

A *First* Approach to Understanding the Anomalies on My PC ^{*}

Luis M. Fernández, Hugo Terashima-Marín, Manuel Valenzuela-Rendón

{A00789695, terashima, valenzuela}@itesm.mx
Center for Intelligent Systems
Tecnológico de Monterrey, Monterrey Campus
Monterrey, Mexico

Abstract. Nowadays, and more than ever before, we live on a *computerized* world. Computers are used every day, all day, all around the world; they come in so many shapes and sizes: while laptop and desktop computers are commonplace, computers can also be found in home electronics, automobiles, airplanes, etc. Many of the world activities now rely on PCs and, therefore, rely on the perfect execution of the applications running in these machines.

Failing to appreciate the importance on the correctness of any application may cause some serious problems. Just to show this, consider the power grid's computers that keep power flowing in 15 USA states and one Canadian province. More than 50 million people were left without electricity on August 14, 2003, thanks in part to a software bug [Hessendahl A. , 2007]. Moreover, consider the 13 Root Domain Name servers, two of which are operated by VeriSign, that each day receive 25 billion requests for Web addresses. This system is a frequent target of computer malcontents, and a February 6 attack caused three of the servers to drop 90% of their queries during a 12-hour period [Hessendahl A. , 2007]. All this gave birth to a set of techniques and tools to deal with these deviations, known as *Anomaly Detection*, which can be described as an alarm for strange system behavior [Tanase M. , 2002].

The anomaly under study in this *first* approach is the one present when a PC freezes as it executes some applications. It is believed that there are certain combinations of applications that are more likely to generate this anomaly, which has the potential to make someone's ongoing work disappear. The approach followed in this starting research is to combine simulation tools to generate instances of applications; and machine learning algorithms to provide rules of the kind $p \rightarrow q$ that may lead to an anomaly.

This starting point is based on the assumption that when all resources of CPU are in use, this *anomaly* takes place. This is not far from the truth since the more programs make use of CPU, the more waiting time for other programs to access CPU resources; thus, slowing the PC down or even freezing it. We are aware of the fact that there are more issues related to CPU-resource allocation, but as the title suggests, this is a *first* approach to trying to understand and prevent anomalies.

^{*} This project is funded by Tecnológico de Monterrey, Research Chair CAT 010

The simulated environment was coded using TurtleKit, a plugin of MadKit [Michell F., 2002]. In this simulation each running application was represented by a different agent. Another agent was in charge of *spawning* these agents based on how frequently the applications are used, for how long and how much CPU resource consumption they involve. These statistics were provided by the MugShot web page [MugShot, 2007]. This spawning agent decides which application to instantiate using the Genetic Algorithm roulette wheel selection technique. Another agent, in this simulated world, is in charge of recording how much CPU would be in use at the current time if the applications, represented by agents, would be running on a PC. This agent not only records which applications are running but also, whenever the CPU usage reaches to 100%, it associates the current list of running agents with a *possible-anomaly* flag. For the present research, this simulation was run ten times and each lasted for five hours. This produced 2GB of data to work on. Once the data was collected, it was necessary to process it and get meaningful information. However, the produced data was quite huge which made it impossible to use brute force. Machine learning algorithms were used, more specifically, the a priori algorithm, k-means and farthest first. In order to process all the data, the “Yet Another Learning Environment (YALE)” suite was used [Mierswa I., et. al., 2006]. The advantage of having used the already mentioned tools for simulation and for processing the data was given by the fact that both were coded in Java making a combination of both something smooth and fast.

The obtained results after using the a priori algorithm showed that there are certain applications that tend to freeze the PC more than others. A combination of a presentation utility and an email manager, a word processor and a presentation utility, an email manager and a document viewer were the most common applications that may lead to an anomaly. It was also found that a simple music player could make a PC freeze. This knowledge could help prevent future crashes but, at the same time, all these results were obtained by simulated a simpler-than-life PC.

It was also wanted to see if the lists of application flagged as possible anomalies had something in common. As such two clustering algorithms were used, the k-means and the farthest first. After applying both algorithms, it was easy to see that they do share certain characteristics by clustering into two defined groups. The results were also validated using Andrew’s Curves [García-Osorio C., 2004], obtaining similar results. This research provided some conclusions. First, the number of lists of applications that could lead to an anomaly is very small, only thirteen were found. This makes sense since most of the times, computers work flawlessly. Second, the number of applications that could lead to an anomaly is also very small. The obtained rules showed that only five applications out of twenty-one might represent some threat. Nevertheless, there are still some things left to be done, and the most important one is to go beyond the simulation and actually get real data, which will help discover false positives and false negatives. Second, it is necessary to use all this knowledge to warn of *possible* anomalies as someone uses a PC. This could be implemented using an agent that monitors the applications in use, alerting the user when any of this found instances are present.