SUSE bug reporter

Bug reporting tool for openSUSE

Abstract

SUSE bug reporter is the tool that will help users submit bug reports to the developers, allowing even people with limited technical knowledge to fill out a relevant and useful bug report in a quick and simple way. This will reduce the delays, confusions and even frustrations between users and developers.

A big advantage of the tool's integration with openSUSE will be the fact that it will use existing package metadata. The *SUSE bug reporter* will offer the user the choice of updating the presumed buggy package in case an update is available but not installed. Also, it will query information from the openSUSE Build Service about the respective pacakge's maintainer so the report will be assigned to the right person. The command-line name will be simple and intuitive: `bugreport`.

All the features mentioned above and several others (described below) add up to a great software product that will improve the openSUSE experience.

Detailed Description

Background

openFATE link

The tool will work in both command-line and GUI-mode.

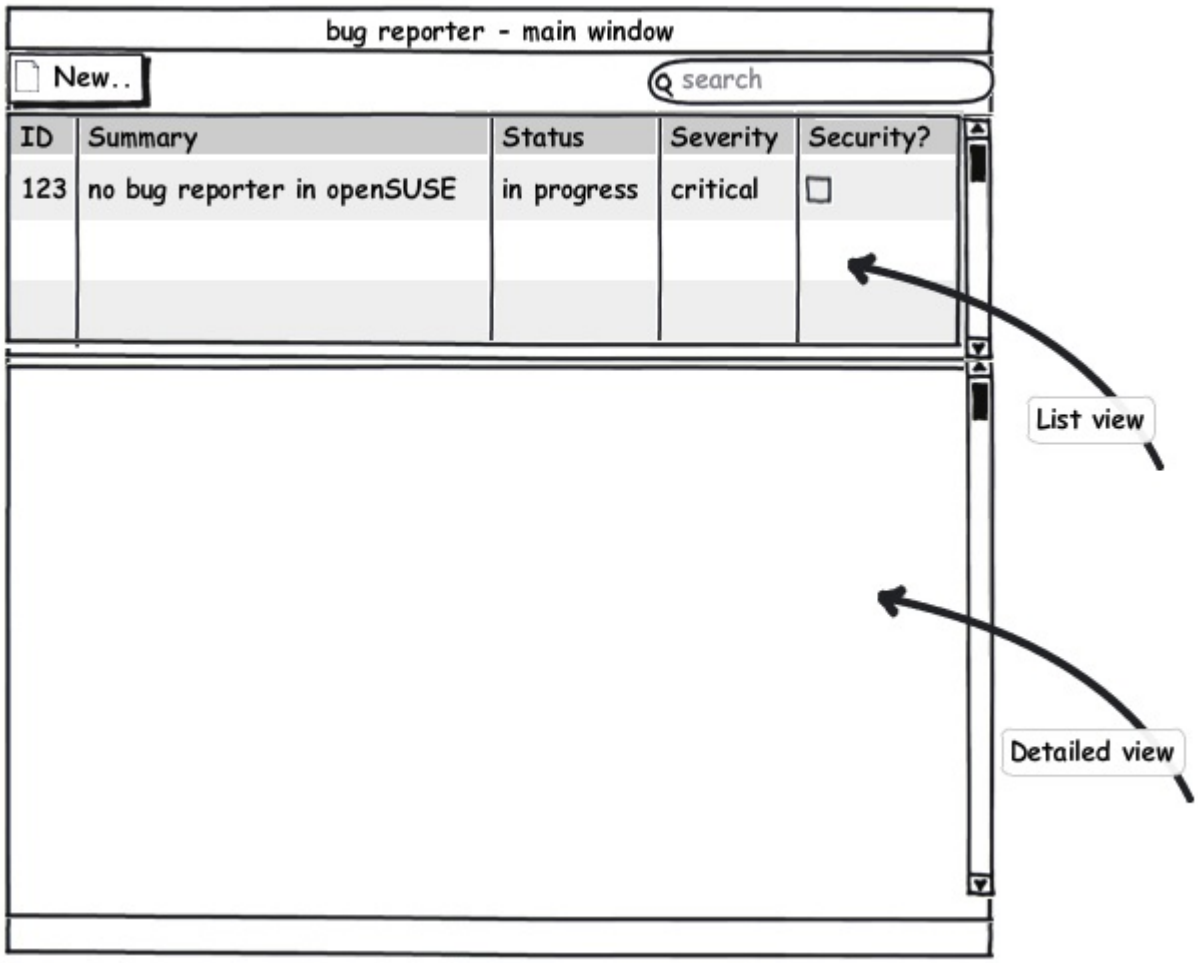
Main Features

- **check if the user has an account**
  - at installation (or the first time the tool is used), prompts the user for the Bugzilla login details and redirects the user to the registration page, if necessary.
- **aid the user in finding if a similar bug report has already been submitted**
  - shows a list of submitted bug reports for the same package
  - option to use the bug description/summary as a cross-check criteria for filtering
- **show info about a package**
  - similar to the zypper info output
- **list the user's already submitted reports and allow updating them**
- **aid the user in finding the package responsible for the noticed issue**
  - check memory & CPU usage
  - let the user choose the window (if applicable) with the cursor
  - if the package's name cannot be determined, the tool will submit a more generic bug report containing as much relevant information as possible; this bug will be sent to the screening team
- **search through the existing reports by various criteria, i.e. package name**
- **when submitting a bug report:**

- warn the user if his current package version is outdated
- find the corresponding maintainer, reducing the number of bugs to be filtered by the Novell screening team
- prompt the reporting user if the bug leads to a security issue; used to raise the severity/importance of the report
- allow the user to assign predefined tags to the bug report

GUI Features

- **Main window mockup:**



- **filtered list view**
  - shows the bug reports that match the currently selected criteria
  - colred keywords depending on severity, status etc.
  - search by keyword, by package name, by tag etc
- **detail view**
  - shows the entire bug report, formatted for easy reading
- **add new / update bug prompt**
  - GUI mode of submitting a bug report
  - intuitive interface for adding tags
- **MVC architecture to facilitate translations**
  - I can provide the first one, in Romanian

Use Cases

- **#1: the user notices unexpected and unidentified general sluggishness**
  - the bug reporting tool checks the %CPU, %MEM and TIME of all running processes and finds a daemon that is using most of the available resources, i.e. 90% CPU and/or 50% MEM
  - the user confirms that this is unexpected and selects the option to file a bug report
  - based on the earlier finding, the tool already "knows" what package to file the bug report against
- **#2: the user wants to report a bug found while using a GUI application but does not know the name of the package**
  - he uses the bug reporting tool with the "--select-window" option
  - now he can click to identify the (considered) "buggy" application and the tool finds the package name
  - the user is prompted to file a bug report against the respective package
- **#3: the screening team needs help in filtering duplicates and finding the maintainer of a package**
  - they can use the bug reporting tool as an end-user would.
  - even more, by doing so, they would help improving the quality of the bug reports database by assigning tags and such

Benefits

- **End-users**
  - the process of reporting a bug is simplified
- **Developers**
  - the bug reports get assigned to them faster, overcoming the bottleneck created by the manual filtering of the reports by the screening team
- **Members of the screening team**
  - there will be less bug reports to filter
- **The Community**
  - all the aforementioned facts lead to faster bug fixing and a better overall level of software quality

Caveats

- **Duplicate check**
  - while the tool does show already submitted similar bug reports, it's up to the user to decide if the one he wants to file would be a duplicate or not
- **Identifing the flawed application**
  - while this might be trivial in certain cases (i.e. "app" crashes), it is not a general fact

- the tool will aid the user in identifying the troublesome software (as described in Background -> Main Features), but the amount of help provided is limited, without taking A.I. into consideration
- because of the limiting fact stated above, the amount of identified bugs will also depend on the user's knowledge and skill level

Technical Details

- **Data to collect**

  1. **package name & package version**
     - the user determines them aided by the tool (internally using tools like `top`, `ps`, `xprop WM_CLASS`, `osc` etc.)
     - i.e. `rpm -q --qf '%{disturl}\n' $packagename` to find the name of the source package
     - also, `xprop WM_CLASS` will support the functionality of the user clicking the application's window he wants to report a bug against
  2. **system (hardware, drivers) info**
     - `lshw`, `lspci` etc
  3. **OS & kernel version**
     - `uname -a`
  4. **steps to reproduce**
     - user input
  5. **expected result**
     - user input
  6. **actual result**
     - user input
  7. **summary (helps also for searching)**
     - user input
  8. **is it a security concerning issue?**
     - user input

- **Interaction with Novell's Bugzilla**
  - the `XMLRPC` interface provided by Bugzilla via something implemented already, maybe (PyZilla or something similar)
  - the `osc` libraries (i.e. to find info about a project's maintainer)

- **GUI**
  - I will use the `PyQt` framework, since openSUSE's default desktop environment is KDE

Timeline

This provides a *rough* guideline of how the project will be done. As you will see, I plan to start working right after the accepted proposals list is published. Because of that, according to my timeline, I will have finished the project earlier. I don't know if this will be

the case, but in all events, I will have time to cover any unexpected issue that may come up. My university exams session will take place between the 21st of May and the 10th of June, but it won't really affect my work on the project because of two facts: 1. the first tasks aren't very difficult and 2. I have studied all year long for those exams.

- 27 April - 4 May
  - familiarize myself with the rpm metadata and osc
  - exercise implementing small operations in python
- 5 May - 12 May
  - familiarize myself with the Bugzilla XMLRPC interface
  - excercise using PyZilla, see if it's enough for the project's needs
  - implement the account check feature.
- 13 May - 20 May
  - implement the modules that gather system info from logs, lshw etc.
  - implement the module that helps the user determine the 'flawed' app.
  - put these modules together into a basic app that gathers all the data, adds user input and generates a bug report
- 21 May - 28 May
  - work on some kind of a predefined template with extra data to insert in the bug report, i.e. in the "Bugzilla Whiteboard"
  - work on some kind of a tagging system and implement it in the tool
- 28 May - 4 June
  - find the maintainer and search through existing bug reports module
  - show info about a package module
  - check updates for a package module
- 5 June - 19 June
  - work on the search & report possible duplicates feature
- 20 June - 24 June
  - write the documentation for the CLI app
- 25 June - 9 July
  - start working on the GUI version
  - do a basic skeleton for the app
  - populating the list view and filtering it
  - generating the detail view
  - "add new / update bug" prompt
- 10 July - 14 July
  - Work on the search feature.
- 15 July - 20 July
  - write the documentation for the GUI version
  - do a Romanian translation

Why Me

My name is Mihnea Dobrescu-Balaur, I'm from Bucharest, Romania and I study at the Politehnica Univeristy of Bucharest, the Faculty of Automatic Control and Computer

Science. I'm a first year student with lots of time that I am willing to spend studying and working on this project. This is mainly because, ever since I wrote my first "Hello, World!" program (back in high school), I wanted to do some "real world" programming, something that other people would use.

I didn't have any real opportunity to do so, until I found out about the CDL course, taking place in our university. I applied, was accepted and I currently work on a Python&Django-based project. It is a browser game that focuses on testing the students' knowledge about operating systems. It is a great occasion before the GSoC for me, because I got to learn about working in an open-source community, using version-control systems like git, asking for help on mailing lists and other stuff like that. The course will end before GSoC starts, so there's no time conflict between these two.

Having mentioned "mailing lists", another reason why I would be adequate to work on this project is that I have already discussed implementation details with the people on the opensuse-project list, and the people there seemed really interested in my ideas and also in helping me out. I have to say, the openSUSE community is great! Also, the openSUSE distro is, to me, the most polished distro around. I don't really know how to describe this, it is just my subjective opinion. It feels that people have really put in hard work and I want to add to that.

In a nutshell, this project is, to me, the perfect opportunity to learn new things, to continue my development as a programmer and to do what I've always wanted: contribute with a piece of software that would actually help the community.

Contact Information

- e-mail: mihneadb@gmail.com
- irc nick: mihneadb
- IM: google talk (gmail address)
