Paper 1: Challenges associated with software logging in Large system

Background on the logging statements:

- **Logging statements record useful information about the state of a system during its execution**
- **Logging statements are evolved from the basic printf statements to logging libraries**
- **Log4j is one of the basic libraries introduced in 2002**
- **Logging libraries provide a centralised output location**
- **Abstraction libraries allow several different logging libraries to be used within the same project**
- **Unification Libraries provide features of both basic and abstraction libraries**

Challenges in modern-day logging

- **Infrastructural Challenges**
    - **Logging library Migration - a case study fo the Apache software foundation projects**
        - **Developers migrate from one logging library to another**
        - **How often Migrated: 33 projects successfully migrated**
        - **But logging migrations are not justified**
        - **Dev should evaluate the feasibility and the performance improvements to avoid the post-migration**
    - **Processing Challenges - Examining the stability of Logging Statements -**
        - **TB of logs in every hour**
        - **How often do logging statements?**
        - **Can we determine logging statement changes**
        - **From git repository extract and track logging statements : (commit history)**
        - **From the extract and tracked loggins statements - (changed and unchanged logging statements) - > collect content, context and developer metric**
        - **- > preliminary analysis , random forest and cox model**
        - **From random forest - > just introduced logging statements**
        - **Cox models - long-lived logging statements**
    - **Abstraction libraries JCL, Slf4j , Unification libraries Logback, Log4j2**
    - **The metric to determining the likelihood of logging statements are developer experience and file ownership**

- **The stability of logging statement is very important**

Paper 2: Which log level a developer should choose for a new statement

Logging statements are used to record valuable information about the runtime of any application. Each logging statement is assigned to a log level such that user can disable some verbose log message while allowing the printing of other important ones. However, prior research find that developers often have difficulties when determining the appropriate level for their logging statements.

This paper considers the development history of four open-source projects and leverage the ordinal regression model to automatically suggest the most appropriate level for each newly added logging statement

Characteristics of the containing block of newly added logging statement the existing logging statements in the containing source code file and the content of the newly added logging statement play important roles in determining the appropriate log level

Intro : Logs are widely used to store the runtime information of the software systems

A logging statement normally specify

1. **Log level : debug/info/warn/error/fatal**
2. **A static text**
3. **One or more variables**

Eg: logger.error("static text" + variable);

Both logging a little (miss important information) and logging too much(system runtime overhead) is undesirable

Mechanism of log levels helps in the tradeoff of the rich information and overhead
Most common logging libraries are Apache Log4j, Logging, SLF4J
Support six log levels trace, debug, info, warn, error and fatal
Where trace - is the most verbose and fatal is the least verbose

Uses the ordinal regression because the six classes has a priority value

- **And this paper is trying to address mainly two issues**
- **How well can we model the log levels of logging statements**
- **What are the important factors**

The existing statements in the file and the content of the logging statement have a great influence on what is the log level of the given statement

http://svnbook.red-bean.com/en/1.7/svn.ref.svn.c.log.html

This website give that the command svn log will display the commit log messages in a given repository
Which is used by the paper authors

This paper uses the svn logs to get the retrieve the development history for each project - uses `use-merge-history` option in svn log.
This will help to reduce the noise due to the code revision and the repetition of code in the code merge

- **Source line of code - Hadoop 485K**
- **Number of logging statements added in the project - 5388 1.2 - 4.6 % experienced the log level change eventually**

Data Extraction
---

1. **Code revision from the version control repository of the studiec project**
2. **Log change identification**
3. **Added Logging statements**
4. **Determine log levels and other metrics**
5. **Run the ordinal regression**
6. **Predict the log level (how well)**
7. **Predict the determining factors of log level**

Preliminary studies

1. **No single level dominates the other log level**
    a. **Trace , fatal and 4 middle levels**
    b. **Most frequents are the middle levels**
2. **Different distribution of log levels across the code block**
    a. **They used AST abstract syntax tree by eclipse JDT to identify the containing clobck of each logging statements**
    b. **The logging statements in the catch blocks are more likely to use less verbose whereas logging statements in try blocks are more verbose log levels**

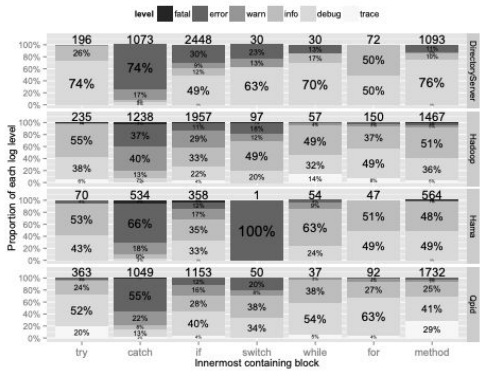

Fig. 3 Log level distribution in the added logging statements in different types of blocks.

3. **This project collected the features in 5 levels -**
    a. **Logging statement metrics**
    b. **Containing block metrics**
    c. **File metrics**
    d. **Change metrics**
    e. **Historical metrics**
4. **Before constructing the ordinal regression model they calculated the correlatio between the collected metrics using the pearson correlation test. They used the varclus function R package**
5. **They used a backward step-down variable selection - it helps to determine statically significant variables**
6. **Evaluation is based on AUC and Brier score - used to measure the accuracy of probabilistic predictions**
7. **They used bootstrapping to avoid the overestimation (optimism) - the general approach is to infer the relationship betweenthe sample data and the population by resampling the sample data with replacement and analyzing the relationship between the resampled data and the sample data - subtract the bootstrap-averaged optimism fro the original performance**
8. **They proved that ordinal model has a better performance over other models**
9. **The second motivation of this paper was to determine which all are the feature which defines the level of the logging statement**
    a. **Wald chi square test: measure the importance of the particular variable**
        i. **The containing block determines the level of the log statement**
        ii. **Containing block code lines also determines (only 6% in hadoop)**
        iii. **FIle metric is also impact on the log level**
        iv. **Logging statement metric also has an important impact on the log level**
        v. **Change metric and historical metric are the least important in explaining the choices of log level**

Step 1: Get the code from git. The author is using svn
https://git-scm.com/book/en/v2/Git-Tools-Searching
https://git-scm.com/book/en/v2/Git-Basics-Viewing-the-Commit-History