAL
ACRO	.	.	3	42	14	.	3	62
APPSPACK	.	.	.	1	1	1	.	3
Coliny	.	.	.	12	3	.	1	16
EXACT	.	.	1	17	1	.	1	20
GNLP	1	.	.	4	.	.	.	5
OPT++	.	.	.	2	.	.	1	3
PEBBL	.	.	.	5	.	2	.	7
PICO	.	2	3	35	10	1	7	58
ParPCX	.	.	.	1	.	.	.	1
TMF	.	.	.	1	.	.	.	1
UTILIB	.	.	.	13	3	3	.	19
TOTAL	1	2	7	133	32	7	13	

Sandia National Laboratories

Code Metrics - Time to Fix Bugs

LL: clear out unfixable bugs



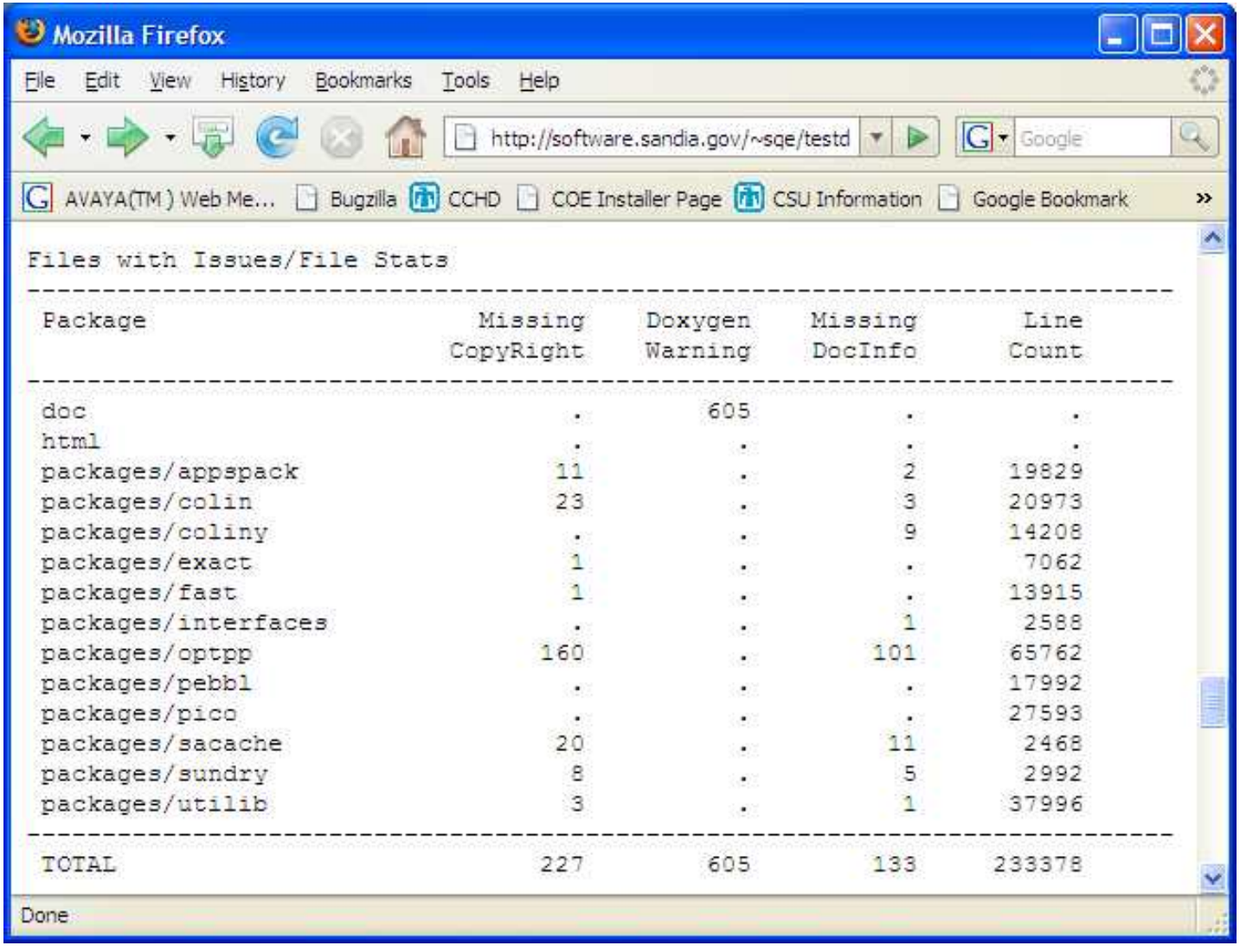
Closed Bugs in Bugzilla - Mean Days Until Resolved by Project

Project	Blockr	Crtcal	Major	Normal	Minor	Trvial	Enhanc	TOTAL
ACRO	1	52	34	133	333	.	317	870
APPSPACK	358	280	132	120	62	157	311	1420
Coliny	1	17	.	242	434	.	.	694
EXACT	1	.	34	123	.	.	.	158
GNLP	11	.	.	4	.	.	.	15
OPT++	78	8	337	286	1	.	.	710
PEBBL	.	32	178	79	.	.	.	289
PICO	.	18	66	137	.	123	149	493
ParPCX	.	.	.	.	.	.	.	.
TMF	.	.	.	.	.	.	.	.
UTILIB	161	5	389	228	1	.	.	784
TOTAL	611	412	1170	1352	831	280	777	

Sandia National Laboratories

Code Metrics - Issues for Releases

LL: proactively identify issues that need to be resolved



Files with Issues/File Stats

Package	Missing CopyRight	Doxygen Warning	Missing DocInfo	Line Count
doc	.	605	.	.
html	.	.	.	.
packages/appspack	11	.	2	19829
packages/colin	23	.	3	20973
packages/coliny	.	.	9	14208
packages/exact	1	.	.	7062
packages/fast	1	.	.	13915
packages/interfaces	.	.	1	2588
packages/optpp	160	.	101	65762
packages/pebbl	.	.	.	17992
packages/pico	.	.	.	27593
packages/sacache	20	.	11	2468
packages/sundry	8	.	5	2992
packages/utilib	3	.	1	37996
TOTAL	227	605	133	233378

Sandia National Laboratories

Release Checklist

LL: Need to adapt release process to the software level of formality

- Simple checklist has proven sufficient for Acro

LL: The master checklist should clearly indicate whether other checklists are needed

- This significantly simplifies the release process

LL: Nightly test artifacts should be archived to support releases

- Acro does this, but not in a convenient manner

Testing with EXACT and FAST

The EXACT and FAST Projects

Tools for doing experimental testing

Overview:

- EXACT supports generic experimental design and analysis
- FAST supports a generic mechanism for nightly testing and data gathering

History:

- Earlier versions were tightly integrated into Acro's testing mechanism
- Recent generalization of EXACT into a separate python module (based on J. Berry's earlier work)
- Recent generalization of FAST from Acro, supporting general-purpose clients and servers for nightly testing and code evaluations.

EXACT Overview

GOAL: Provide a software framework for defining and analyzing computational experiments

- Managing computational experiments
  - Systematic control is needed for large-scale experimentation
  - Design of experiments to limit the cost of experimentation
  - Archiving experimental results in a standard manner
  - Integration of statistical analysis capabilities

- Applications
  - Experimental evaluation of heuristics
  - Comparisons between algorithms
  - Robust (user) parameter settings (over many problem domains)

Overview (Motivation Continued)

- Software testing
  - Automation of tests
  - Flexible notion of what a “test” means
  - Integration with diagnostic tools (e.g. valgrind, lcov)
  - Distributed test management and test summary

Observation: testing of large complex software begins to look like a computational experiment

Example: integer programming solver

- Lots of algorithmic parameters
- Lots of hard test problems
- Costly tests

XML Description

```
<experimental-study name="example1">
  <tags>
    <tag> example </tag>
  </tags>

<experiment name="ht">
  <factors>
    <factor name="hashfn">
      <level> Jenkins </level>
      <level> FNV     </level>
    </factor>
  </factors>
```

XML example continued

```
<controls>
   <executable> hash_script </executable>
</controls>
</experiment>

<analysis name="LoadFactorUB" type="validation">
   <data experiment = "ht"/>
   <options> _measurement = LoadFactor
             _value       = 0.75
   </options>
</analysis>

</experimental-study>
```

EXACT Measurement File

```
"Number of Evaluations" numeric/integer 110
"Best Value" numeric/double 0.0001231
"Termination Condition" text/string "Max Evals Limit"
exit_status numeric/integer 0
```

XML Specification with Experimental Options

```
<factors>
  <factor name="search">
    <level> </level>
    <level>initialDive=true</level>
    <level>initialDive=true integralityDive=true</level>
  </factor>
  <factor name="problem">
    <level>_data=bm23 _optimum=34 _opttol=1e-8</level>
    <level>_data=p0033 _optimum=3089 _opttol=1e-6</level>
  </factor>
</factors>
```

The FAST Project

Overview:

- FAST supports a generic mechanism for nightly testing and data gathering
- Supports general-purpose clients and servers for nightly testing and code evaluations.
- Uses CVS commits to work around restrictive firewalls

Impact

- General framework for coordinating nightly builds
- Supports “code checks” – analyses that assist in SW management
  - Bugzilla summaries, commit activity, copyright documentation, analysis of subversion externals

EXACT/FAST Impact

- Software testing
  - Being used to manage computational tests for several code projects: DAKOTA, Acro, SPOT, Zoltan, …
- Interactive experimentation
  - Being used to for computational experiments in ongoing research
- Bug diagnosis
  - Has found “bugs” not reported by the previous testing techniques in Acro and Zoltan
  - Nightly archive has been useful for archealogical debugging

Future Directions

Bill's Wishlist

- Database for experimental results
  - Analyze performance over time
- Active web pages based on database results
- Rework or replace FAST
  - Use buildbot?
  - Explicit promotion of test artifacts for releasee
- Rework of EXACT functionality
  - More DOE tools and DOE analysis
  - Randomization of experiments, blocking, etc.
  - Incremental tests
  - Robust DOE that allow for test failures

Thanks!

See http://software.sandia.gov/Acro for more details...

Sandia National Laboratories