

# Profile of openSUSE 11.0 updates

Daniel Massaguer  
dani@a65.cat

Jordi Massaguer  
jordi@a65.cat

–White Paper (UFO-WP-09-01)–

**Abstract**—This paper shows some statistics on how the updates of openSUSE 11.0 happened. This is a first step towards building a prediction model of when updates in a linux distribution will happen.

## I. INTRODUCTION

The cost of maintaining linux-based software appliances and distributions depends on the updates of their different components. However, when these updates happen is not pre-set and depends on a series of observable and non-observable factors. Some of these factors include the experience and dedication of the programmers, the size of the programming teams, the time of the last update, and the maturity of the project.

No formal study exists on when updates happen. As part of our preliminary study before building a prediction model for when the next updates for a set of components will be, we ran some basic statistics on the updates regarding openSUSE 11.0. openSUSE 11.0 has been out for a year, already—it was released by June 19th, 2008. In this paper, we report a series of statistics on this 1-year old distribution and discuss the next steps. Note that this is only a description of past updates. It is, by no means, an update model.

## II. STUDY SETUP

We developed a Java + SAX2 application that parses an XML file with information regarding the updates of the distribution and outputs a spreadsheet. The XML file (*updateinfo.xml.gz*) was downloaded from openSUSE's official update repository at <http://download.opensuse.org/update/11.0/repdata/>. The output spreadsheet contains information regarding each new rpm file released as part of a new update. This information includes the name, time, type of update (e.g., security), architecture (e.g., i586, ppc), as well as how long it has been since the component implemented by the rpm was last updated. We used OpenOffice.org Calc to analyze the output spreadsheet. The results are summarized in the next section.

The source code used is available from SourceForge.net at <http://sourceforge.net/projects/updateforecast/> under the GPLv3 license.

## Number of updates per module openSUSE 11.0 - i586 - update rep

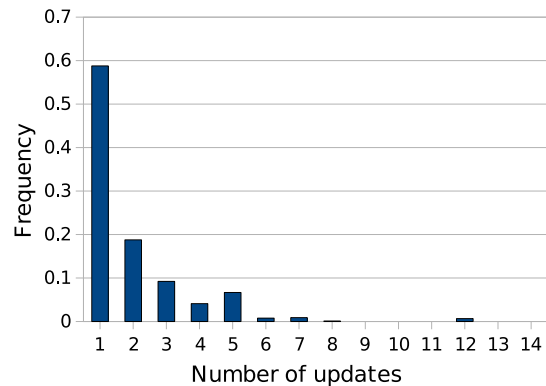


Fig. 1. Distribution of the number of updates per module

## III. RESULTS

We focused on studying when the updates happen and how much time happens between updates. Knowing when the updates happen allows managers to schedule the amount of resources to system managing and software testing—the more updates the more modules (which depend on the updated software) might fail and might need either reconfiguration or recoding.

Knowing the time between updates allows system managers and developers to schedule their time—if the new update for module A is expected soon, one should be ready to check A and all the modules that depend on A.

### A. Number of updates vs time

A year after its official release on Thu June 19th, 2008, openSUSE 11.0-i586 has had 1742 updates regarding 900 modules. The earliest update was 13 days before the official release. The average number of updates a year is 1653.5 with 4.53 updates per day. The number of updates per module distribution is depicted in Figure 1. The average updates per module is 1.82 *updates/year*. Note that 59% of the modules had only 1 update.

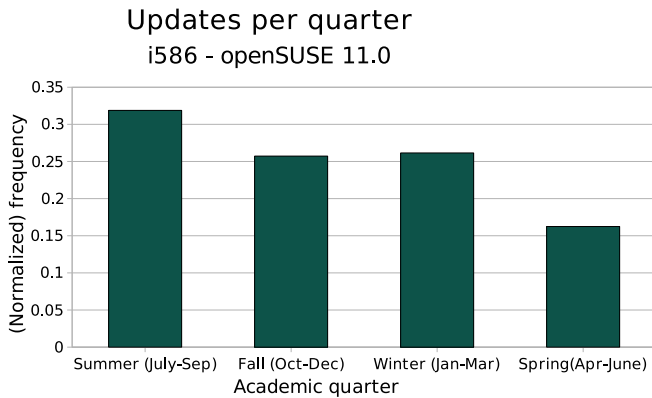


Fig. 2. Distribution of updates along the different seasons

Number of updates for 1/2 month

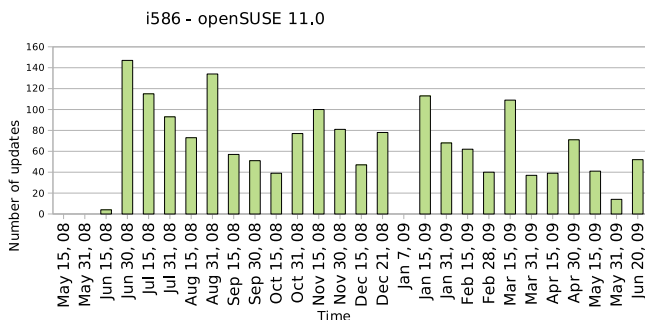


Fig. 3. Distribution of updates for every half a month

The number of updates by season is depicted in Figure 2<sup>12</sup>. Summer was the busiest season (32% of all updates concentrated on that season). Note, however, that one cannot discern whether this is because more updates happen in Summer or because Summer was the closest season to the official release.

Figure 3 shows the number of updates w.r.t. time where the bins correspond to half of the month, with the exception of the time around the “Winter break”, where bins are smaller. Note how there were 0 updates from December 22<sup>nd</sup> to January 7<sup>th</sup>. No updates were released during the “Winter break”.

The busiest period, on the other hand, is during the second half of June, followed by the second half of August and the first half of July (July and August were the busiest months (Figure 4)). Note as well that, whereas there were no updates for the first quarter of January, there were a high number of updates on the second quarter—almost as many as there were in the first 15 days of July but with half as many days—making January (along with November) the 3<sup>rd</sup> busiest month (Figure 4).

Also Figure 3 shows some periodicity, with a peek on the number of updates every 2 months approx, with an overall downward trend.

<sup>1</sup>For delimiting the seasons, we followed the academic quarters’ convention where Summer={Jul, Aug, Sep}; Fall={Oct, Nov, Dec}; Winter={Jan, Feb, Mar}; and Spring={Apr, May, Jun}

<sup>2</sup>Note that we approximated the last 10 days of June 2009 based on the number of updates for the first 20 days

Frequency of updates per month

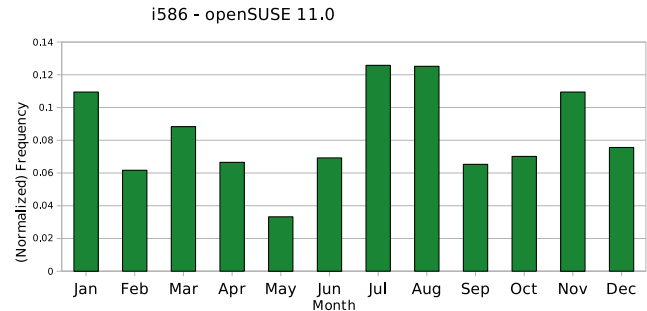


Fig. 4. Average distribution of updates per month

Time between updates

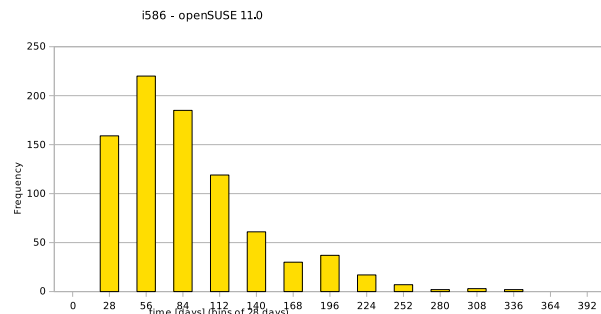


Fig. 5. Time between updates

## B. Time between updates

We modeled the updates of a given a module  $i$  as a random process where updates arrive according to a random variable  $x_i$ . As a starting point, we assumed all  $x_i$  to behave similarly. Figure 5 shows a histogram of the time between updates for all those modules that had more than one update. The average time between updates was 73.5 days with 75% of the updates in  $[21.4, 138.75]$  (in other words, between 21.4 days and 4.6 months). The standard deviation was 53.19 days.

If we were to assume that all events are independent, continuously, and happen at a constant rate (i.e., a Poisson process), one would expect the time between updates to follow an exponential distribution with an estimated rate of  $\lambda = 1/73.5$ . However, the shape of Figure 5 suggests that the hypothesis of the time between updates following an exponential distribution should be rejected. In fact, we ran the Anderson-Darling, Kolmogorov-Smirnov, and Chi-squared tests and they all supported to reject such a hypothesis.<sup>3</sup>

## IV. CONCLUSIONS AND FUTURE WORK

This paper presents the results of a statistical analyses of both the time of and the time between openSUSE 11.0 updates. Regarding the time of the updates, we observed that there is a downward trend on the number of updates as the distribution becomes more mature, with Summer being the busiest season. We are planning on running a similar analysis with respect to openSUSE 11.1, which was released last December.

<sup>3</sup>We ran the previous tests for other well-known distributions (Normal, Lognormal, Inverse Gaussian, Gamma, Frechet, and so on) and none of the distributions was a good fit.

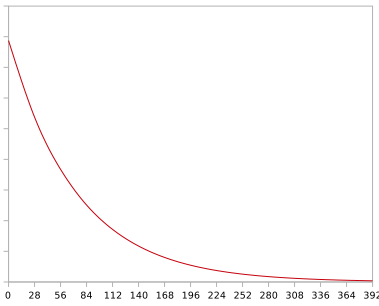


Fig. 6. An exponential distribution

Regarding the time between updates, we observed that the average is almost 2 months and a half (2.45 months). This value is, nonetheless, not that useful since the variance is large. The standard deviation is almost 2 months (1.8 months to be exact) and 75% of the updates are spread between 21.4 days and 4.6 months. A prediction based on these statistics would be highly inaccurate. Furthermore, the nature of the process suggested that the time between 2 updates of the same module could be modeled as an exponential distribution but standard statistical tests showed that this is not the case. Therefore, it is not a good modelling approach to assume that all modules behave similarly in terms of the time between 2 updates.

On the other extreme, building a model for each module independently is not feasible, either. In average, there were less than 2 updates a year per module, not enough data to build a reliable model for each module.

We plan, instead, on identifying the distribution (if any) that each module belongs to and its parameters (rate, mean, etc) based on a subset of features that best describe each module (e.g., size of the rpm, version) and update (e.g., security vs recommended update) as well as other parameters such as time of the year, distance to the official release date, and the dependencies among different modules. We plan as well on studying if the same module behaves similarly in different distributions.

In terms of coding, we plan on expanding our application with a back-end database that provides both a dataset with all the updates of the latest distributions and basic statistics, a web-based front-end where one can select aggregated statistics and predictions based on a subset of modules, and a background process that dynamically updates the prediction models.