Predictux: A Framework for Predicting Linux Kernel Incremental Release Times

Subhajit Datta
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4530, USA
Email: sd05@fsu.edu

Robert van Engelen
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4530, USA
Email: engelen@scs.fsu.edu

Andy Wang
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4530, USA
Email: awang@cs.fsu.edu

***Abstract*— Reliable software systems typically have a version release mechanism that is well organized and documented. This can be drawn upon to predict release timelines, which is helpful in gauging the quality of the software development and maintenance activity. In this paper we present initial results from developing and applying *Predictux* – a decision-tree-based framework to predict release times of Linux kernel versions. We compare predictions from the framework with actual data and discuss our future plans for refining Predictux further.**

I. Introduction and Motivation

Reliability of a software system depends to a large extent on the development time invested in a particular release. Development time is influenced by a number of factors, not the least of which is the initial estimate committed to key stakeholders. Although sophisticated software estimation techniques exist, their use is often too involved for quick and reasonably accurate "ballpark" predictions of how long a particular release is likely to take. We use *release* to mean a subset of a software system's functionality that is released to users for testing, use, and feedback. In this paper, we present Predictux, a decision-tree-based framework for predicting how many days the next Linux kernel version will take to be released, based on analyzing some parameters of its past releases. Linux was chosen to apply and test the framework since information regarding its releases are easily available in the public domain [6], and its releases are organized through log files and well-defined naming conventions etc.

Breiman et al.'s book *Classification and Regression Trees* [1] gave wide visibility to the use of tree-like structures in the process of knowledge discovery [3]. The decision-tree approach described in [1] is commonly referred to as the CART algorithm. "A Decision Tree is a tree-structured plan of a set of attributes to test in order to predict the output" [7]. Knab et al.'s paper presents a decision-tree-based mechanism for predicting defect density using evolution data extracted from the Mozilla open source web browser project [5]. Izurieta and Bieman's paper examines the evolution of FreeBSD and Linux at the system and sub-system levels, by studying the growth rate measures and plotting them against release numbers, release calendar dates, and by code branches [4]. We draw upon some of these ideas to explore whether a decision-tree-based framework can help us predict Linux kernel release times. The use of decision-trees was inspired by the ease of understanding and interpreting them. In the next few sections we describe Predictux, discuss its experimental validation as well as open issues and future work.

II. The Predictux Framework

Predictux is built around the hypothesis: *Incremental release times of Linux kernel version releases can be predicted through a decision-tree model based on certain parameters of past releases.* The parameters of past releases considered are *number of files added*, *number of files changed*, *number of files deleted*, *number of lines added*, *number of lines changed*, *number of lines deleted* – the *predictor* variables – and *incremental time in days between successive kernel versions of Linux*, which we will call *incremental time* – the *target* variable.

While designing and applying Predictux, we consider the following strategy: Extract values of the predictor variables out of the release logs, build a data set from it, use the data set for building, pruning and learning of a decision-tree, predict the values of the target variable using the decision-tree, and evaluate the accuracy of the predicted versus actual data. Based on this, the major functional areas of the framework are identified as a *pre-processor*, which will parse release logs (a sample log may be found at `http://www.linuxhq.com/kernel/v2.5/index.html`), extract relevant information, and the build data set; a *decision-tree analyzer*, which will build the decision-tree, and make predictions using the tree. A set of Java components were developed to serve as the pre-processor. The data set was fed to the *DTREG* [2] software for building the decision-tree, its subsequent pruning and learning and for predicting the values of the target variable. The data set consisted of 586 rows of data from Linux kernel release 1.0.0 to 2.5.75, containing the predictor variables mentioned earlier. The whole decision-tree generated from the data set consisted of 135 nodes, which was pruned to the one in Figure 1 to predict the incremental times for the 20 releases from versions 2.6.20 to 2.6.1. The method used by DTREG to determine the optimal tree size is V-fold cross validation [2]. We recognize the fact that Linux versions we used to build the data set are very different kernels. We make the assumption that even when a piece of software goes through generations of changes, the amount of work involved (which influences the incremental

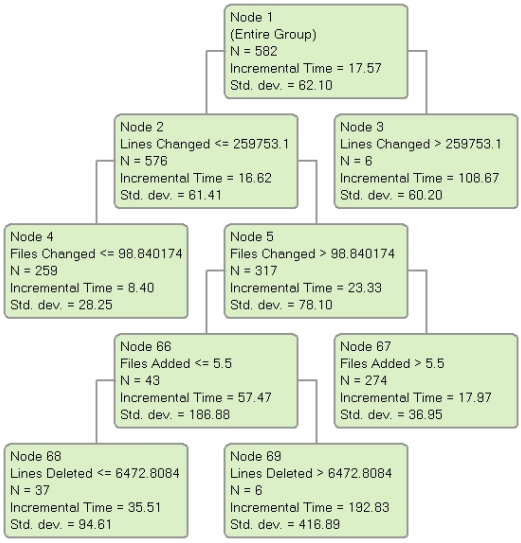

Fig. 1. The pruned decision-tree

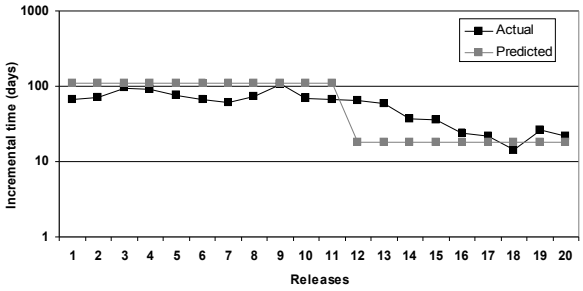

Fig. 2. Incremental times of Linux kernel releases 2.6.20 to 2.6.1: Actual and Predicted

release times) to evolve the software can still be predicted using our set of parameters.

III. Experimental Validation

The predicted and actual incremental times for the 20 releases are shown in Figure 2. Figure 3 shows the percent *deviation* – calculated as, (*Predicted incremental time - Actual incremental time)/ Predicted incremental time* * 100% – 16 out of 20 (80%) of the predictions that lie within ±45%. These have a mean deviation of 30%. 14 out of these 16 (70% of the total) predicted incremental times are within ±40% of deviation, and have a mean deviation of 27%.

IV. Open Issues and Future Work

The Predictux framework in its current form has a number of limitations. We use a data set with only 586 rows to build and train the decision-tree, which can be enhanced to include more release data. Moreover, we take parameters such as the *number* of files changed etc. as predictor variables without considering the actual functionality introduced or modified by the changes in the files. We are also not considering *patches* in the analysis even as sometimes major bug-fixing takes

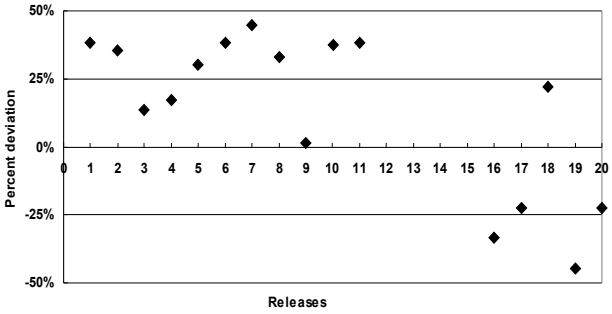

Fig. 3. Percent deviation between predicted and actual incremental times

place through them, which affect the timing of subsequent releases. To address some of these concerns, we are in the process of developing an efficient and reliable natural language processing tool which will automatically *read* change logs, understand the scope and context of the specific changes that lead to a new release, and refine the predictor variables based on such understanding. We are also examining how our approach compares to other prediction techniques; whether Predictux can be extended to become a general purpose prediction framework by applying it on other software systems; and whether we need to consider additional predictor variables which reflect issues such as developer skill, organizational maturity, problem domain etc. which may influence the timing of a system's releases. Another question of interest is whether reliability data of past releases – such as Mean-Time-Between-Failures – can serve as effective predictor variables for future release times.

V. Conclusion

In this paper, we have presented the decision-tree based Predictux framework for predicting the incremental release times of the Linux kernel version releases. 70% of the total 20 predictions for Linux kernel releases from 2.6.20 to 2.6.1 are within ±40% of the actual incremental release times, with a mean deviation of 27%. Our ongoing and future work is aimed at increasing the prediction accuracy by refining Predictux, as well as exploring the framework's application on other software systems.

References

[1] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. *Classification and Regression Trees*, new ed ed. Chapman and Hall/CRC, 1984.

[2] DTREG. Dtreg: Software for predictive modeling and forecasting. `http://www.dtreg.com/`, 2008.

[3] Groth, R. *Data Mining: Building Competitive Advantage*. Prentice Hall PTR, 1999.

[4] Izurieta, C., and Bieman, J. The evolution of freebsd and linux. In *ISESE '06: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering* (New York, NY, USA, 2006), ACM Press, pp. 204–211.

[5] Knab, P., Pinzger, M., and Bernstein, A. Predicting defect densities in source code files with decision tree learners. In *MSR '06: Proceedings of the 2006 International Workshop on Mining Software Repositories* (New York, NY, USA, 2006), ACM Press, pp. 119–125.

[6] LinuxHQ. Linuxhq: The linux information headquarters. `http://www.linuxhq.com/`, 2008.

[7] Moore, A. Decision trees: A tutorial. `http://www.autonlab.org/tutorials/dtree.html`, 2007.