

# Event Correlated Usage Mapping in an Embedded Linux System

## A data mining approach

Oscar Linde ama09oli@student.lu.se, Anton Norell ada10ano@student.lu.se

**Keywords**—Data mining, Generalized Sequential Patterns, Debugging, Logging, Embedded Linux.

### I. INTRODUCTION

Today we are often surrounded by hundreds of digital devices that makes everyday life easier. Many of them are hidden and not even noticeable or interact able. These devices are often referred to as embedded devices and serves a single purpose such as controlling a dishwasher, an alarm clock or access to a door. An embedded device often consists of a processing unit, which can execute some program, other hardware and surrounding electronics mounted on a circuit board. The simplest devices will run a single program line by line, while others are much more complex running operating system with support for multithreaded real time applications. A software system running on an embedded device is in many cases hard to monitor and understand since different processes interact in a complex way and it may lack the use of graphical user interface. Automated test can be created to see if the result of the execution is the expected. However, as the scale and complexity of the software grows, containing thousands of lines of code, those ways of finding errors gets time consuming and can miss errors. The problem with this approach is that some problems depends on timing, synchronization and communication between different processes running on the system. To get an insight of the system behavior and to be able to find those hidden errors, one can use different techniques to log and monitor the system. The information gathered is useful for improving the system performance and can help developers to understand what caused a malfunctioning. This work explores different techniques of logging and will investigate the possibilities to analyze logs with the use of data mining, on a complex system consisting of multiple interconnected devices. Data mining is a subfield of computer science that focus on extracting knowledge and patterns from large amount of data. The test platform is an embedded Linux device that runs a system with several processes communicating with each other via the Linux data bus.

### II. DATA COLLECTION

To be able to collect data that is interesting when debugging an embedded Linux system, we have implemented a data collector. This collector runs as a daemon process, which basically is an application that runs in the background. To find proper information to log the data collector is using a

file system that exists on Linux which is called the proc file system (procfs for short). This is a file system that reflects the state of a Linux system and provides information such as memory usage, CPU load and process status. Another thing that the data collector uses in order to retrieve information that is valuable to measure how the system is behaving is Dbus. Dbus is a messaging protocol with the purpose of making it easier for processes to communicate with each other. The system used during this project uses this bus to send standard message about what is happening inside the system. All this information is stored in a database and when something goes wrong, the data collector takes a snapshot of the database and prints a debug message. Also, if a certain compile flag is used, the data collector generates a function trace that simply trace every function call that is made on a certain application. Snapshots of the function trace is generated at the same time as the snapshots of the database. The snapshots allows us to go back and see how the system was behaving when something went wrong, and also it gives us logs to perform data mining. An overview of our data collector can be seen in Figure 1

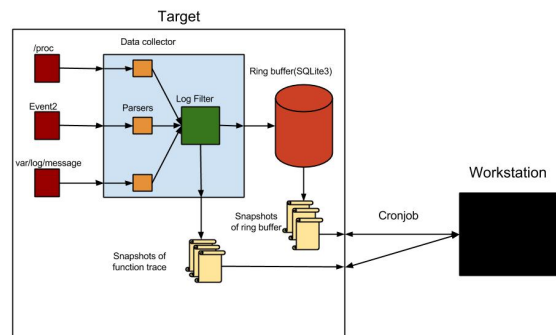


Fig. 1. An overview of the data collector

### III. DATA MINING

Data mining (a.k.a Knowledge Discovery in Databases) is a term referring to multiple methods and algorithms for extracting patterns and information from large data sets. The algorithms can be divided into different classes, in this work the focus is on applying the class association rule mining on the collected data. Association rule mining is one of most common methods were relationship and patterns of frequently occurring items are searched for. A classic example of association rule mining is the mining of customer basket data.

If a customer buys potatoes, she is likely to buy salad too. The algorithm might then find e.g. that this is true for 82% of the cases. In this work we choose to use an association algorithm that takes the the sequence that the data is logged into account. The algorithm is called Generalized Sequential Pattern (GSP) and is a data mining algorithm developed to find frequent sequential patterns of items in large data. The GSP algorithm basic structure is as follows. The algorithm goes through the data several times and builds up sequences. In the first iteration the support of each item is calculated, the support is the frequency of an item that exist in all the item sets. In the next iteration all the items that exceeded the minimum support is used to build sequences of two items. Then the support of all the sequences with length two is counted and those that exceeds the minimum support continues to the next iteration. The algorithm continues with iterating over the data increasing the length of the sequences with one item each time until there are no more sequences exceeding the minimum support. Except for the minimum support, the algorithm has some rules when it combines items to build sequences. The other parameters are window, minGap and maxGap, which can be set to obtain different properties.

#### IV. EXPERIMENTAL RESULT

To try out our findings and the data collector that we implemented, we ran several tests to verify. In the first test we planted an intentional bug that basically generated a faulty dbus message if the data collector received a certain sequence of dbus messages. We divided logs from this test into bad logs (where the faulty dbus message had appeared) and good logs(no faulty dbus message included) and ran them through our data mining process to extract unique sequences of dbus messages that only exists in the bad logs. An overview of the data mining process is shown in Figure 2. In a second test, we

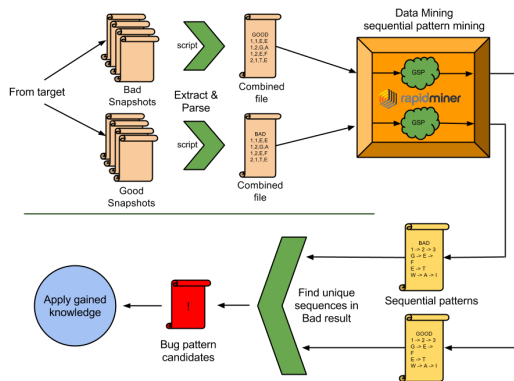


Fig. 2. An overview of the data mining process to extract bug patterns

used a different setup. Instead of looking at dbus messages, we looked at function traces generated when a process was leaking memory to be able to isolate which functions could include the reason behind the memory leak. To stress the system we ran a test that was known to generate a memory leakage in order to generate logs that were generated during this behavior.

#### V. CONCLUSION

The results from the the tests showed that our approach to this problem actually works. By planting a bug sequence inside the system we could extract that with our data mining process and we were also able to narrow down what part of a system that could be the cause of the memory leakage by using data mining on the function trace generated by our data collector. By running the logs generated by our data collector through our data mining process we were able to minimize the sequences of interest from tens of thousands to a hundred.