**Knight Capital Glitch**

CSC 179-01

Assignment 01 Cost of Quality

9 February 2016

Daniel Jackson

Suzanne Kharaka

Kurt Larson

Sebastian Winarto

The Knight Capital trading glitch occurred on Wednesday August 1, 2012 and may very well be the costliest software bug in history. (Popper 2012) Knight Capital is a large player in high speed, high cost trading of stocks. They utilize sophisticated algorithms to detect when to buy and when to sell. The bug in question caused the algorithm to trade irrationally and ended up buying an enormous amount of incorrect stocks. The impact that this caused on the company was detrimental, all of which could have been avoided by properly testing their software prior to deployment.

As a result of the software glitch, the blunder cost the company an estimated 440 million dollars in revenue and seriously damaged their reputation according to Bloomberg Business. (Phillips 2012) As the program began buying stock at an accelerated rate, the value of each stock had been rapidly driven up. When the company finally caught on to the issue, they were forced to sell the stock at a price that was well over the market value. During this time, the company was said to have been losing 10 million dollars every minute. (Phillips 2012) When all was said and done, the incident caused the value of Knight Capital’s stock to drop by 63 percent. Within four months of the software glitch Knight Capital entered into a merger with Getco to avoid bankruptcy and CEO Thomas Joyce leaving the company after the merger.

Unfortunately, this software bug could have been avoided by performing some simple tests prior to deploying to production. According to the article published in Network World, Knight Capital not only lacked in managing code development but coupling this with the limited personnel to monitor software code created devastating results. (Gibbs 2013) Although no specific fault has been made public by Knight Capital, a method called regression testing could have eliminated this tragic bug. Regression testing ensures that, for each additional modification, a new fault is not introduced; this is done by testing many of the same use cases throughout the life cycle of a system. (Naik and [Tripathy](http://www.wiley.com/WileyCDA/Section/id-302475.html?query=Priyadarshi+Tripathy) 17) Each additional designed test case, although costly, can help uncover any new faults, which will in return mitigate risk for the final software iteration. Even if Knight Capital employed this method, it is likely that the company did not utilize enough test cases to account for the fault that occurred. Having valid test cases would have better equipped Knight Capital to respond quickly to changes in software as well. In June 2012 the New York Stock Exchange received approval to implement a new method of trading called the Retail Liquidity Program (RLP). The RLP program was implemented less than two months after approval on August 1, 2012. Knight Capital experienced their software error within the first 30 minutes of trading August 1, 2012 (Heusser 2012) and clearly did not have sufficient test cases to validate the new program deployed by the New York Stock Exchange.

The people responsible for developing the software used could have also benefited from other methods after it was introduced, such as a production monitoring service and an internal control on certain transactions. (Heusser 2012) Having an automated process checking errors in addition to human monitoring could have helped minimize Knight Capital’s losses. Another method would be to flag certain high value transactions for approval in order for such transactions to continue. Utilizing not just one, but multiple of these testing methods could have saved Knight Capital from the devastating financial results.

What we are able to take from this catastrophic loss of millions of dollars is to make sure any program/algorithm can and are prone to failures. The instance a new software has been developed, the program must be checked thoroughly with all pre-existing software and protocols before released and executed. Extensive software testing in the beginning can avoid losses such as this. The loss from not spending enough time and money to test the algorithm is what lead to even greater loss in the end.

References

Gibbs, Mark. *"Knight Capital Fined a Measly $12M for a Software Bug that Cost $460M."* Network World, 23 October. 2013. Web. 07 Feb. 2016. <http://www.networkworld.com/article/2225633/software/knight-capital-fined-a-measly--12m-for-a-software-bug-that-cost--460m.html>.

Heusser, Matthew. *"Software Testing Lessons Learned From Knight Capital Fiasco."* CIO. N.p., 14 Aug. 2012. Web. 6 Feb. 2016. <http://www.cio.com/article/2393212/agile-development/software-testing-lessons-learned-from-knight-capital-fiasco.html>.

Mamudi, Sam. "*Getco posts Net Loss as Merger Partner Knight’s Trading Improves.*" Bloomberg Business. N.p., 7 Aug. 2013. Web. 7 Feb. 2016. <http://www.bloomberg.com/news/articles/2013-08-07/getco-posts-net-loss-as-merger-partner-knight-s-trading-improves>.

Naik, [Kshirasagar](http://www.wiley.com/WileyCDA/Section/id-302475.html?query=Kshirasagar+Naik), and [Priyadarshi Tripathy](http://www.wiley.com/WileyCDA/Section/id-302475.html?query=Priyadarshi+Tripathy). *Software Testing and Quality Assurance: Theory and Practice*. Hoboken: John Wiley & Sons, Inc., 2008. Print.

Phillips, Matthew. *"Knight Shows How to Lose $440 Million in 30 Minutes."* Bloomberg Business. N.p., 02 Aug. 2012. Web. 7 Feb. 2016. <http://www.bloomberg.com/bw/articles/2012-08-02/knight-shows-how-to-lose-440-million-in-30-minutes>.

Popper, Nathaniel. “*DealBook Knight Capital Says Trading Glitch Cost It 440 Million*.” The New York Times, 02 Apr. 2012. Web. 07 Feb. 2016. <http://dealbook.nytimes.com/2012/08/02/knight-capital-says-trading-mishap-cost-it-440-million/?\_r=1>.