

ABRT Project Document

Karel Klíč, Michal Toman, Miroslav Lichvár

May 11, 2012

ABRT is a set of technologies that collects and evaluates software crashes coming from Linux operating system deployments.

This is a top-level project document for ABRT.

TODO: What is ABRT doing.

TODO: Purpose of ABRT.

Contents

1	Project Charter	4
1.1	Statement of Work	4
1.2	Business case	4
1.2.1	Enterprise Client-Side Use Case 1	4
1.2.2	Enterprise Client-Side Use Case 2	5
1.2.3	Enterprise Support Use Case	6
1.2.4	Fedora	6
1.2.5	Related Work	6
2	Stakeholders	8
3	Requirements	9
4	Scope	10
5	Work breakdown structure	10
5.1	ABRT Server Overview	10
5.2	ABRT Server Services Overview	12
5.3	ABRT Server Storage Overview	13
5.3.1	Problem Storage	14
5.3.2	Problem Storage – Reports	15
5.3.3	Problem Storage – Report backtraces	16
5.3.4	Problem Storage – Report packages	16
5.3.5	Problem Storage – Report security aspects	17
5.3.6	Problem Storage – Report in-depth information	17
5.3.7	Problem Storage – Report history	18
5.3.8	Problem Storage – Symbols	18
5.3.9	Problem Storage – Clusters	18
5.3.10	Data Storage – Builds	19
5.3.11	Data Storage – Packages	20
5.3.12	Data Storage – Red Hat Bugzilla	21
5.3.13	Data Storage – Red Hat Bugzilla Users	22
5.3.14	Data Storage – LLVM Bitcode	22
5.3.15	Sanity Checker – Debuginfo checker	22
5.4	ABRT Server Interface Overview	23
5.4.1	Summary Page	24
5.4.2	Problems Overview Page	25
5.4.3	Problems Item Summary Page	26
5.4.4	Reports Overview Page	27
5.4.5	Reports List Page	28
5.4.6	Server Tasks Page	28
5.4.7	Server Status Page	29
5.4.8	Problem Reporting Interface	30
5.5	ABRT Client Overview	32
5.6	ABRT Client Interface Overview	33
5.6.1	Apology Dialog	34
5.6.2	Detailed Report Dialog	35
5.6.3	Applet	35
5.6.4	Problem Browser	36

6	Project Time Management	36
6.1	Fedora 17 Phase	36
6.1.1	Sprint 1	36
6.1.2	Sprint 2	37
6.1.3	Sprint 3	37
6.2	Fedora 18 Phase	37
6.3	Activity List	37
7	Project Quality Management	38
8	Project Risk Management	38
9	Project Management Plan	38

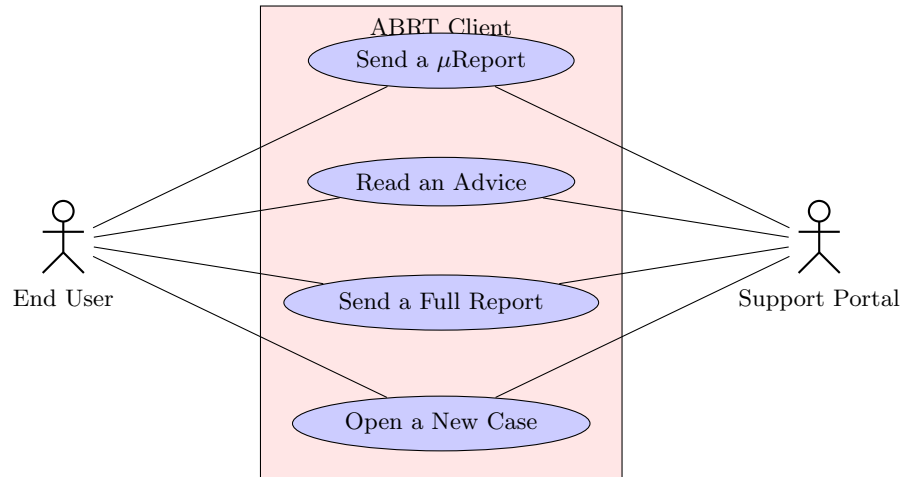
1 Project Charter

1.1 Statement of Work

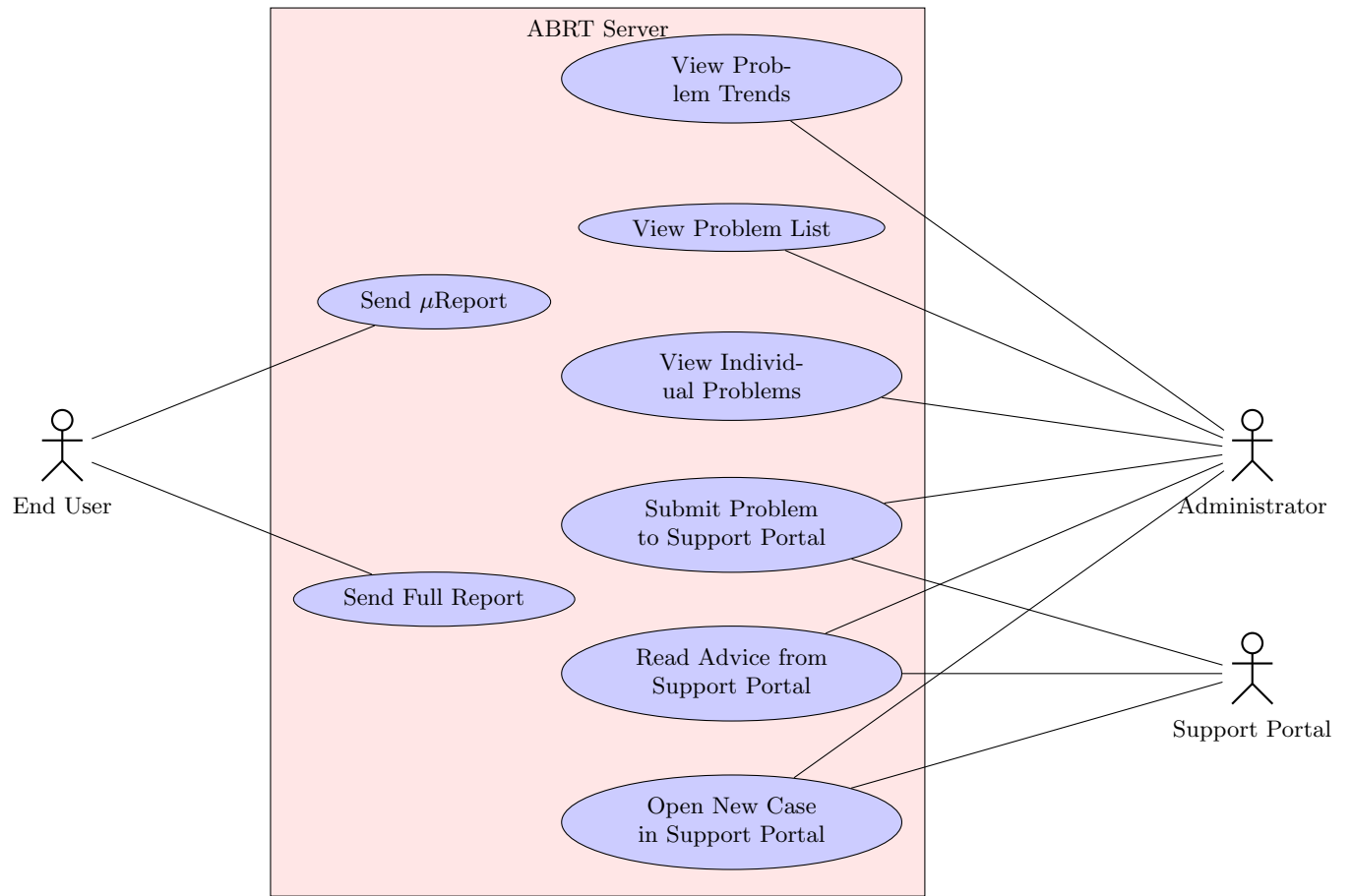
business need, product scope, strategic plan

1.2 Business case

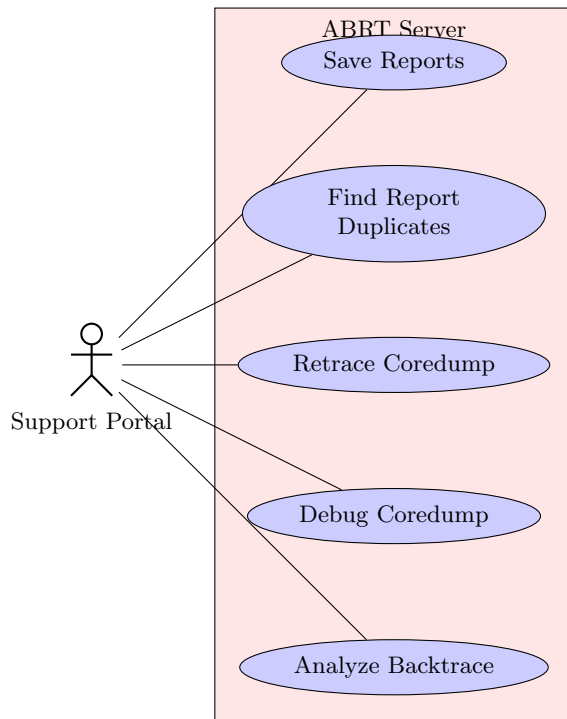
1.2.1 Enterprise Client-Side Use Case 1



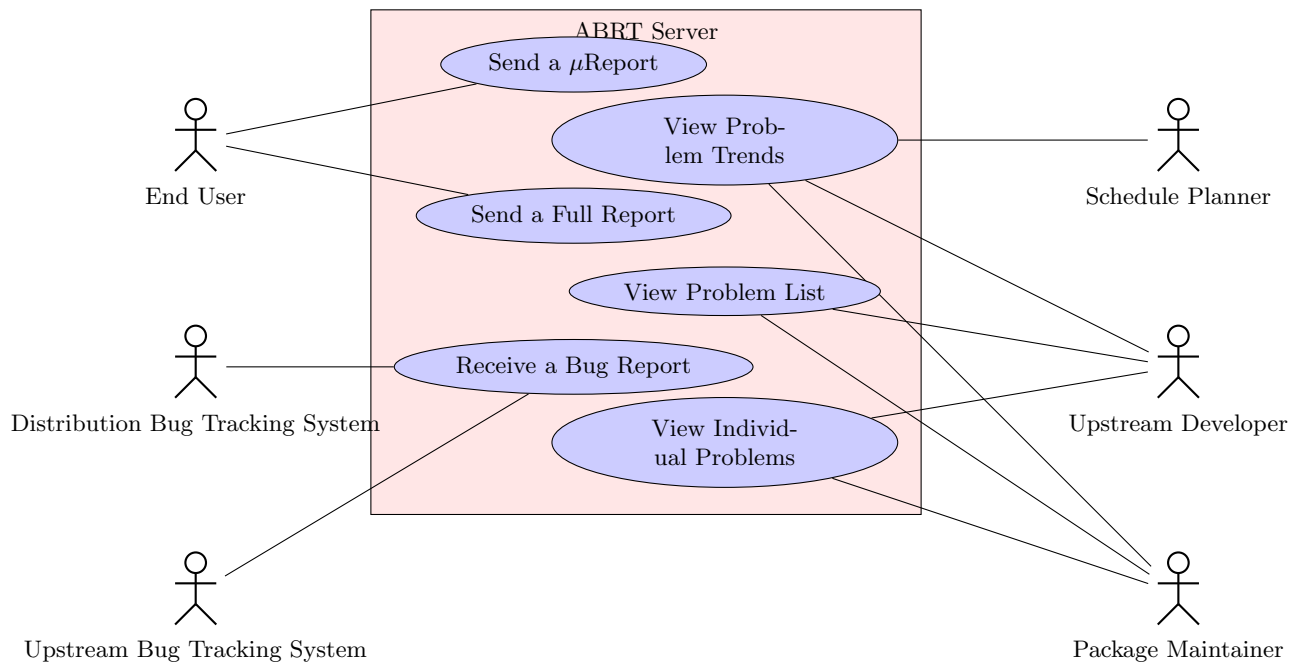
1.2.2 Enterprise Client-Side Use Case 2



1.2.3 Enterprise Support Use Case



1.2.4 Fedora



1.2.5 Related Work

There are several existing implementations of problem management systems.

TODO: [4].

Jon McCann of GNOME team envisioned a problem reporting architecture[3] that splits the responsibilities of a problem management system into several components:

1. *System Logger* collects data for anomalous behavior of system such as crash dumps and SELinux access denial logs. It is proposed to include this functionality into **systemd**, a system and service manager for Linux. In internal communication, Jon also proposed to stop using core dumps in favor of minidumps.
2. *Problem Detector* watches the output of System Logger for new events, and runs Report Generator on every new event. Its name should be **problem**d, and this tool is not implemented yet.
3. *Report Generator* gathers supplementary details about a problem, and stores problem data to a non-volatile memory. It should be handled by either **systemd** or **problem**d.
4. *User Problem Notifier* notifies user about a problem. Jon proposes to include this functionality into **gnome-settings-daemon**.
5. *Reporting Mechanism* delivers problem report to a Collection Server, scp, ftp, email...
6. *Problem Reporting and Review* shows problem reports of a system, and allows report submission. Jon designed *Oops!*[1], a graphical user interface for such a component.
7. *Problem Report Collection Server* accepts anonymous submissions, supports filling reports to Bugzilla, scrubs sensitive data from reports, detects duplicates, performs coredump analysis and retracing (generating backtrace from coredump). In internal communication, GNOME team proposes using Socorro, a crash statistics server project of Mozilla, in this role.

Advantages and good aspects of the proposal

1. The idea of pushing generic code to packages that are being used across distributions. Coredump catching can definitely be done by **systemd**. *Problem Detector* and *Report Generator* can also be made portable and shareable across distributions, and it could live in <http://freedesktop.org> as **problem**d.
2. The design of *Oops!*.

Criticism of the proposal

The most important point that should be re-evaluated is the proposed direction of *statistics-based* problem management and bugfixing. This includes the usage of minidumps in all situations, and the deployment of Socorro.

Data from the current problem management system shows that statistics-based bugfixing misfits the operating system level use case (management of full stack of applications). Statistics-based bugfixing is based on the assumption of large amounts of users hitting and reporting the same crash. This assumption is valid for desktop applications, which are quickly changing, contain large amount of bugs (GUI code is difficult to handle well), and have large amount of users (desktop shell, internet browser, e-mail client). Nevertheless, this assumption is *invalid* for most of operating system packages, including server-side software. Many packages are used by relatively low number of people, customized and deployed just on a few servers. Crashes in this situation are less frequent, but dangerous.

1. Implementing *User Problem Notifier* into **gnome-settings-daemon** brings the requirement to implement the same functionality for other desktops as well (KDE, XFCE).
2. *Oops!* design is incomplete. It is not clear how the reporting target can be configured (problem report collection server URL, e-mail, FTP, SCP destinations). The window with report details is not presented.

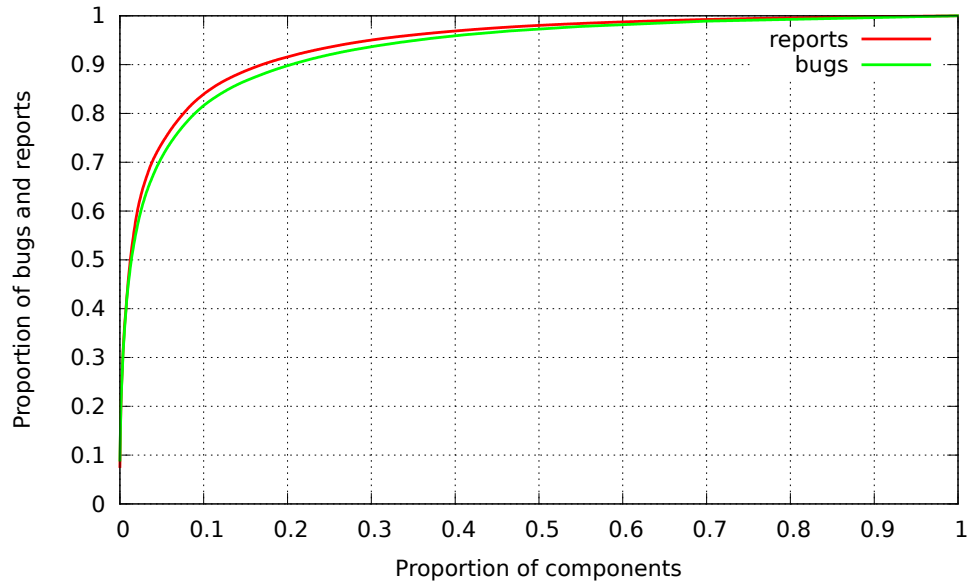


Figure 1: Cumulative distribution of ABRT bugs and reports per component

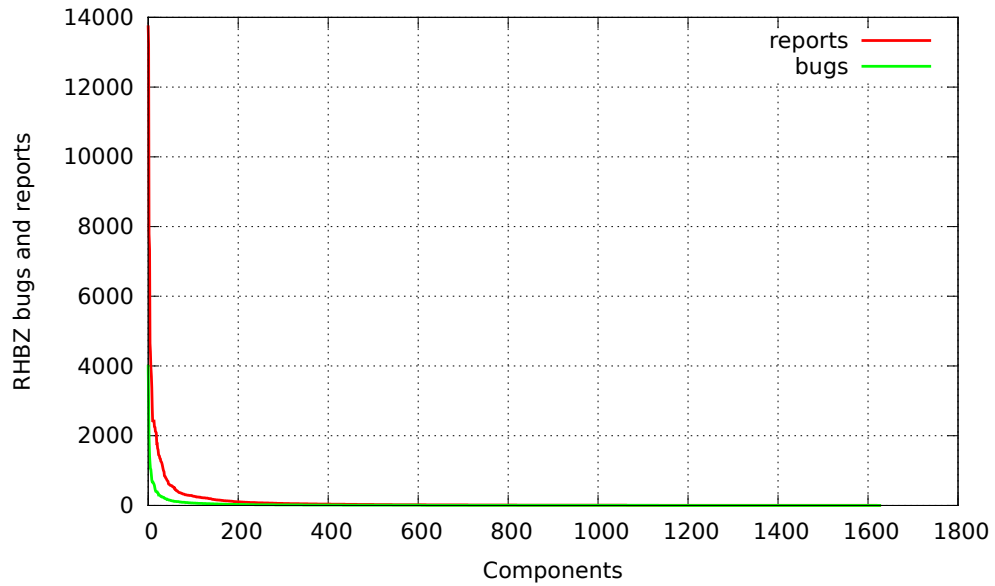
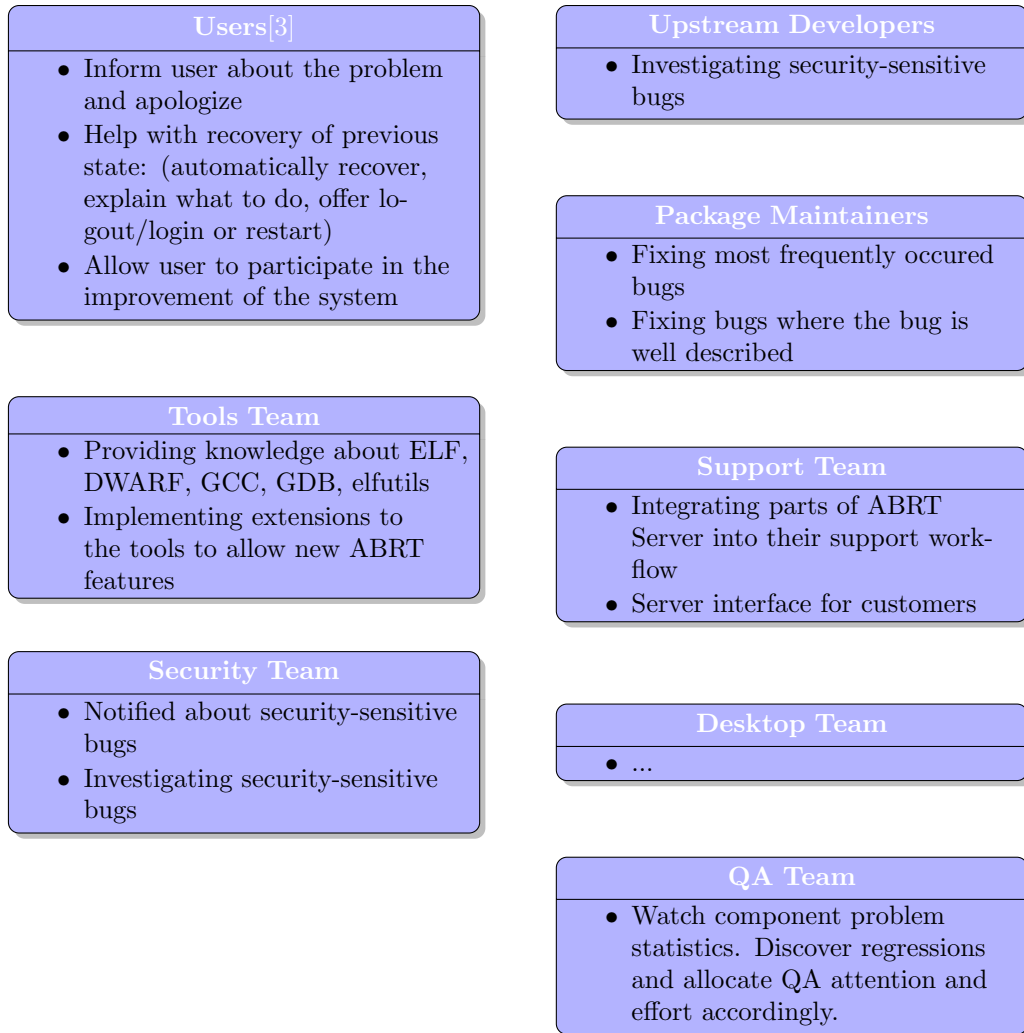


Figure 2: Distribution of ABRT bugs and reports per component

2 Stakeholders

Here is a list of people impacted by the project, and relevant information regarding their interests, involvement.



Stakeholder register
Stakeholder management strategy

3 Requirements

Large-scale data collection —————
 Collect anonymous small reports
 Show quality of operating system - overall quality of a release over time - quality of a single package
 Highlight important reports - some reports require fixing soon
 In-depth bug fixing —————
 Collect core-dump for frequently occurred reports
 Retrace coredumps on demand
 Allow interactive support for debugging of coredumps
 Developer support —————
 Analyze source code around a crash and find the problem
 Provide source code browser
 Communication —————
 Report only problems with enough information

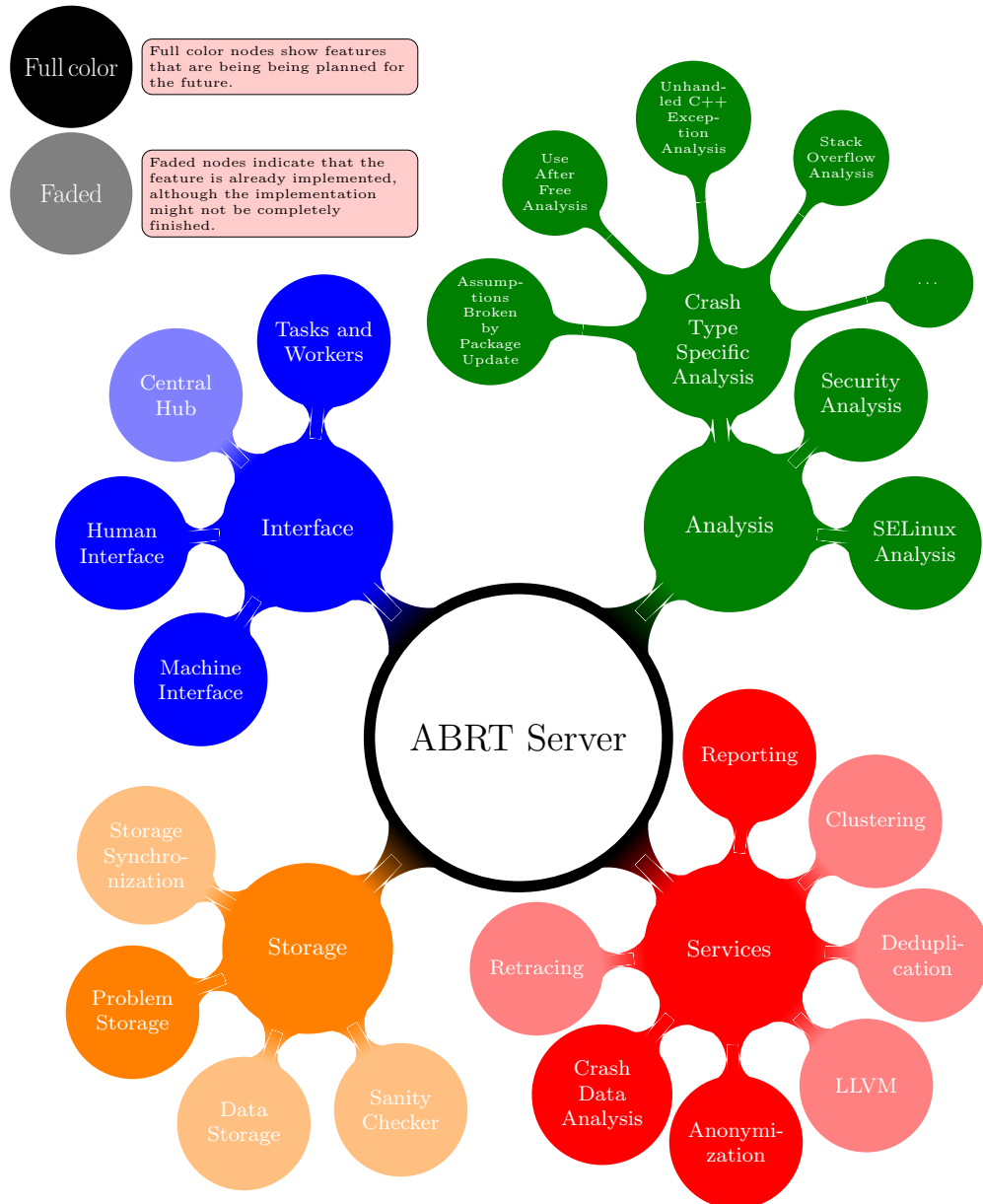
Report to operating system bug tracker
 Report to upstream bug tracker
 Watch bug resolution in remote tracker

4 Scope

5 Work breakdown structure

Deliverable oriented decomposition of server project into smaller components.

5.1 ABRT Server Overview



Storage Server's storage is a combination of a database and a file server.

Storage Synchronization Fetches data from external systems, such as RPMs and builds from Koji, bugs, comments, attachments from Red Hat Bugzilla, components, maintainers, releases from Fedora Package Database.

Problem Storage We store all issues reported by users here. Problems can be program crashes, uncaught Python exceptions, Kernel oopses, VM cores, and SELinux denials.

Data Storage We download and store RPMs of all versions of all packages in operating system. This is necessary for correct retracing of both coredumps and minidumps. We require both binaries and debugging information. Static analysis requires data files from RPMs for greater accuracy.

Sanity Checker We measure quality of data (builds, RPMs, bugs) downloaded from Fedora Project. We found quality measurement and defect detection to be necessary to keep any service operational. For example, many packages in Fedora and RHEL lacked proper debugging information, and this issue blocked retracing of many coredumps. We started to track the quality of debugging information and watch for regressions.

Services Separate services working on the top of Problem Storage and Data Storage. They are triggered by creating a new issue, changing the state of the issue, or at a certain time interval.

Retracing Generates full backtrace from a coredump stored in Problem Storage. Generates function names from a minidump (coredump-level backtrace) stored in Problem Storage.

Crash Data Analysis Depends on Retracing.

Anonymization Depends on Crash Data Analysis.

LLVM Depends on Anonymization.

Deduplication Deduplication happens on several levels: minidumps (coredump-level backtraces) are compared when receiving a new issue. Backtraces and/or function names (we call the list of function names from the crash thread *optimized backtrace* in Faf) are compared when Retracing step is done. Analyzed issues are compared when the Crash Type Specific Analysis is done.

Clustering Clustering finds clusters of issues that are close (similar) to each other. Clusters are created from multiple distances. They are used to determine possible components and even program functions that are the root cause of the bug.

Reporting

Analysis Based on static analysis techniques applied to LLVM bitcode, the Analysis framework investigates reported issues and provide insight at the source code level.

Crash Type Specific Analysis

Security Analysis

SELinux Analysis

Interface World-facing communication is implemented in the web-based Human Interface, and JSON-based Machine Interface. Internal interactions between server parts are organized as tasks performed by workers. Task queue is managed by central hub, which also provides the public communication interfaces.

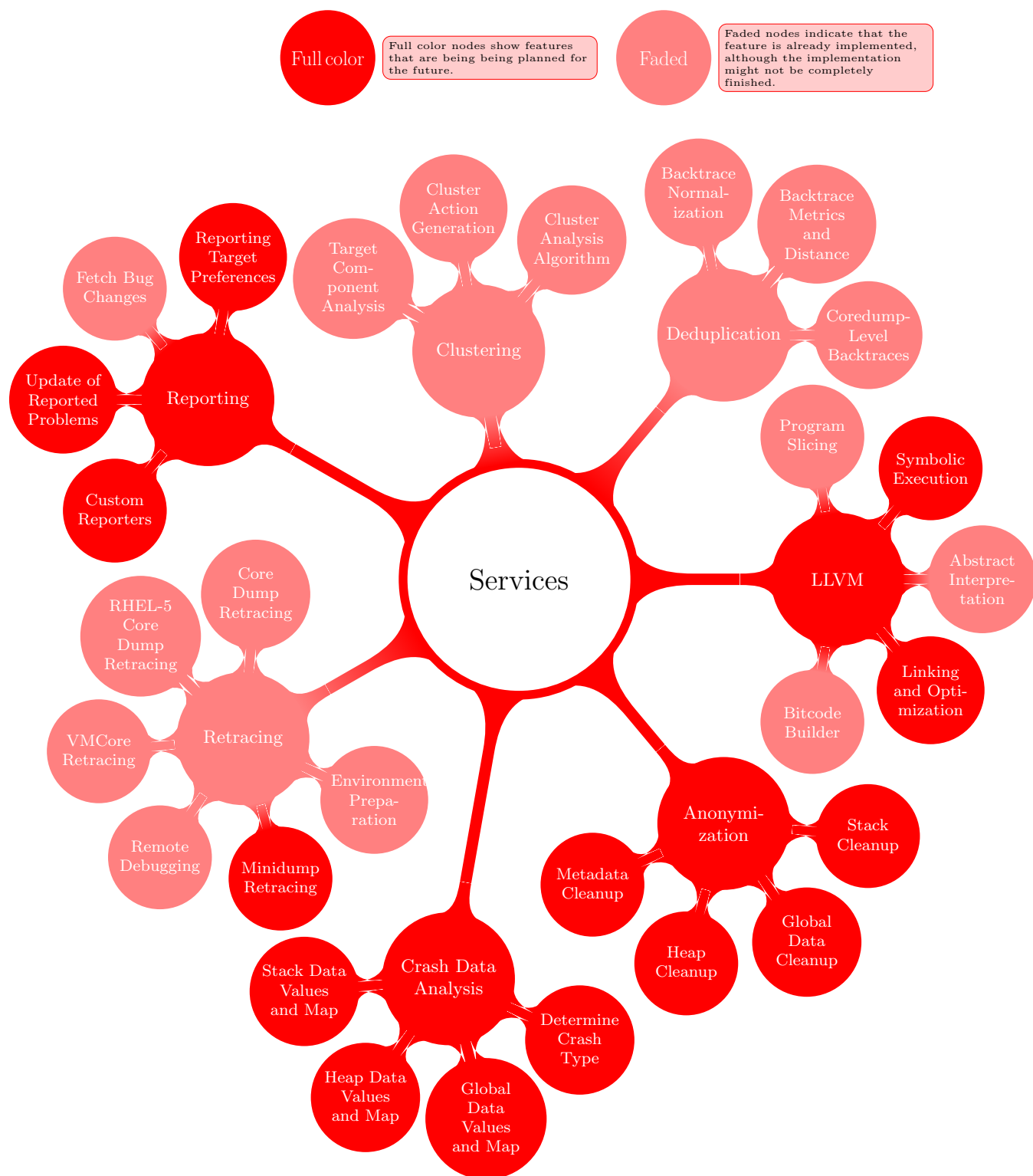
Human Interface

Machine Interface

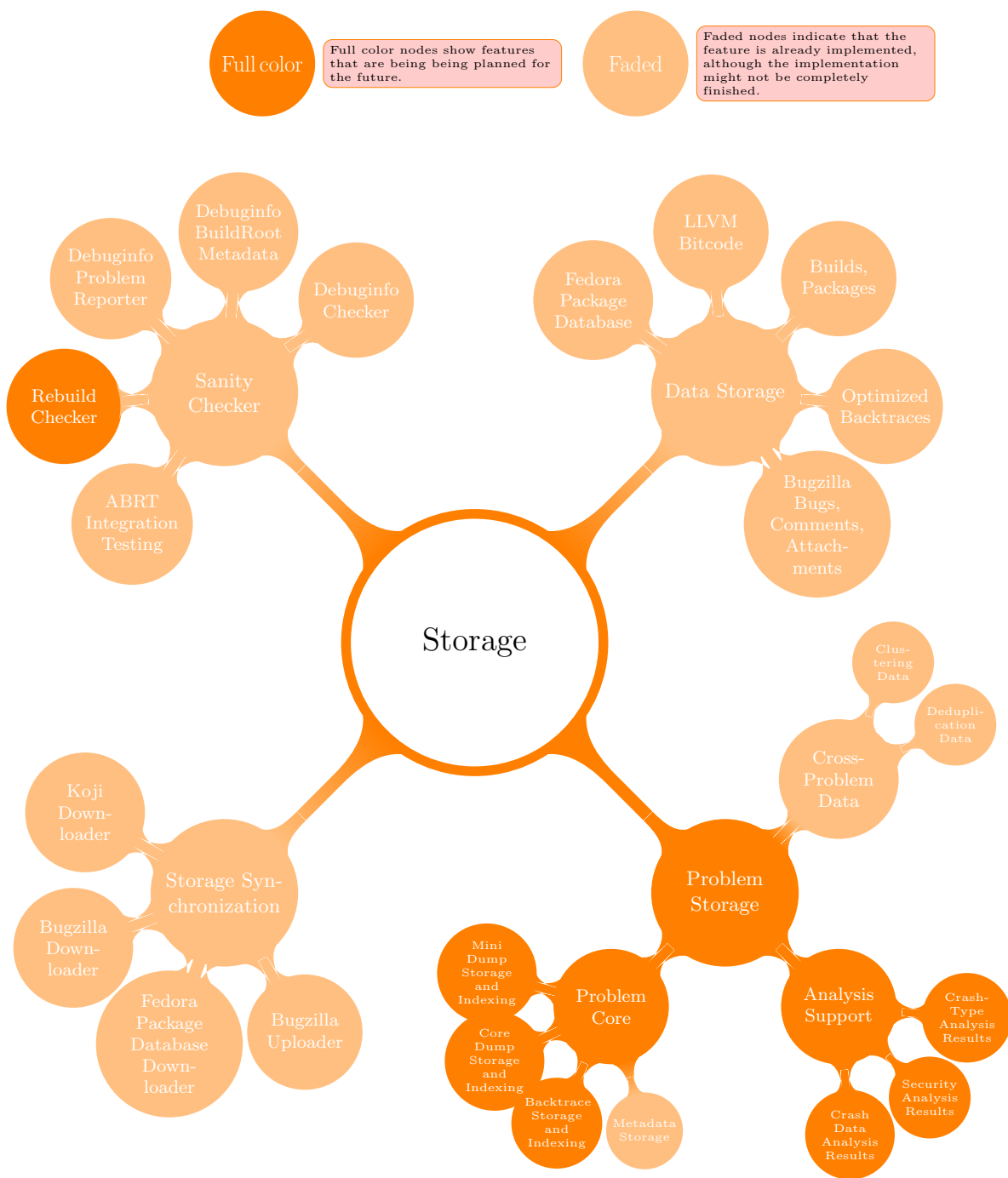
Central Hub

Tasks and Workers

5.2 ABRT Server Services Overview



5.3 ABRT Server Storage Overview



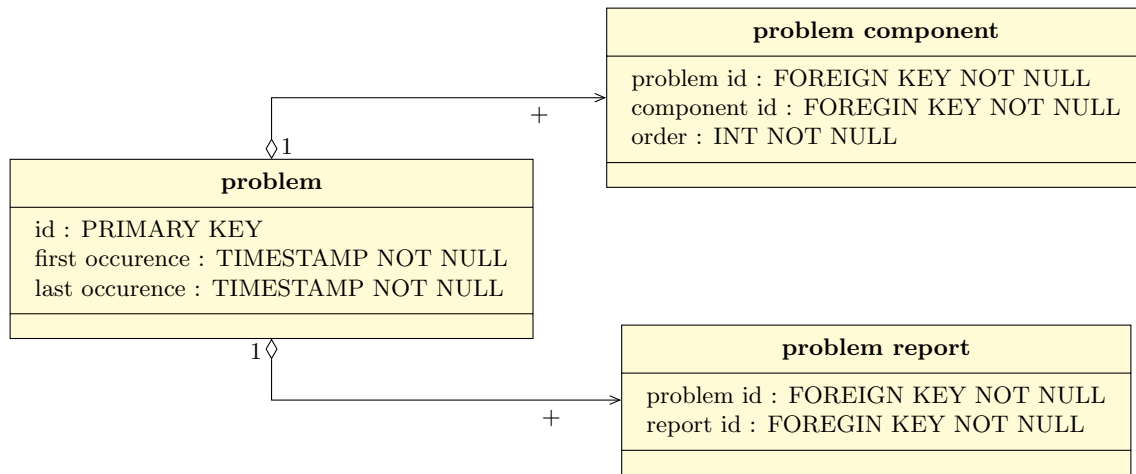
Storage Synchronization Bugzilla Downloader Downloads server-related bug reports from Red Hat Bugzilla.

Data Storage Database and file storage for data required for the analysis, evaluation, and processing of reports and problems.

LLVM bitcode We store LLVM bitcode of every binary and dynamic library compiled from C/C++

source code.

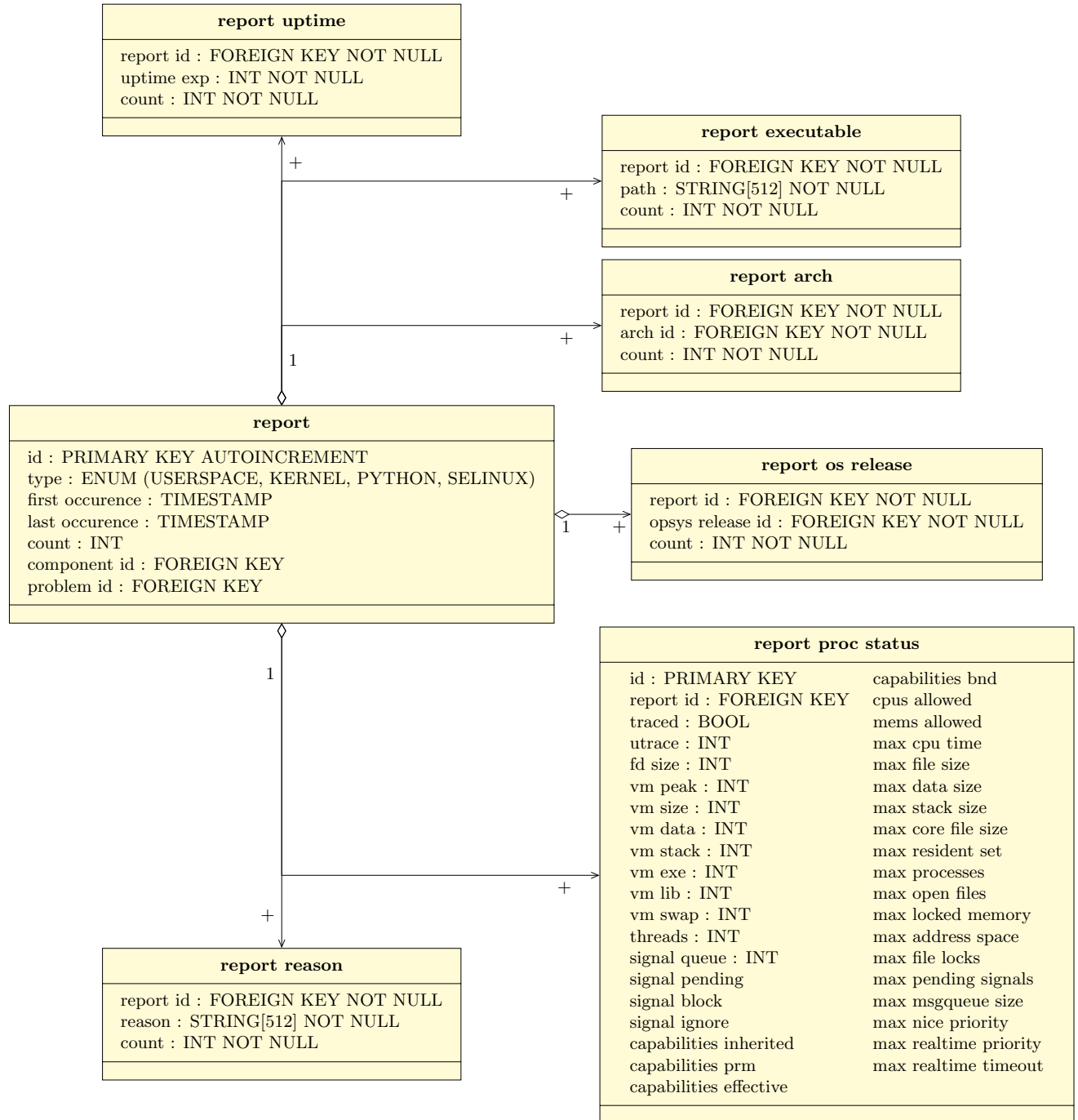
5.3.1 Problem Storage



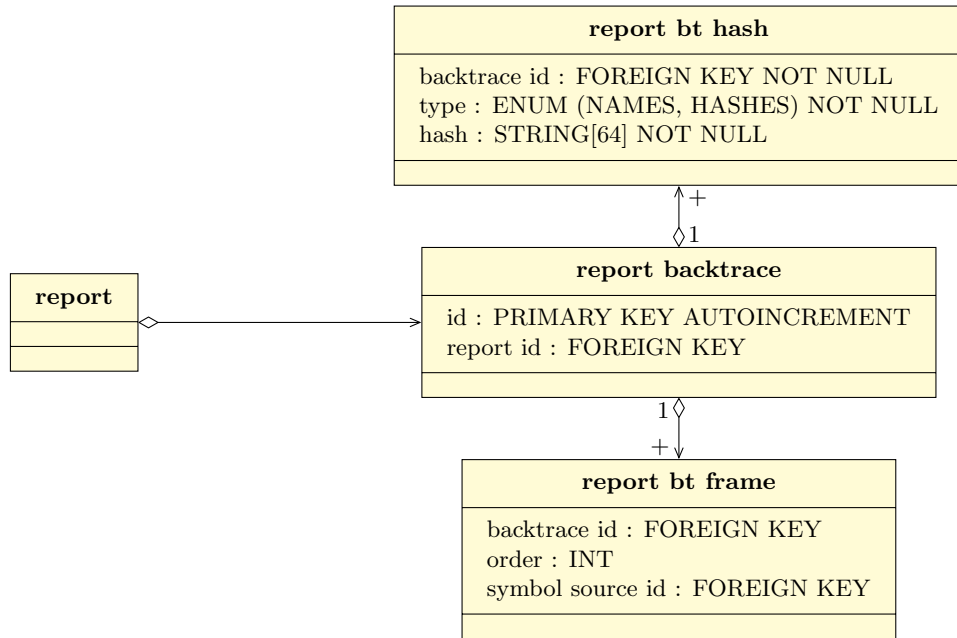
Problems are dynamically created from reports.

5.3.2 Problem Storage – Reports

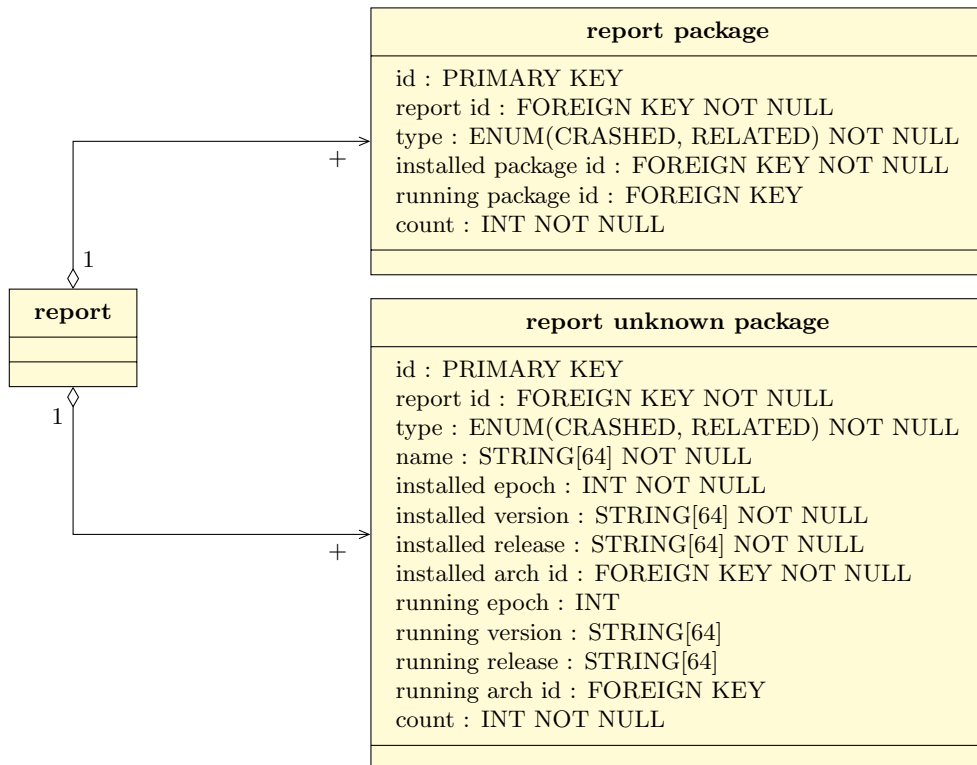
Part of reports is populated from μ reports.



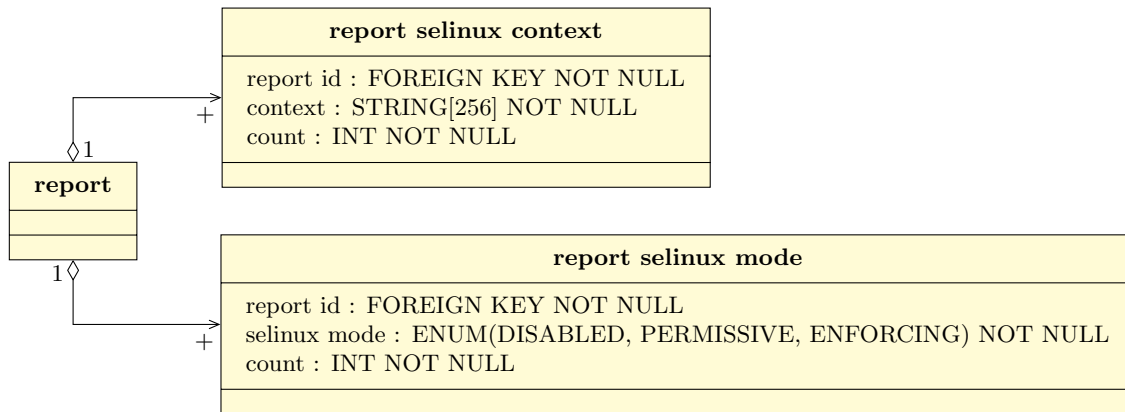
5.3.3 Problem Storage – Report backtraces



5.3.4 Problem Storage – Report packages

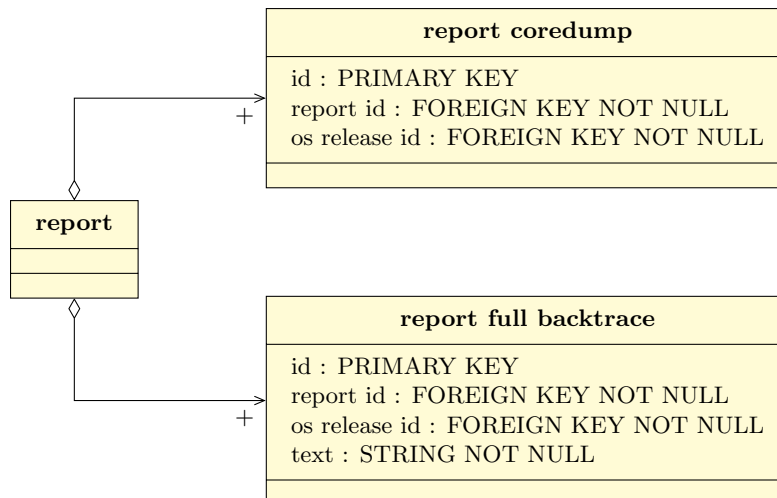


5.3.5 Problem Storage – Report security aspects

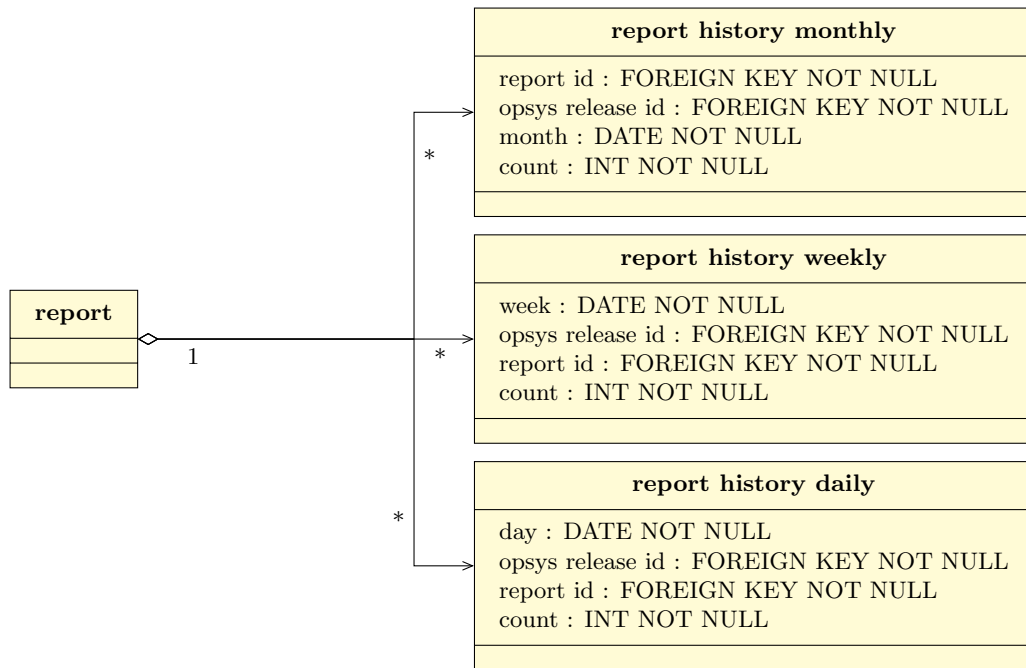


5.3.6 Problem Storage – Report in-depth information

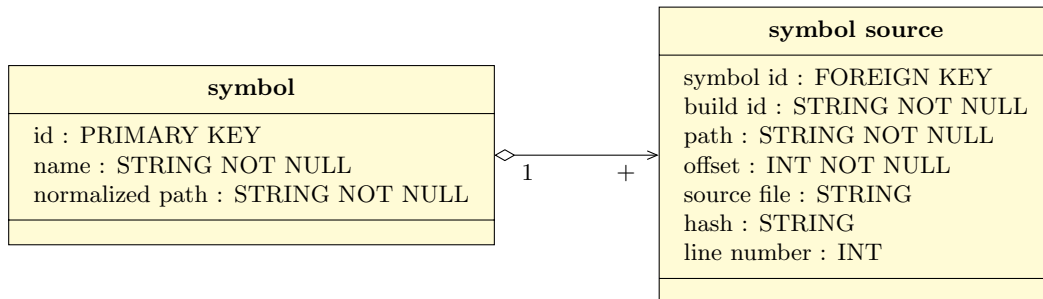
Part of reports is populated from full reports.



5.3.7 Problem Storage – Report history

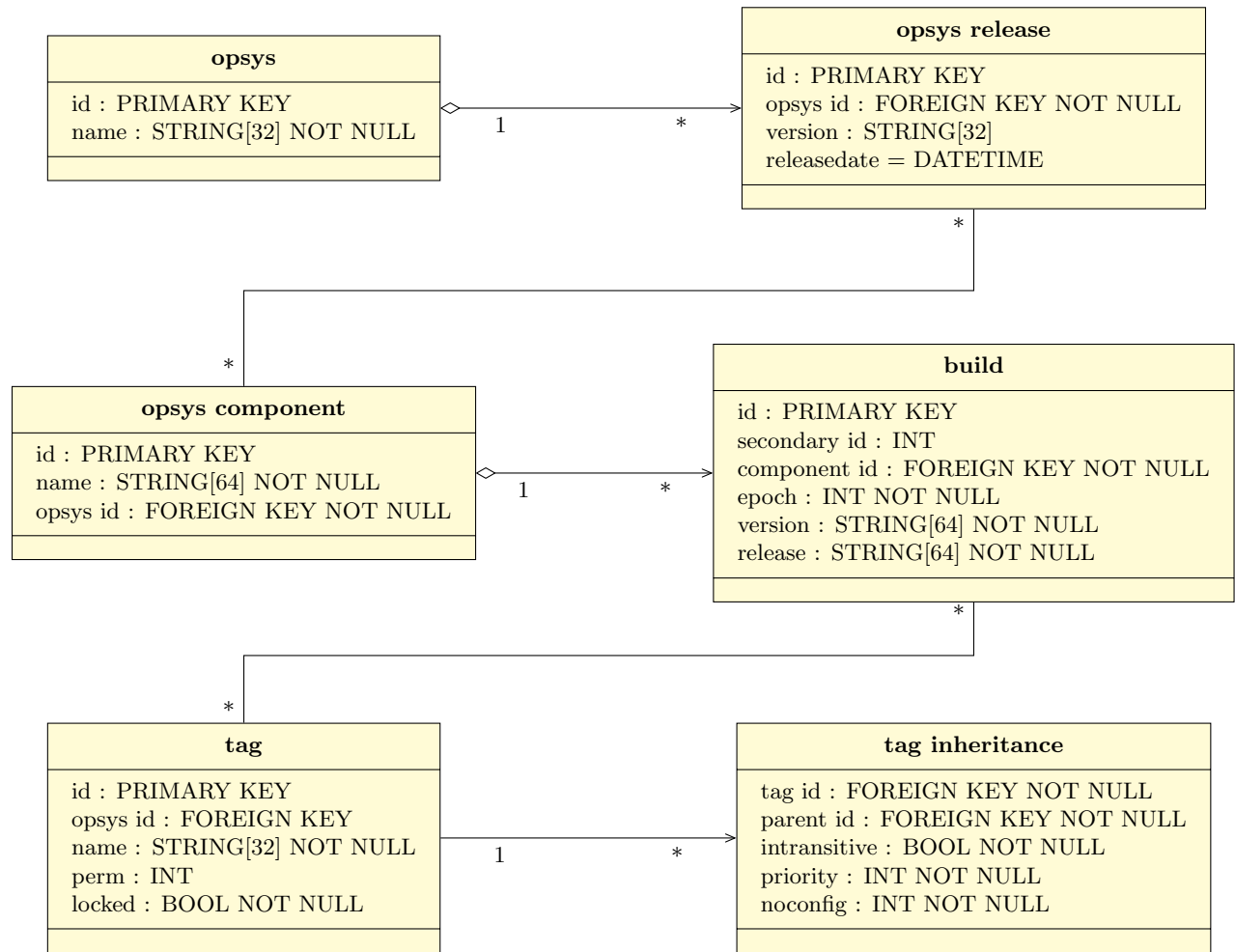


5.3.8 Problem Storage – Symbols

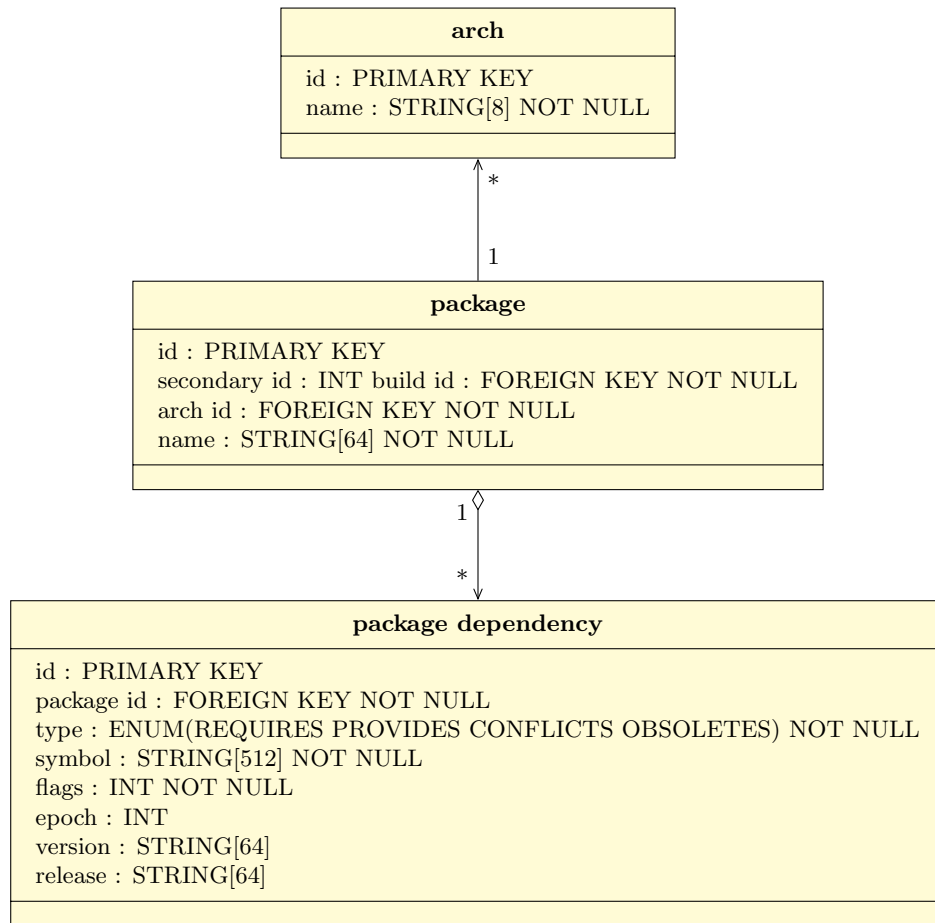


5.3.9 Problem Storage – Clusters

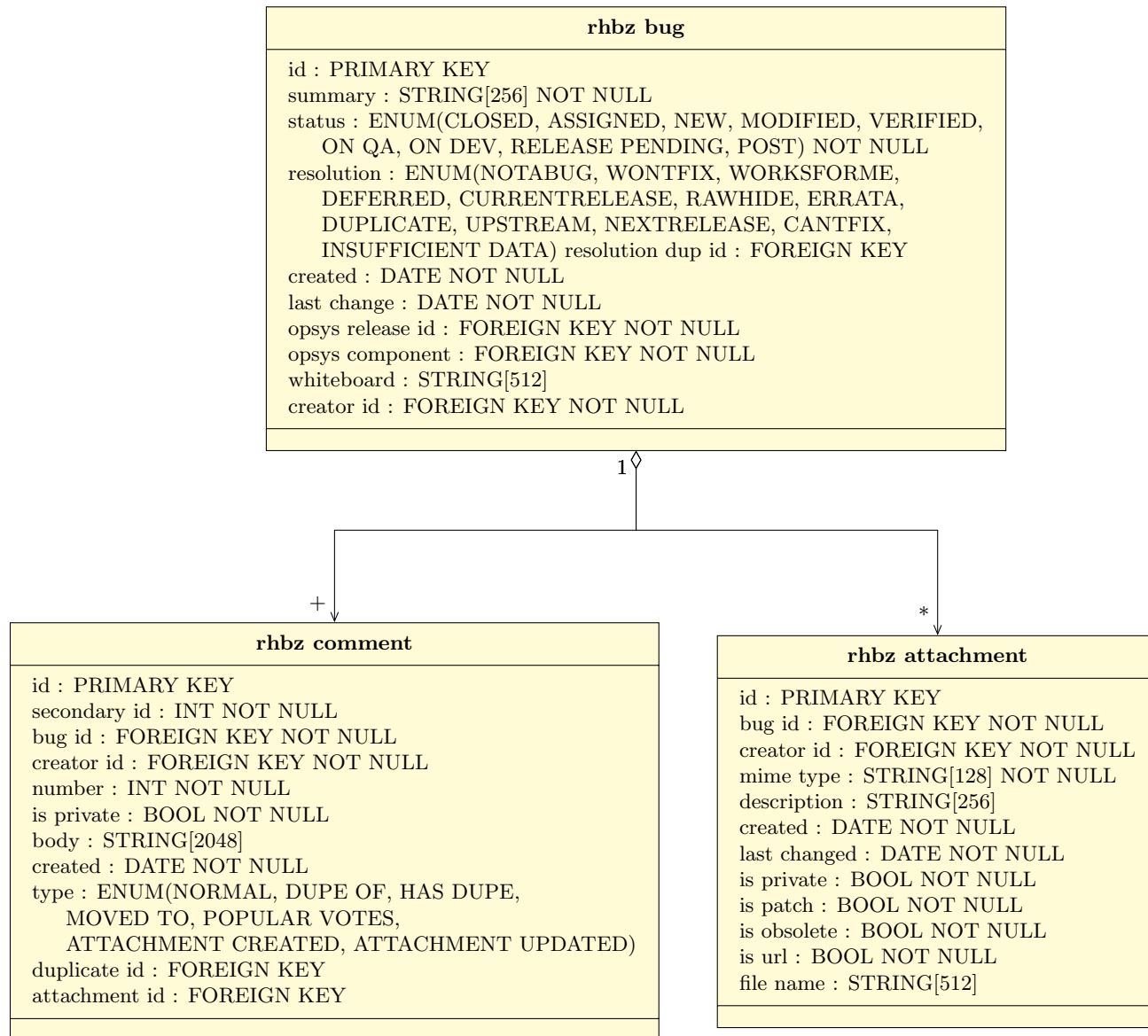
5.3.10 Data Storage – Builds



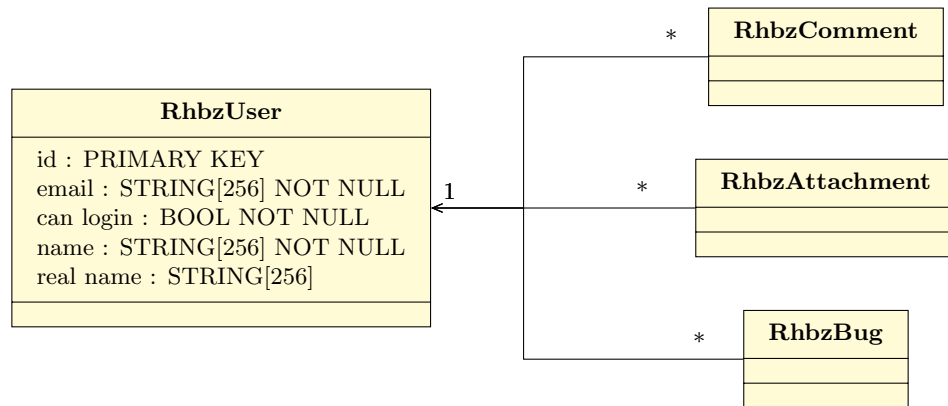
5.3.11 Data Storage – Packages



5.3.12 Data Storage – Red Hat Bugzilla



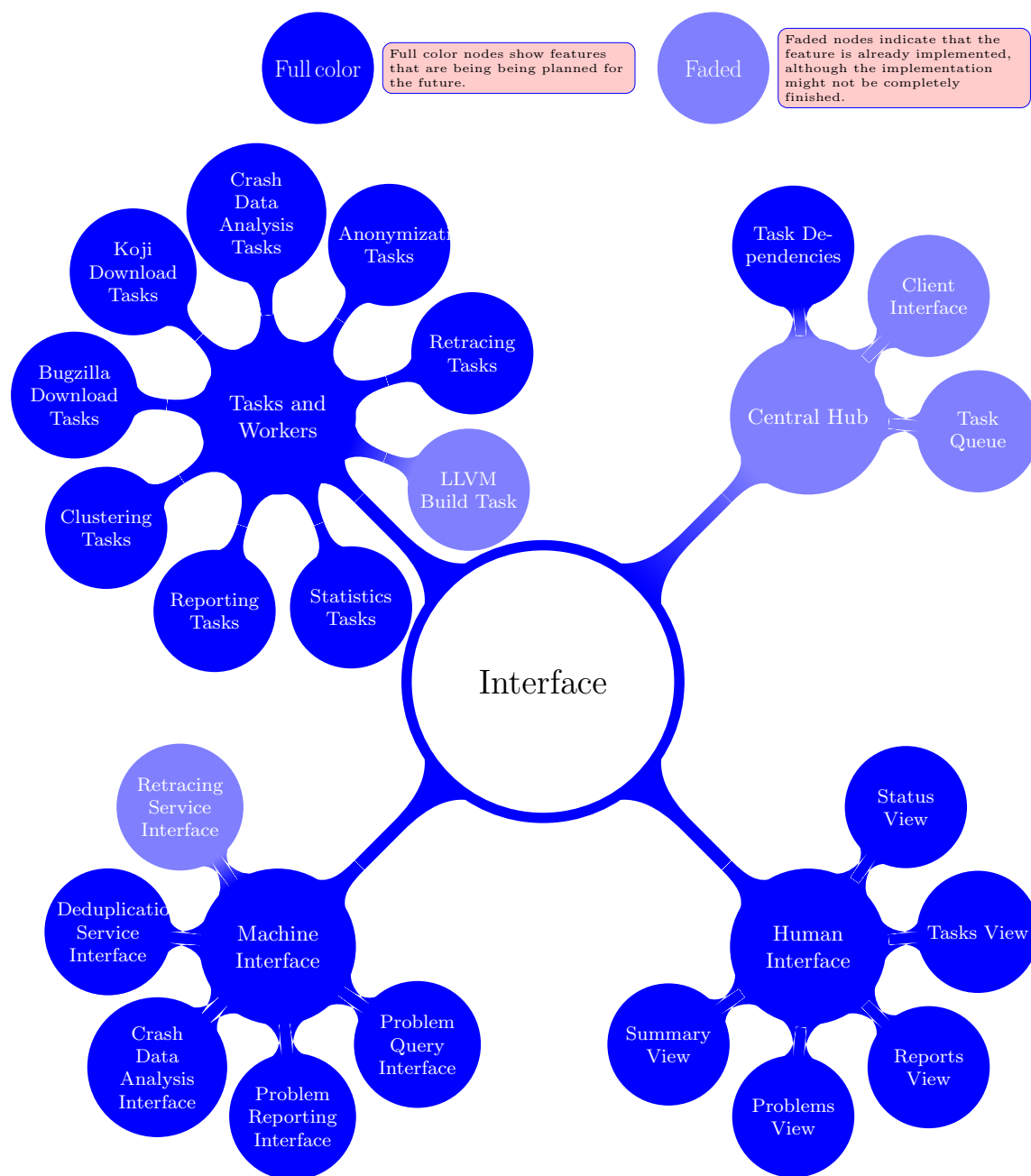
5.3.13 Data Storage – Red Hat Bugzilla Users



5.3.14 Data Storage – LLVM Bitcode

5.3.15 Sanity Checker – Debuginfo checker

5.4 ABRT Server Interface Overview



5.4.1 Summary Page

[login](#)

Fedora Problem Tracker

search for problem

Summary

Problems

Reports

Tasks

Status

Operating System:

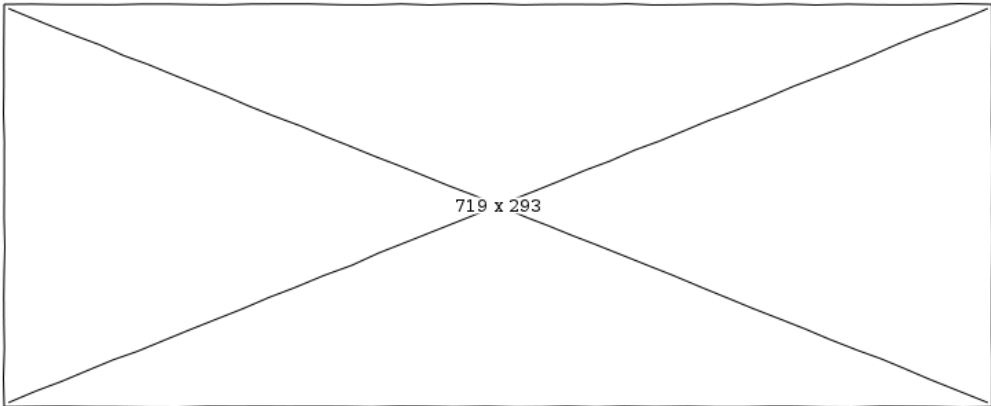
Fedora 16 ▼

All Components ▼

days

weeks

months



New and noteworthy

2012-03-18

GNOME Desktop become unstable in Fedora Rawhide since the last week. The average number of crashes in 5 weeks before that was 10 crashes/week. In the last week it is 86 crashes/week.

Emacs become stable 3 days ago. The average number of crashes in 5 days before that was 12 crashes/day. In last 3 days it is 1.2 crashes/day.

Operating System The available options include Fedora releases, pre-release branched Fedora, Fedora Rawhide, All. The list of releases should be obtained from Fedora Package Database (this is handled by Storage Synchronization).

Components The available options include All Components, the list of all components for the selected Operating System, and selected `comps.xml` groups (such as GNOME Desktop, KDE Desktop, Xfce, Web Server, Electronic Lab, Engineering and Scientific, Font design and packaging, System Tools, Sound and Video, Office/Productivity)

Graph Displays the number of problems (individual events) in time. The underlying data are updated once a day.

Days The graph shows problems in the last 14 days.

Weeks The graph shows problems in the last 12 weeks (3 months).

Months The graph shows problems in the last 12 months.

New and noteworthy Shows automatically discovered interesting trends that can be detected from data. It tells visitor which combinations of Operating System and Component might be worth looking.

5.4.2 Problems Overview Page

[login](#)

Fedora Problem Tracker

search for problem

SummaryProblemsReportsTasksStatus

Hot ProblemsLong-term Problems

Operating System: Fedora 16 ▼ All Components ▼ 7 days 14 days 4 weeks

Rank	Signature	Count	First Appearance	Reports
1	Segmentation fault in gdk_check_xpending()	1253	2012-01-03	gnome#800653
2	Segmentation fault in memmove()	1024	2012-02-05	rhbz#725365
3	Kernel oops - dmar bios	561	2011-03-03	kernel#533214

5.4.3 Problems Item Summary Page

[login](#)

Fedora Problem Tracker

search for problem

Summary

Problems

Reports

Tasks

Status

Segmentation fault in gdk_check_xpending()

Summary

Backtraces

Clusters

History

Environments

Reports

Security

Operating System	Number of Events
Fedora 16	212 (58%)
Fedora 17	113 (30%)
Fedora Rawhide	48 (12%)

Build	Number of Events
emacs-23.1-18.fc15	48 (48%)
emacs-23.2-1.fc16	22 (23%)
emacs-23.2-2.fc16	5 (18%)

Binary	Number of Events
/usr/bin/emacs-23.2	48 (48%)
/usr/bin/emacs-23.1	22 (23%)
/usr/bin/emacs-nox	5 (18%)

Architecture	Number of Events
x86_64	348 (100%)
i686	0 (0%)

Binary objects present in all problem reports:

Binary Object	Component	From	To
/usr/lib64/libgtk-x11-2.0.so.0	gtk	2.2-3.fc15	2.4-4.fc16
/usr/lib64/libatk-1.0.so.0	atk	2.2-3.fc15	2.4-4.fc16
/lib64/libglib-2.0.so.0	glib	2.0-15.fc15	2.0-15.fc16
/lib64/libnss_files.so.2	nss	2.0-15.fc15	2.0-15.fc16

26

5.4.4 Reports Overview Page

[login](#)

Fedora Problem Tracker

search for problem

SummaryProblemsReportsTasksStatus

OverviewList

absolute

relative

Operating System: Fedora 16 ▼

All Components ▼

days

weeks

months

719 x 293

5.4.5 Reports List Page

[login](#)

Fedora Problem Tracker

SummaryProblemsReportsTasksStatus

OverviewList

Operating System: All Components: Destination: Status:

Rank	Report	Status	Last Change	Created
1	rhubz#800653 - Double free in compile()	FIXED	2012-01-03	2012-01-01
2	rhubz#725365 - Buffer overflow in main()	FIXED	2012-02-05	2011-12-06
3	rhubz#565487 - Assert failure in cli.cpp:35	FIXED	2011-03-03	2011-08-23

5.4.6 Server Tasks Page

[login](#)

Fedora Problem Tracker

SummaryProblemsReportsTasksStatus

State: Method:

Id	Name	State	Finished	Arch
65465	LLVM Build of emacs-23.1-16.fc17	FAILED	2012-01-03	x86_64
45554	LLVM Build of btparser-0.17.fc18	FAILED	2012-02-05	x86_64
22455	LLVM Build of abrt-2.2.fc18	FAILED	2011-03-03	x86_64

5.4.7 Server Status Page

[login](#)

Fedora Problem Tracker

search for problem

SummaryProblemsReportsTasksStatus

OverviewBuilds and RPMsLLVM BitcodeDebuginfo CheckIntegration TestsWorkers

Last synchronization with Koji: 2012-03-16 15:36

Last synchronization with Red Hat Bugzilla: 2012-03-16 18:02

Last check of debuginfo issues: 2012-01-10 13:52

Last run of integration tests: 2012-03-17 10:00 [Fedora Rawhide test failed!](#)

Cache: 1.2 TB (31%) used; 3.1 TB available

5.4.8 Problem Reporting Interface

Server accepts microreports. Microreport is a JSON-formatted data structure described below:

Name	Format	Mandatory	Notes
type	python , userspace , or kerneloops	yes	Format depends on the type . Number of seconds from program start to the problem.
reason	Unicode string, max. 128 characters.	yes	
uptime	Unsigned integer.	yes	
executable	Full path, max. 512 characters.	yes	
installed_package	Dictionary; see the package table below.	yes	
running_package	Dictionary; see the package table below.	no	
related_packages	List of dictionaries; see the related_package description below.	yes	
os	Dictionary; see the os table below.	yes	
architecture	x86_64 or i386	yes	
reporter	Dictionary; see the reporter table below.	yes	
core_backtrace		yes	Program that created the report.
os_state	Dictionary; see the corresponding table below.	yes	
user_type	root , nologin , local , or remote	no	
selinux	Dictionary; see the corresponding table below.	no	
proc_status	ASCII string, max. 2 kB.		The contents of /proc/pid/status .

The **os** structure:

Name	Format	Mandatory	Notes
name	ASCII string.	yes	Numeric. No codenames.
version	ASCII string	yes	

The **os_state** structure:

Name	Format	Mandatory	Notes
suspend	yes or no	no	Problem happened during suspend, hibernate, or resume.
boot	yes or no	no	Problem happened during boot process.
login	yes or no	no	Problem happened during login process.
logout	yes or no	no	Problem happened during logout process.
shutdown	yes or no	no	Problem happened during the shutdown process.

The **reporter** structure:

Name	Format	Mandatory	Notes
name	ASCII string, max. 128 characters.	yes	
version	ASCII string, max. 128 characters.	yes	

The **related_package** structure:

Name	Format	Mandatory	Notes
installed_package	Dictionary; see the package table below.	yes	
running_package	Dictionary; see the package table below.	no	

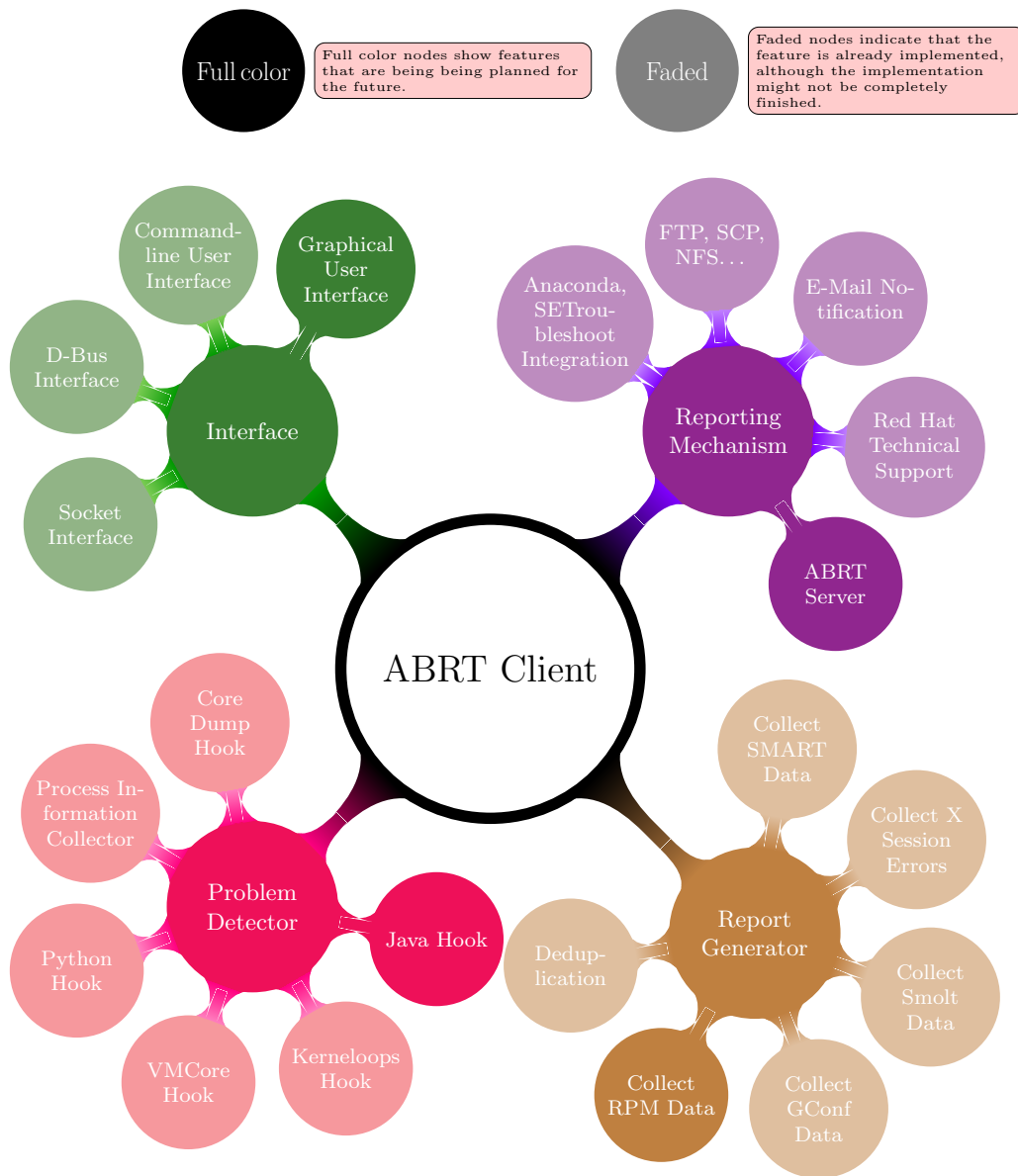
The **package** structure:

Name	Format	Mandatory	Notes
name	ASCII string, max. 128 characters.	yes	
version	ASCII string, max. 128 characters.	yes	
release	ASCII string, max. 128 characters.	yes	
architecture	ASCII string, max. 128 characters.	yes	
epoch	ASCII string, max. 128 characters.	yes	

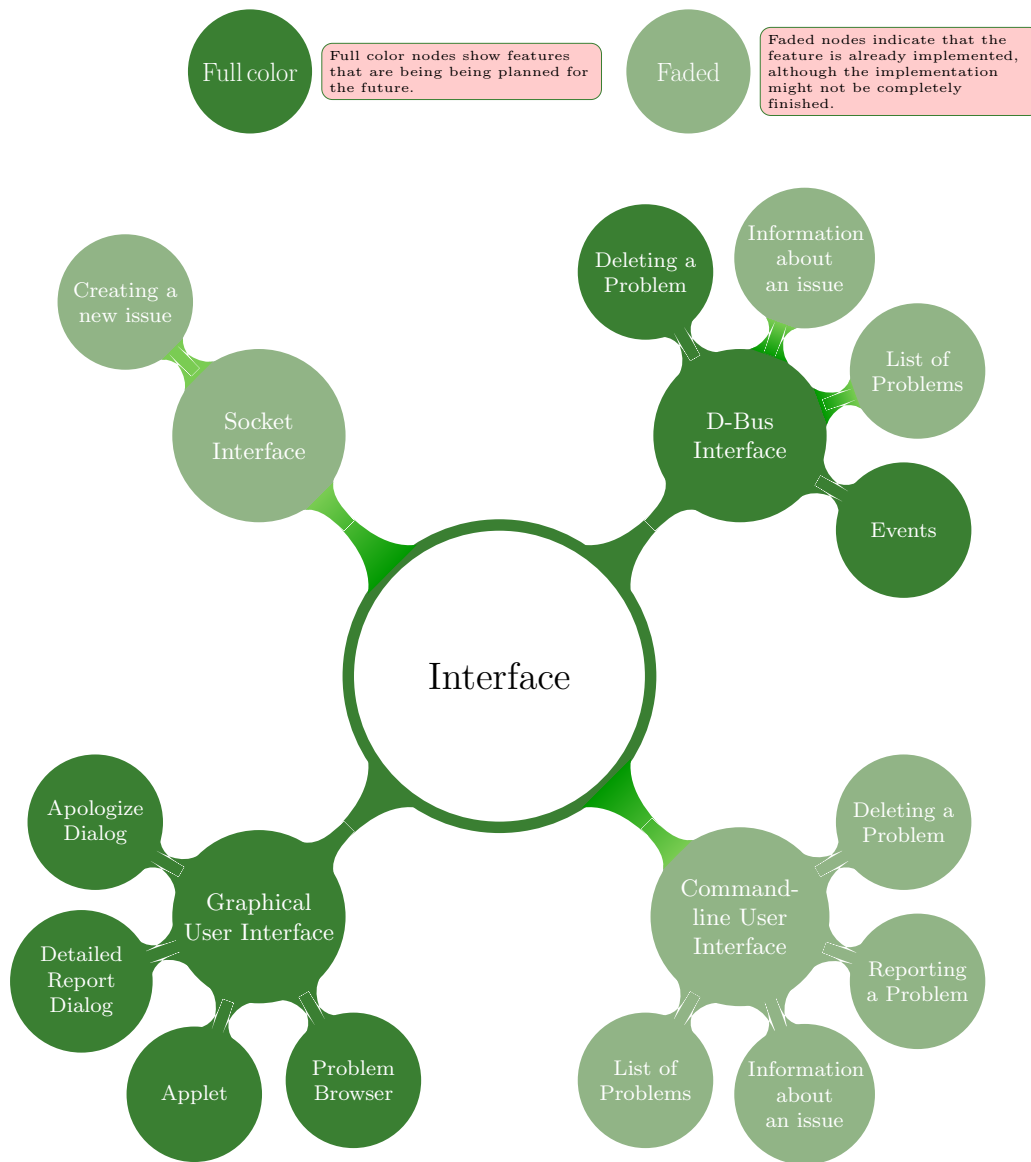
The **selinux** structure:

Name	Format	Mandatory	Notes
mode	enforcing , permissive , or disabled	yes	
context	ASCII string, max. 128 characters	yes, if the mode is either enforcing or permissive	ps -e --context
policy_package	Dictionary; see the package description.	no	

5.5 ABRT Client Overview

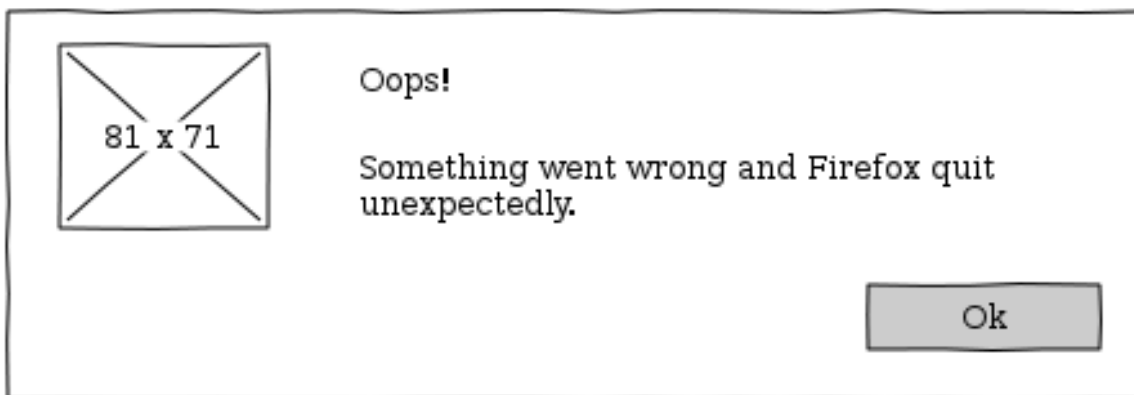
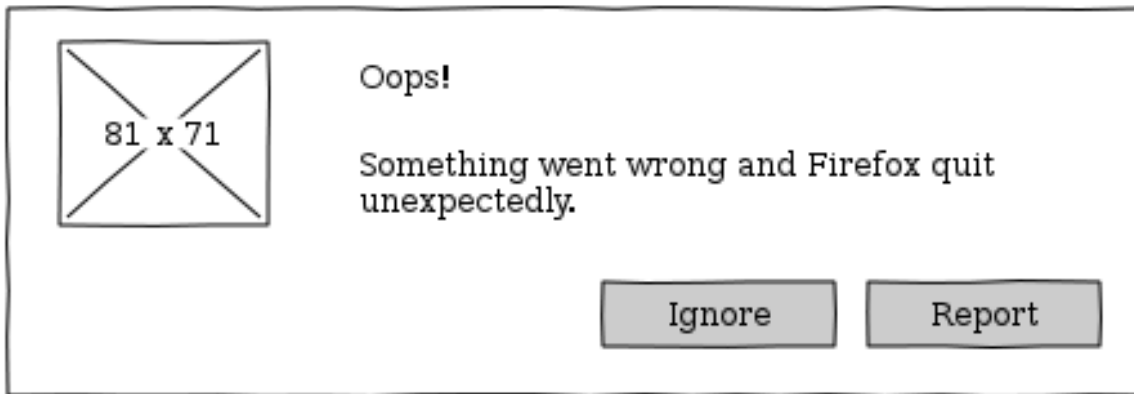


5.6 ABRT Client Interface Overview

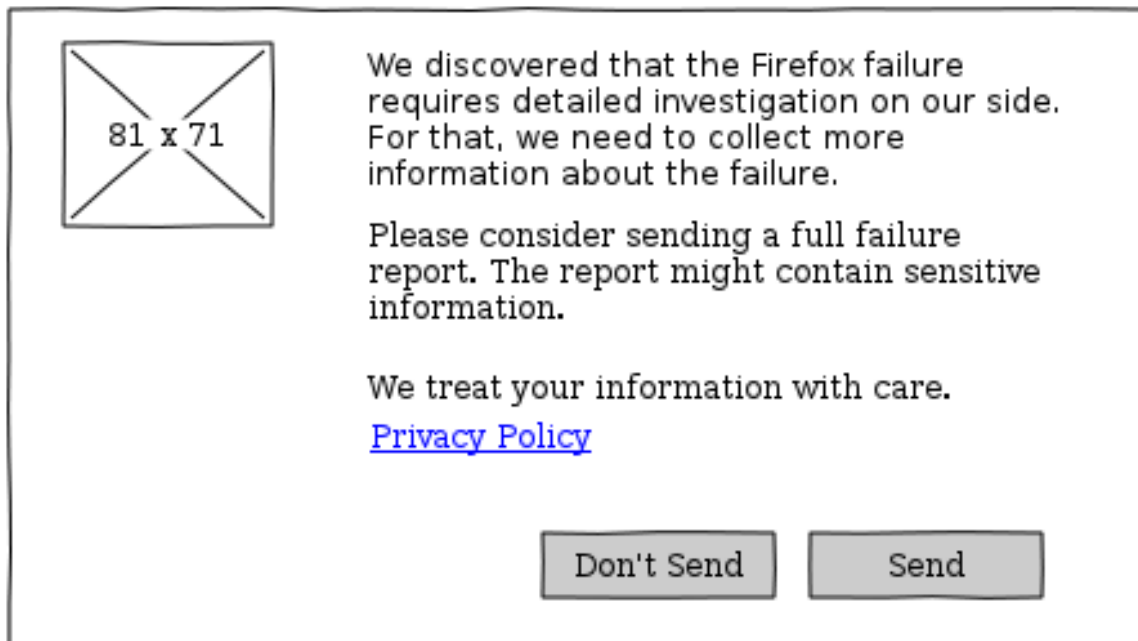


5.6.1 Apology Dialog

The dialogs come from [1]. Please see [1] for more detailed design.



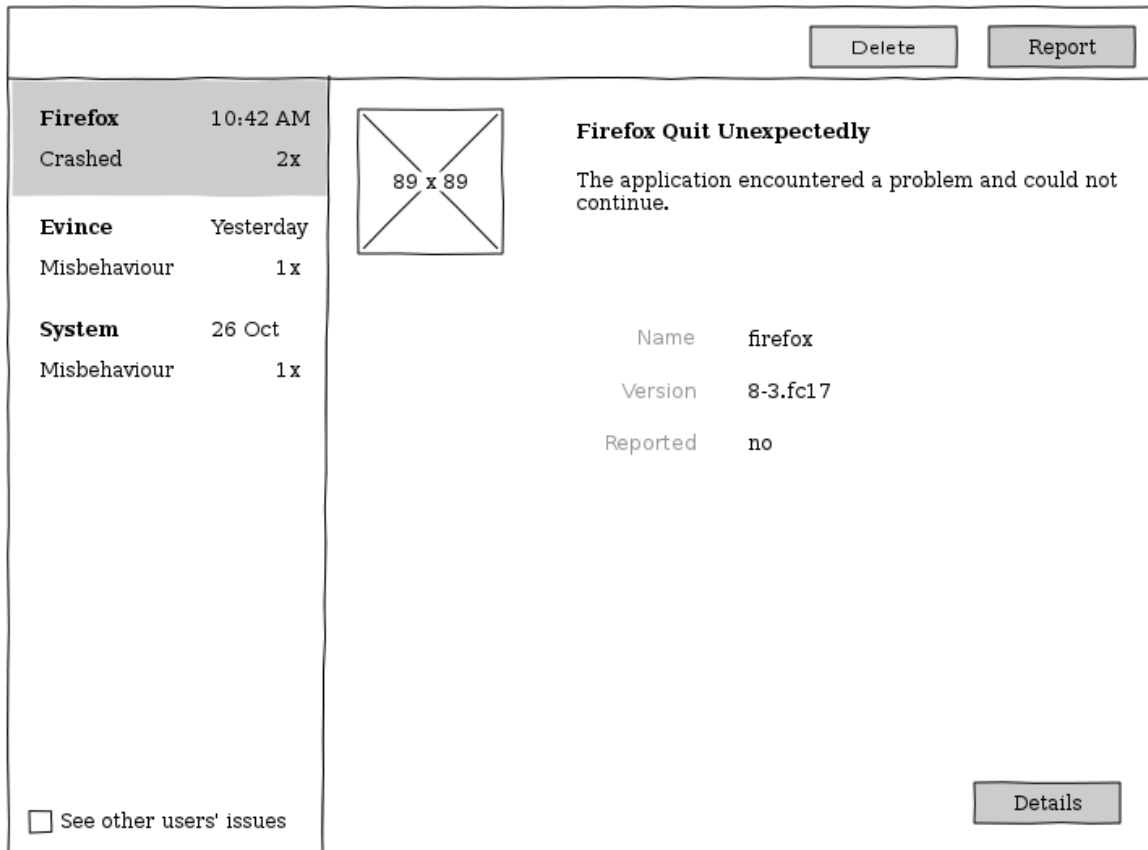
5.6.2 Detailed Report Dialog



5.6.3 Applet

5.6.4 Problem Browser

The Problem Browser dialog comes from [1]. Please see [1] for more detailed design.



6 Project Time Management

6.1 Fedora 17 Phase

Finish date: 2012-05-22.

6.1.1 Sprint 1

Finish date: 2012-04-20 Friday

Duration: 3 weeks

- Server
 - Problem storage
 - * Database schema [mtoman,mlichvar]
 - * Storage of incoming problems [mlichvar]
 - Data storage

- Initial database schema [mtoman]
- Deduplication
 - * Deduplication of incoming problems according to hashes
 - * Deduplication of retraced problems according to symbols
- Retracing
 - * Retracing of microreports
- Machine interface
 - * Receive report
- Migration to SQLAlchemy [mlichvar]
- RHEL6 compatibility [mtoman]
- Client
 - Coredump-level backtraces [mmilata]
 - Microreport sender [npajkovs]
 - User interface [dvlasenk]

6.1.2 Sprint 2

Start date: 2012-04-23 Monday

Finish date: 2012-05-11 Friday

Duration: 3 weeks

- Server
 - Human interface
 - Machine interface
 - Tasks and workers
 - Clustering
 - * Adapt existing source code to match the server
 - * Properly create problems from reports

6.1.3 Sprint 3

Start date: 2012-05-14 Monday

Finish date: 2012-05-22 Tuesday

Duration: 7 days

6.2 Fedora 18 Phase

Finish date: 2012-11-01

6.3 Activity List

1. Implement the problem storage tables in SQLAlchemy. [server]
2. Implement server machine interface for communication with client. [server]
3. Implement saving the received microdump to the database. [server]

4. Set-up and document regular, automatic database backup. [server]
5. Finish the design of minireport and fullreport backup. [server]
6. Implement minireport and fullreport backup. [server]
7. Implement deduplication logic on arrival of a new report. [server]
8. Implement deduplication logic for clustering and merging of existing reports. [server]
9. Implement coredump-level backtraces. [server]
10. Implement retracing of coredump-level backtraces. [server]
11. Implement the basic layout of human interface. [server]
12. Implement graph on the summary page. [server]
13. Implement initial problems overview page. [server]
14. Implement initial problem summary page. [server]
15. Implement both variants of apology dialog. [client]
16. Implement detailed report dialog. [client]
17. Implement ABRT client uploader. [client]
18. Prepare proposal for Anaconda changes. [client]
 - Define Activities – Activity List – Activity attributes – Milestone list
 - Sequence Activities – Project Schedule Network Diagram
 - Estimate Activity Resources – Activity Resource Requirements – Resource Breakdown Structure
 - Estimate Activity Durations – Activity Duration Estimates
 - Develop Schedule – Project Schedule – Schedule baseline – Schedule data

7 Project Quality Management

- Plan Quality – Quality management plan – Quality metrics – Quality checklists – Process improvement plan

8 Project Risk Management

9 Project Management Plan

References

- [1] Jon McCann, *Oops!*. <https://live.gnome.org/Design/Apps/Oops>.
- [2] Jon McCann, *Problem Recovery and Reporting*. <https://live.gnome.org/GnomeOS/Design/Whiteboards/ProblemReporting>.
- [3] Jon McCann, *Problem Reporting Architecture Proposal*. <https://live.gnome.org/GnomeOS/Design/Whiteboards/ProblemReporting/Proposal>.
- [4] *Debugging in the (Very) Large: Ten Years of Implementation and Experience*. <http://msdn.microsoft.com/en-us/windows/hardware/gg487440>.