Determining Software Product Release Readiness by the Change-Error Correlation Function: On the Importance of the Change-Error Time Lag

Roman Wild
University of Applied Sciences Neu-Ulm, Germany
email@Roman-Wild.de

Philipp Brune
University of Applied Sciences Neu-Ulm, Germany
Philipp.Brune@hs-neu-ulm.de

Abstract
In software development determining the release readiness plays an essential role. The number of errors is frequently used as an important measure to decide about the quality of a software implementation. Therefore, error prediction techniques have been intensively studied in the literature for many years. Despite this, their adoption in practice is still strongly limited to date. In this paper, an alternative model for error prediction in software projects based on linear-response theory and the change-error cross-correlation function is proposed. It is applied to data collected in projects of a major embedded systems vendor in the communication industry. Under similar conditions, a universal behavior of the change-error cross-correlation function is observed. Moreover, a time lag of 4-6 weeks between the implementation of changes and the detection of related errors is discovered. This clearly demonstrates that for reliable release decisions not only the current number of errors but also of changes is essential.

1. Introduction
Today’s life depends strongly on software. Nearly every electronic device actually is an embedded system using software of many hundreds or thousands lines of source code. And software complexity is continuously increasing. Thus, in the development cycle of electronic devices increasing emphasis is put on quality assurance efforts.

The number of bugs or defects (referred to as errors in this paper) detected in functional software testing is frequently used as an important measure to decide about the quality of a software implementation. In practice, a sufficiently small number of detected errors is usually used as the main criterion for scheduling a new releases of a software. Therefore, software error prediction techniques have been intensively studied in the literature for many years [1-3].

On the other hand, in recent years data mining has been successfully applied in domains not related to software development. In many practical applications it is already highly adopted [4-8]. Consequently, various data mining techniques have also been applied to software repositories to predict the future time-evolution of errors by means of historic data [9-12].

However, most previous work in this area can be classified as white-box approaches, based on detailed knowledge of the source code and the respective development process and environment and trying to predict errors also on the level of source code artifacts [2].

In this paper, a more phenomenological approach is presented. An error prediction model is developed considering the developed software product together with the whole process and environment as a black-box system. This is modeled as linear system in time, taking the number of software changes as input and the number of errors as its respective output. Linear response theory [13] is used to obtain a transfer function for this system, describing the dependency of errors on the changes during the development process. The relative simplicity of the approach is intended to support an easier implementation and thus wider adoption in practice.

The model is applied to real data from software development of a major embedded systems vendor in the communication industry. The model proves to describe the observed time-evolution of errors detected in testing quite well. For a selection of products with an identical development process the obtained transfer functions qualitatively show a characteristic universal behavior. Moreover, a time lag of 4-6 weeks is discovered between the implementation of changes and the detection of related errors during testing. This result clearly indicates that a temporarily vanishing number of errors is not a reliable criterion for scheduling new releases. Errors may show up with a significant delay. Reliable error prediction models thus need to take also software changes into account.

To implement the model in real-world projects, an appropriate and easy-to-use software tool is required. Open source tools provide a powerful yet easily available and cost-effective solution for most tasks in software development [14]. Thus, the relevant open
source software tools for data-mining are compared and evaluated with respect to implementing the proposed model. The results of the evaluation and the user experiences are presented.

The structure of this paper is as follows: Section 2 discusses the related work. Section 3 outlines the real-world context and practical motivation for the specific analysis. In section 4, as a prerequisite for the implementation of the model, the most commonly used open source data mining tools are evaluated and a suitable selection is made. Section 5 describes the proposed methodology based on linear response theory and the assumption made in this model. In section 6 the actual data analysis based on this methodology and using the selected tool is presented. The obtained results for the correlation between changes and errors are discussed in detail in section 7. We conclude with a summary of our findings.

2. Related Work

For many years, error prediction has been an important field of research in software development. Previous work i.e. examines different types of multivariate regression models to predict future error-prone source code areas [20]. Other authors use software metrics as a basis for the prediction [10, 15]. In recent years, the influence of previous software changes on the error distribution in code artifacts has also been studied [15-17]. Various attempts to measure and compare the effectiveness of the different techniques and to improve the prediction quality have been published [2, 3, 18-20].

In a significant part of the data mining literature, successful real-world applications of data mining techniques to different domains are examined. This is especially true for data consisting of nominal attributes, like i.e. the prediction of rock strength and deformability [4] or even the prediction of wine quality [5].

However, the degree of real-world adoption declines when it comes to time series analysis. Even though methods like regression for prediction are quite commonly used, e.g. to make sales forecasts [6] or predict the development of stock prices with different models [7, 8], published results on real-world case studies using clustering, rule learning or others are sparse [21]. Some authors even state that the clustering of streaming time series data is meaningless at all [22].

Recently, applying data mining techniques to software repositories has attracted considerable research interest [9-12]. Most of this work focuses on using i.e. source code or process measures as the basis for predicting source code artifacts (i.e. files, classes or modules) where errors are likely to be found in the future [20].

All the mentioned error prediction models have in common that they can be classified as white-box approaches. Closely related to software metrics they are built on analyzing source code artifacts or development process and environment data [2]. Since implementing these approaches requires a comparably high effort regarding know-how and tools, their adoption in practice is still limited to date [20].

In this paper, an alternative „black-box“ approach based on linear-response theory is presented. By means of an appropriate data mining tool it has been successfully applied in a real-world software development process with comparably little effort.

3. Project Context and Motivation

The data captured during the software development process at the major vendor of embedded systems is stored in a core data warehouse and thus is available for data mining tasks. These time series data contains i.e. the open errors, requirements, testing results and implemented changes over time. For the present analysis, these valuable data has been available on a weekly basis and for different products. These data are an important cornerstone of the quality management process and for sales readiness decisions in the development division of the company.

Currently, these data is distributed to the program and product managers in the form of metrics. The managers are able to interpret these data only informal and intuitively due to their experience and knowledge from the past. Ongoing activities aim to integrate data mining mechanisms into the development tool chain in order to formalize and automate the evaluation of the current state of a product under development.

Possible applications of data mining in this context might be:

1. The prediction of trends in the different data domains (errors, changes, etc.) utilizing a model based on the data gathered in the past. The data of further domains could be used as overlay data for the prediction [23].

2. Clustering and later classification of products based on their past performance in order to find the right benchmark products [23].
Rule learning to underpin informal knowledge and formal integration in early warning systems. Possible found rules could be: If the amount of changes is dramatically increasing in a timeframe of two weeks before sales start, then the likelihood of an high released error count is increasing (due to missed side effects of fixes) [23].

Statistical methods like cross-correlation in order to find most probable time lags between implemented changes and related errors. This would allow making a sales readiness decision with a higher level of confidence.

Due to its high business relevance, application (4) has been selected to be examined first. The results are presented in the present paper.

4. Tool Selection

For implementing an error prediction model as a standard means of release schedule decisions, it must be integrated by default in the regular software development process and project decision tool chain. This requires automatization by a data mining tool usable for development project staff and managers. Thus, the selection of an appropriate data mining tool is crucial for the practical adoption of a prediction model. To fulfill these requirements, an appropriate tool especially needs to be easy to use for end users, allow for an efficient integration in the overall tool chain, and provide powerful support for time-series data analysis.

Figure 1 shows the results of the 2010 and 2011 KDNuggets poll [24]. Considering only open source projects, the obvious candidates for realizing the test case are RapidMiner, KNIME and Weka. R as standalone tool shall not be considered, as it is more a tool for statistics rather than the whole data mining process. Since the latest release of R is also integrated into RapidMiner, it is used implicitly.

In the following evaluation process the candidates where reviewed regarding the following criteria: The actual usage, the overall user interface (UI) quality and user experience (UX), the featured support for time series analysis, the possible integration into a complete business intelligence process, the extendibility with new algorithms or extensions, the amount and quality of documentation and support, the data input compatibility, the evaluation and validation support, and the project management support.

The evaluation of the three products showed that RapidMiner [25] seems to be the best candidate for the task. Besides the huge number of users, RapidMiner also features currently the best time series analysis capabilities. However, the biggest advantage is the complete integration of R. Furthermore, there is a large list of extensions available and the UI as well as the UX are very appealing and offer the user convenient features like problem highlighting and quick fixes.

The other products are also competitive, but they are slightly behind RapidMiner in respect of some criteria.

KNIME [26] has also an appealing UI, but it lacks exhaustive out-of-the-box time series support as it mainly features an extension for basic date instructions like “Date Field Extractor” or “String to Date/Time”.

Weka [27] can be considered as the mother of all open-source data mining tools. However, its UI appears quite old-fashioned to normal end users. This can be considered the biggest weakness of Weka. Thus, Weka might be the best choice for very experienced users or primarily scientific applications.

In the evaluation process the nine criteria were weighted according to their relevance in respect of the
project requirements and the company’s needs. The weight is ranging from 1 (less important criteria) to 10 (very important criteria). After that each criteria received a rating as well in the range of 1 (does not support the criteria) to 10 (supports the criteria in a very high degree). The product of weight and rating was calculated for each criterion and then summed up over all criteria in order to receive the resulting score for each product.

Table 1 shows the weight of the different criteria as well as the individual rating and score of each product.

Table 1. Criteria, weights and scores of the evaluation

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Rating</th>
<th>Score</th>
<th>Weight</th>
<th>Rating</th>
<th>Score</th>
<th>Weight</th>
<th>Rating</th>
<th>Score</th>
<th>Weight</th>
<th>Rating</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adoption</td>
<td>2</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>2</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>User Interface and Experience</td>
<td>3</td>
<td>8</td>
<td>24</td>
<td>5</td>
<td>15</td>
<td>9</td>
<td>5</td>
<td>15</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Support for time series analysis</td>
<td>6</td>
<td>8</td>
<td>48</td>
<td>6</td>
<td>36</td>
<td>4</td>
<td>24</td>
<td>6</td>
<td>36</td>
<td>4</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>Integration into the whole BI-Process</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expandability</td>
<td>8</td>
<td>9</td>
<td>72</td>
<td>7</td>
<td>56</td>
<td>6</td>
<td>48</td>
<td>7</td>
<td>56</td>
<td>6</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>Documentation and Support</td>
<td>9</td>
<td>8</td>
<td>72</td>
<td>6</td>
<td>54</td>
<td>6</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Input</td>
<td>8</td>
<td>8</td>
<td>64</td>
<td>6</td>
<td>48</td>
<td>6</td>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation and validation</td>
<td>7</td>
<td>8</td>
<td>56</td>
<td>6</td>
<td>42</td>
<td>5</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Management and Support</td>
<td>5</td>
<td>8</td>
<td>40</td>
<td>4</td>
<td>20</td>
<td>3</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUM</td>
<td></td>
<td></td>
<td></td>
<td>399</td>
<td></td>
<td>286</td>
<td></td>
<td>287</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Following this evaluation, RapidMiner was selected for the proof-of-concept implementation. Unfortunately, also RapidMiner doesn’t feature by default a cross-correlation-node as required by the proposed model. However, it is possible to integrate the statistical programming language R into RapidMiner. This feature is especially useful when it comes to time series data analysis.

With R the cross-correlation can easily be calculated by using a pre-defined command in an R-Script. In order to use R with RapidMiner the according extension from the repository needs to be installed. This can be done by using the integrated package manager. Besides this it is required to install a local instance of R to the system. R itself needs to be prepared by expanding it with a plug-in in order to communicate with RapidMiner. After RapidMiner is set up to use the R environment locally, the R nodes can be used in any data mining process within RapidMiner.

Using the R Console of RapidMiner also allows the user to produce a plot as output of the function. Therefore, the output connector of the “Execute Script (R)” node is connected with the output connector of the process.

In the case presented, RapidMiner proved to be a very sophisticated open source software which covers a huge variety of applications and is supported by a very enthusiastic community. Its UI and UX make it comparatively easy to use for end users.

5. Methodology

In the presented approach, the software products under development including the respective development environment each are considered to form a „black-box“ linear system. This system in general reacts on software changes at time t by errors detected at time τ, where τ > t. The system is described by means of the so-called linear response theory.

Originally developed by Ryogo Kubo for applications in theoretical physics [13], it is widely used today in other fields like information theory or engineering. In these areas the concept is also known as susceptibility, transfer function or impedance [28, 29].

Assuming that the errors and changes of software developed for a standardized embedded system are dependent on each other in linear form, their relation can be modeled by a transfer function or linear response function:

\[ y(t) = \sum_{\tau} a(\tau)x(t - \tau) + b(t) \]

where \( x(t) \) describes the deviation of the changes from their mean value and \( y(t) \) that of the errors at time \( t \), \( a(t) \) is the weight and \( b(t) \) is noise which is corrupting the system. The „noise“ in this model statistically describes all the errors not directly related to the changes, i.e. those resulting from accidental side effects. This is a so called transfer function-noise model [28].

It is based on the assumption that the input \( x(t) \) to a system causes a response \( y(t) \) such that \( y(t) \) depends linearly on all past values of \( x(t) \). The dependency is assumed to be linear, as the mean number of errors per line of code is regarded to be perpetual over all projects at the company. It is also assumed that one change is not influencing another change. Furthermore, it is assumed that the process is stationary so that the functions \( x(t) \) and \( y(t) \) have constant means as well as constant variances.
The impulse response of the system is then given by

\[ a(t) = \frac{R_{xy}(t)}{\sigma_x^2} \]

where \( R_{xy}(t) \) is the cross-correlation given by

\[ R_{xy}(t) = \frac{1}{N} \sum_{t'} y(t - t') x(t) \]

and \( \sigma_x^2 \) is the standard deviation of the stimulus impulse on the system [29].

So the data gathered in the considered software development process can be used for calculating the cross-correlation function and thereby its impulse response. Thus, calculating the cross correlation as stated above describes the dependency of errors and changes in a software development process.

6. Data Analysis

The change and error data of eight products were used in the proof-of-concept to calculate the respective cross-correlations.

In the data mining process the observations of both value pairs and a timestamp are loaded from a Spreadsheet which is an excerpt from the core data warehouse. It contains the error count and the number of changes for each week until the end of the development process. The results from the testing activities of a certain release are usually reported one week after the release was build.

After the import of the spreadsheet into RapidMiner, the data stream is passed to an “Execute Script (R)” node which is provided by the R Extension. To calculate the cross-correlation function, within that node the ccf() function of R is used, which receives the two data streams with the value pairs and the maximum time lag as parameter. With this process the values for the eight selected products were calculated and plotted in order to analyze the results graphically.

7. Results

For eight different products the change and error time-series data from the whole development process of many releases (approximately 48 per product) were used to calculate the respective cross correlation functions. All products were developed using the same development process, a company-specific sequential, waterfall-style process. The considered duration of the development was about 30 weeks. Other products which did not complete the entire process because of premature project cancellation or termination due to other reasons were not considered.

The eight products were chosen according to form a representative sample out of all previously developed products of different types and development branches. Furthermore, when selecting the set of products the focus lied on a complete time-series data set of high quality.

Changes in this case represent the number of actual source code modifications which are implemented into a new software release compared to its predecessor. The changes are measured with help of the version control system, which detects every change to the source files. This measure is seen as a risk indicator and was already used before in decisions regarding the release quality, because it is assumed that more changes can potentially lead to more new errors. The error numbers in this case are extracted out of the error management database.

In Figure 2 the calculated change-error cross-correlation functions are plotted on the y-axis and the time lag on the x-axis with each time unit representing one week, the usual reporting cycle in the development process at hand. Each curve represents the result for one product. It can be easily seen that the cross-correlation functions for the considered test group of eight products exhibit a significant universal shape as a function of time. In fact, the main peaks of all curves lie in between four and six weeks on the x-axis.

This means that an implemented change leads to new errors due to side effects approximately after four to six weeks after its implementation. This result shows clearly that a low number of known errors is not a sufficient criteria when it comes to the decision regarding the release readiness of a product at the end of the development process.

When the number of implemented changes at the time of the decision is greater than zero it is very probable that there will be a number of released errors missed, which are caused by the late implementation of changes. Thus it is very important to also regard the count of changes when deciding about the release schedule of a software product.

In the beginning, the cross-correlation is negative for all products, indicating that an increase in the number of changes leads to a decrease in the number of errors. This is due to the fact that testing activities usually start later than the implementation work. After testing starts, the cross-correlation becomes positive as expected.
Verifying the dependency of errors on changes justifies taking changes into account when making decisions regarding release readiness and enables quality authorities to use it on an evident basis. With the proposed model it would also be possible to predict an error count in the future. This count then should be added to the number of known errors in order to get the real number of released errors. Based on this count the decision if a product is ready for release could be taken with a higher level of confidence.

In the company, within two or three years data mining might be integrated in the decision making process of the so-called product quality gate approving release readiness in the company. This is feasible and would definitely improve the value of the existing tool chain.

8. Conclusion

In conclusion, the presented results describe the dependency between the implemented changes and thereby caused errors in software product development for embedded systems. This dependency is obtained using linear response theory with time series data representing the errors and changes.

The results indicate that the linear response theory is applicable to that dependency. A transfer function can now be used to estimate the number of errors resulting from software changes. There exist several different ways to build the transfer function out of the calculated cross-correlation function. The most promising solution for prediction would be the combination of different regression techniques, leveraging the benefits of more than one prediction mechanism. The application of a Support Vector Machine algorithm should be taken into account here, as it looks promising regarding the characteristics of the data at hand.

In the presented case an approximate time lag between input and output of the system of about 4 to 6 weeks is discovered. The actual duration is a result specific for the particular development process, mainly impacted by the long testing cycle.

It can be assumed that these findings are applicable to other areas of software development. However, it remains an open issue to verify this for different project setups. Further research is needed to examine if and under which conditions the model assumptions, i.e. the linear dependency between changes and errors, are valid for different types of projects.

This work clearly stresses the importance of considering also the number of changes in the decision regarding the release of a software product. In general, the relation of changes and errors needs more attention when the quality is judged – this dependency can no longer be questioned.

9. References


