Statement of Intent: I am studying Knight Capital Group's (KCG) software error in 2012 that cost the company $440 million as I want to better understand how the network surrounding KCG destabilized in order to help companies make sense of the role that human and non-human actors play in developing resilient large-scale software.

Using Actor-Network Theory to Examine Knight Capital Group's Downfall

STS Research Paper
Presented to the Faculty of the
School of Engineering and Applied Science
University of Virginia

By

Karan Chawla

February 29, 2020

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____

Approved: _____ Date _____
Benjamin J. Laugelli, Assistant Professor, Department of Engineering and Society

Introduction

Over a 45 minute period in 2012, Knight Capital Group LLC (KCG or Knight), a leading market making firm posted a net loss of over $460,000,000 - a few months later, KCG was bought out by a rival firm, Getco LLC. The collapse of one of the leading global market making firms came about from the inadvertent reactivation of a piece of dead test code known as *Powder Peg*, a mistake largely attributed to the engineers responsible for developing and maintaining the overarching software.

Many scholars argue that the engineers are at fault for a primarily technical oversight - emphasizing a lack in competency showcased by a failure to properly refactor code, lack of regression testing, and the reliance on manual production-level deployments. If we continue to believe that the engineers are solely responsible for the utter mishap that befell Knight, we are critically overlooking many other factors that played a role in the overarching and long-impending collapse of the unstable software.

Actor Network Theory (ANT) will serve as a framework that will showcase how instability in the maintenance of the network composed of a multitude of actors - not just engineers, led to KCG's downfall. Drawing on ANT, I argue that KCG's root failure lay in the power imbalance of the relationships between the actors involved in KCG's network. Specifically, factors such as the mismanagement of talent, time constraints, and improper risk management protocols alongside the technical lapse of the engineers developing the software will be examined to give insight into KCG's critical collapse.

Literature Review

A large amount of the research currently analyzing the downfall of KCG focuses solely on the technical error present within the software, passing the majority of the blame onto the

engineers internal to the company as opposed to looking at the organizational dysfunction that would inevitably lead to failure in one regard or the other.

In Jamie Lynch's *The Worst Computer Bugs in History: Losing $460m in 45 minutes*, a detailed timeline recounts how the technical faults of an outdated algorithm known as *Powder Peg* mistakenly deployed to a production server led to an unfortunate turn of events (Lynch, 2017). Powder Peg was an algorithm which, contrary to conventional wisdom, was meant to buy stocks at a high price and then sell them for a lower price - software meant only to be used in testing environments to ensure other portions of the software were functioning correctly. The primary point of failure is argued to be in the repurposing of a flag in deployment code that was previously used to enable the Power Peg code, leading the program to believe it was in a testing environment and activating problematic code. The second point of failure is discussed to be the inclusion of outdated and unused code - Power Peg was utilized in testing until 2005, yet remained in the codebase up to eight years later when it was inadvertently brought to life and led to disaster.

Lynch also highlights a report by the Security Exchange Commission (SEC) which details how no formal code review process nor proper deployment processes were in place to check if software had deployed properly. Overall, it appears that Lynch and others such as Sandeep Yada take the view that the engineers responsible for the writing and deployment of the code are solely at fault for the failure that ensued, while ignoring the dynamic relationships and network present within KCG that led to this long-impending lapse.

A similar analysis developed by Henrico Dolfing (Dolfing, 2019) discusses similar facets of prevention failure on the part of the engineers - namely the inclusion of 'dead code', lack of proper regression testing, and error in both manual deployment of production code as well as no

safety checks to ensure a single engineer did not err in action. Henrico does examine situations in which KCG did implement hazard implementation, however they were not applicable to the specific errors caused by the Power Peg code.

Specifically, KCG followed SEC regulations by creating a circuit breaker 'kill switch' that would be activated if the stock market experienced price magnitude fluctuations of greater than 10% during a five minute period - however the nature of the Powder Peg program was in executing a large quantity of trades that fluctuated at a small percentage (much less than the 10% catch-all regulation). The next fault Henrico documents lies within the lack of documentation for incident response with no clear path forward; the initial response taken by KCG was by the firms top IT people and resulted in incorrect action - it wasn't until engineers took a further look at the code that they realized the issue, shutting down the servers after catastrophic loss.

Hendric's approach to analyzing the KCG failure discusses facets of failure relating to both the prevention and aftermath of the disaster, mentioning parties with faults apart from just the engineers. However, where both Hendric, Lynch, and other scholars fail to provide critical insight is within the complex relationships and, more specifically, the power imbalance between engineers, top officials, and other actors which led to the collapse of KCG's network.

Conceptual Framework

My analysis of KCG's software error in 2012 draws on Actor-Network Theory (ANT), which is useful in the analysis of events in which the separation of human and non-human elements is difficult (Tatnall & Gilding, 1999). ANT was introduced by Bruno Latour in 1987 as an approach to social theory in which everything in our social and natural worlds exist in constantly changing networks of relationships. A main focus of the theory which separates it from other constructivist studies is the emphasis on inanimate entities and their effects on social

processes, these technological actors having just as much if not more agency than their human counterparts in society. ‘Actors’, then, may refer to any humans, things, ideas, and concepts (Cresswell & Worth & Sheikh, 2010) which constitute a heterogeneous network of technical and social relationships.

ANT places less of an emphasis on the given properties of the actors that compose a network, and rather looks at how these actors evolve and relate to one another as a result of their position and relationships within a network (relational ontology) (Greenhalgh & Stones, 2010). As networks are often dynamic and unstable, they must be aligned such that routines are set amongst conflicting entities (people, technologies, etc.) - a process known as ‘translation’. This stabilizing translation and thereafter network evolvement is achieved through four stages: problematization, interessement, enrolment, and mobilization (Hu, 2011). Problematization defines the problem and relevant actors, establishing a problem-solving network in which one’s own interests must be realigned to those of the controlling actor (Rivera & Cox, 2016). Interessement involves assigning actions to actors in order to resolve the problem set forth. Enrolment specifies the interrelation of the roles specified through interessement, often involving negotiations between actors. Finally, mobilization ensures that the actions of actors within a network both serve towards the common end-goal, as well as satisfying personal interests.

With relation to software, it is often hard to distinguish which parts may be a result of only technical instruction and those that come about due to human interactions. For example, schedules may *dictate* certain design decisions and a tight budget may *limit* certain options, placing emphasis on the idea that an actor “makes other elements depend upon itself” (Pollack & Costello & Sankaran, 2013), even if it is non-human in nature.

I propose ANT is a useful tool in examining the KCG software failure due to its innate focus on relationships within a network, leading to the notion that an action is not an independent choice, but rather the result of the influence of surrounding actors - both technical and social. Specifically, the failure of any individual person/party to predict or timely resolve the situation that arose due to faulty software is more a direct reflection of the systemic organizational issues rooted within the network that encompasses KCG. Overall, this systemic failure will be described through the stages of translation popularized in ANT, as I demonstrate how KCG's network was unable to maintain a stabilized state.

Analysis

Network Formation

The actions of solely the engineers responsible for KCG's stock trading software are not to blame for the loss of excess $400m - to demonstrate this, I will follow the translation pattern described in my overview of ANT. First, the motivation behind creating KCG should be highlighted as they qualify any actions subsequently taken as direct or indirect translations of the primary goal of being a market making firm for large brokerage-dealers and institutional investors. Furthermore, it is important to understand KCG's consumer base and business scope as engaging in automated trading at industry-high volumes (Tabbaa, 2019) brings to light the technical feat and organizational complexity required to operate the company.

In accordance with the ANT framework I set out to use, I will introduce the human actors of primary concern in KCG's heterogeneous network, grouped into relevant social groups pertaining to large-scale business operations as follows: (i) *technical staff* responsible for software development (e.g. software development team, deployment engineers, etc.); (ii) *supervisory staff* who are responsible for the management of KCG operations (e.g. project

managers, CEO, CIO, etc.); and (iii) *third-party influencers* who conduct business with and have relationships with KCG (e.g. NYSE, SEC, broker-dealers, etc.). With regards to the non-human actors in this network, the following actor groups are also defined, (iv) *risk management protocols* necessary for any modern organization; (v) *BNET* - an internal warning system; (vi) *SMARS software* that was the source of the faulty code; and (vi), the *timeline* for project delivery.

Additionally, within the first step of translation, *problematization,* it is important to highlight the controlling actor from which all other actors must realign their goals to be in accordance with, the then-CEO of KCG, Thomas Joyce. Upon taking over operations of KCG in May 2002, Joyce shifted the company's business model to high-volume market making and asset trading through new acquisitions. With this shift in focus, a new trading technology, *SMARS* (Smart Market Access Routing System) was developed in order to execute thousands of trades per second (O'Connell & Ciccotosto & Lange, 2014). As I have defined the supervisory staff, and more importantly, *Thomas Joyce* as the primary actor within KCG's network, it is important to understand the various associations and relationships between the actors pivotal to successful network formation and stabilization - stemming from Joyce's re-envisioned business plan.

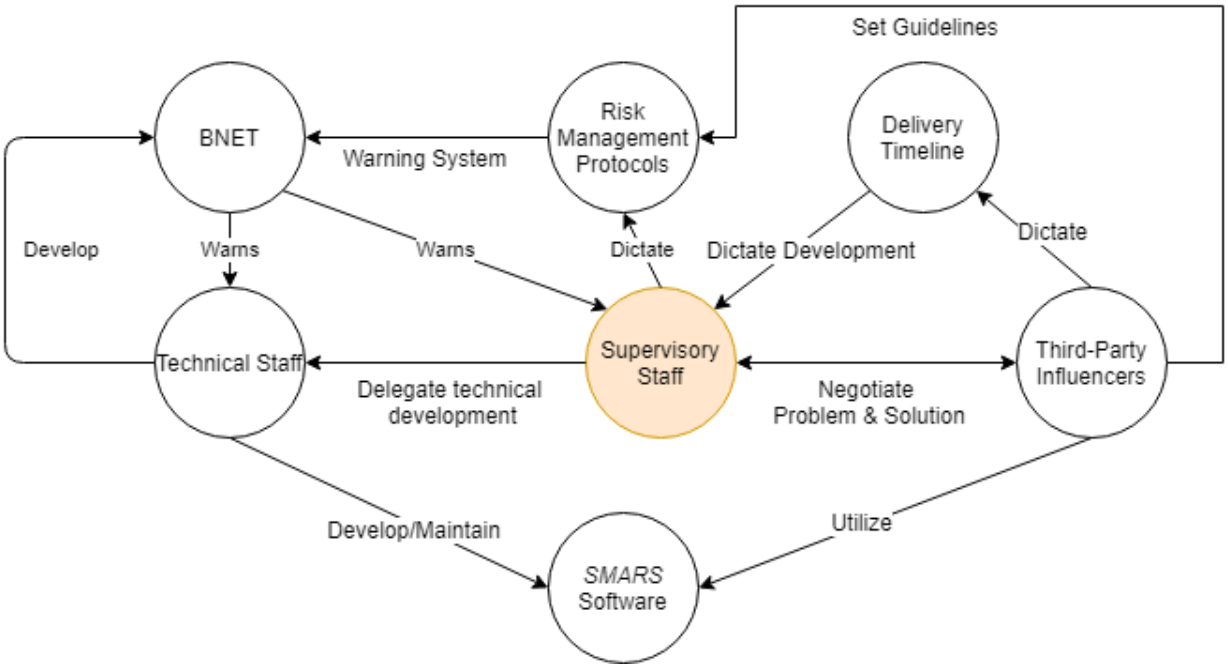

Figure 1. Knight Capital Group's Actor-Network Map

The general form of the network stems out from the supervisory staff as connections are directional and labelled with the relevant information that passes from one actor to another. As the supervisory staff are responsible for overall company vision and project execution, in accordance with the first step of translation, *problematization*, they must determine that shifting from human-based exchange trading in favor of computer execution will be a profitable business model. Based on this problem definition, the supervisory staff then identify technical staff required to create and maintain this software, as well as the third-party entities that will utilize and pay for this software.

During interessment, the supervisory staff actually recruit engineers who would like to take part in this high-demand and profitable venture, hiring software developers, DevOps personnel, engineers, and others that are necessary to develop and maintain the necessary technology. Additionally, customer acquisition takes place as relationships form between KCG

and online brokerages that would benefit from Knight's shift towards a combination of speed and accuracy in exchange trading such as TD Ameritrade, E*Trade, Vanguard, and others.

Additional parties such as the NYSE, NASDAQ, and SEC are introduced to the network as entities external to KCG's operations that house certain regulatory powers over Knight, dictating certain actions through the enactment of rules for market making firms. In an ideal scenario, we move on to the enrolment phase in which the roles of the human entities in our network are defined, leading to the introduction of the non-human actors into the network.

Assuming such a scenario; KCG engineers, as dictated by supervisory staff, would develop trading technologies that are in accordance with the rules set forth by third parties such as the SEC - to be utilized by those external to KCG, such as TD Ameritrade. Throughout this process, actors such as *BNET*, which is an automated warning system, and other risk management protocols are put into place by engineers and supervisory staff in order to maintain structure for long-lasting processes. Additionally, third parties may introduce timelines, budget constraints, and other items relevant to project delivery that must be managed and appropriately delegated by the supervisory staff aboard KCG to ensure relationships are maintained and business continues to grow. In this scenario, effective cooperation and communication between the human actors along with the implementation of rules and adherence to third-party guidelines in the form of the non-human actors results in the mutual success of KCG and institutions serving hundreds of millions of Americans such as the NYSE and NASDAQ. Seemingly so, the network constructed appeared to stabilize with a high degree of success - Knight became the largest trader of U.S. equities with a market share of 17.3% on NYSE and 16.9% on NASDAQ (Knight Capital Group - Liquidity Volume Statistics).

Network Destabilization

While on the surface Knight's operations appeared to be steady, streamlined, and effective, maintaining the status quo of operations was not enough to continue to grow business. In October of 2011, the NYSE proposed a private market of trading known as the Retail Liquidity Program (RLP) in which large institutions would be able to anonymously make market transactions for fractions of pennies less than what the public markets would be able to (Russolillo, 2012). Thomas Joyce, CEO of KCG, didn't give much credence to this proposal, being quoted saying "...I don't see how the SEC can possibly be OK with it (NYSE's Sub-Penny Quandary, 2012)." Much to the surprise of Joyce and others, this proposal was approved in June 2012, with plans to be rolled out by August 1st, 2012. With less than 30 days between the approval and live roll-out dates, KCG engineers were tasked with modifying the SMARS code - first developed and continually maintained up to 2012, with changes to its algorithmic routing taking place.

This action by the SEC and NYSE serves as a crucial point in investigating the power display between forces in KCG's network; the primary actor, KCG, is now beholden to third-party regulators. The bidirectional ties that arose over eight years prior that saw KCG become the single largest market-making firm have been severed and replaced with a dependency of KCG towards NYSE and others in order to maintain profitability and stay afloat. Although the CEO and supervisory staff were against the proposed changes they were ultimately forced to adapt to the situation placed upon them and that too with a turnaround time that far exceeds normal business practices (Tate, 2019). With just 30 days to implement changes to *SMARS*, the software developers were pushed to work quickly and effectively in order to meet their constrained timeline.

While it is understandable that on a day-to-day basis it is often the direct responsibility of software developers to ensure that any code written and deployed undergoes strict testing and adheres to previous tests created as well, the system-level lapse is more truthful to the situation at hand. Foremost, it is pivotal to understand that the "Powder Peg" program lay dormant for over eight years in the SMARS codebase - this lapse in judgement cannot be attributed to any single engineer who may change jobs year over year and rather falls on the supervisory team composed of project managers and DevOps leaders whose job priority it is to ensure the durability of any projects in their scope. Moreover, a *single* developer's failure to not check for a variable in code unused for over six years prior should not trigger a catastrophe on the scale that occurred.

Additionally, as highlighted in my discussion of ANT, it is important to understand that the power an actor has in a network is a function of its relationships within the network, not solely a measure of the actor's abilities prior to engaging with the network. This is imperative in understanding as it is foolish to believe that KCG habitually hired software developers that were technically incompetent and did not understand the importance of proper coding practices. Instead, it was the introduction of actors external to just the technical staff and the software they developed that brought upon limits on their capabilities to provide software at a pace that was also technically sound.

Moreover, the lack of adherence to sound coding practices and risk management protocols is a direct reflection of the power imbalance and unstable relationships surrounding both the supervisory staff and technical staff of KCG, as well between the third party regulators and the supervisory staff. To the later point, due to the timeline imposed upon KCG by the NYSE, the supervisory staff pushed for the quick development of the RLP code, leading to lapses in adequate testing for the new code. It should instead been the prerogative, and I believe

it was indeed the *obligation* of the supervisory staff of Knight to push back on the strict deadlines imposed by the NYSE - enacting major changes to an algorithm that daily oversaw billions of dollars worth of transactions is no simple task and it would be reckless to think it would be properly handled within 30 days.

Furthermore, when looking at the short-term cause of Knight's downfall, it is important to note that the computer glitch highlights Knight's overall long-term failures. What should have been commonplace adherence to risk management protocols put in place by the SEC and integrated into routine business practice at KCG was neither thoroughly developed nor checked upon throughout the years KCG operated. Specifically, redundancy checks such as conducting code reviews, regression testing, and automated deployments are industry-standard practices that would have all caught any mistake in code before it reached production; these practices are put into place so that any single engineer who may develop code should not have to needlessly comb through all other parts of the code to see in search for any unintended consequences. Additionally, although Knight had developed a risk monitoring tool, it required human monitoring at all times; Knight also failed to adhere to financial industry standards of having circuit breakers present to automatically kill all servers and suspend trades in the face of a catastrophe such as this (SEC, 2011). Overall, these lapses in Knight's internal operations showcase habitual failures in operations and subsequent network destabilization on multiple fronts.

Regarding the Engineers

With regards to the network as I have described, a counterpoint to the destabilization KCG faced may be that the transition away from human-focused technologies to automated processes are the root cause of systemic failure. While I understand that upon a cursory glance

one may derive that the lack of a direct human element in the execution of trades may have prevented this exact problem from occurring, it is not a realistic nor practical fix suited for the domain in which KCG operates. Namely, market making and high-volume trading, just on the scale of KCG, was billions of dollars a day worth of operations - the volume at which trades were managed and Knight's business grew would be upended if Knight stayed with human exchangers; computers are simply more efficient at delegating tasks that are: (i) repetitive in nature, and (ii) able to be broken down into a series of logical decisions. Additionally, the destabilizing dynamics within KCG's network did not arise due to the automation of trading - whether or not humans were involved in the actual trading process would not cover up the failure to adhere to best coding practices throughout development, nor the power imbalance between third party regulators and those internal to KCG's operations.

Conclusion

In this paper, I used the Actor-Network Theory framework to demonstrate both the short- and long-term lapses in operations that plagued KCG's network. With respect to the latter, the supervisory staff did not place proper emphasis on the risk management protocols the third party regulators called for - leaving themselves *and* the technical staff to be left with poorly managed software projects and risk mitigation techniques. With regards to the former, unrealistic timelines and demonstrably poor code writing and deployment techniques led to one of the greatest software collapses in modern history. It is my belief that through understanding how KCG's network destabilized, readers will better understand the importance and interplay of properly architecting large-scale software systems with the ever-shifting power dynamics at play within real business operations.

Word Count: 3,449

References

Cresswell, K. M., Worth, A., & Sheikh, A. (2010, November 1). Actor-Network Theory and its role in understanding the implementation of information technology developments in healthcare. Retrieved from https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/1472-6947-10-67

Dolfing, H. (2019, June 5). Case Study 4: The $440 Million Software Error at Knight Capital. Retrieved from https://www.henricodolfing.com/2019/06/project-failure-case-study-knight-capital.html

Greenhalgh, T., & Stones, R. (2010, May). Theorising big IT programmes in healthcare: strong structuration theory meets actor-network theory. Retrieved from https://www.ncbi.nlm.nih.gov/pubmed/20185218

Hu, D. (2011, February). Using actor-network theory to understand inter-organizational network aspects for strategic information systems planning. Retrieved from https://essay.utwente.nl/63005/1/master_thesis_s0211494_final_version.pdf

Knight Capital Group - Liquidity Volume Statistics. (n.d.). Retrieved from https://web.archive.org/web/20120609063400/http://www.knight.com/ourfirm/volumestats.asp

NYSE's Sub-Penny Quandary. (2012, January 2). Retrieved from https://www.tradersmagazine.com/departments/regulation/nyses-sub-penny-quandary/

O'Connell, B., Ciccotosto, S., & Lange, P. D. (2014, August 12). Understanding the application of Actor-Network Theory in ... Retrieved from https://researchonline.jcu.edu.au/34366/3/34366 O'Connell et al 2014.pdf

Pollack, J., Costello, K., & Sankaran, S. (2013, January 18). Applying Actor–Network Theory as a sensemaking framework for complex organisational change programs. Retrieved from https://www.sciencedirect.com/science/article/pii/S0263786312001834

Rivera, G., & Cox, A. M. (2016, August 1). An Actor-Network Theory perspective to study the non-adoption of a collaborative technology intended to support online community participation. Retrieved from http://eprints.whiterose.ac.uk/97837/3/WRRO_97837.pdf

Russolillo, S. (2012, August 2). What Exactly Is NYSE's Retail Liquidity Program? Retrieved from https://blogs.wsj.com/marketbeat/2012/08/02/what-exactly-is-nyses-retail-liquidity-program/

SEC to Publish for Public Comment Updated Market-Wide Circuit Breaker Proposals to Address Extraordinary Market Volatility; 2011-190; September 27, 2011. (2011, September 27). Retrieved from https://www.sec.gov/news/press/2011/2011-190.htm

Tabbaa, B. (2019, September 8). The Rise and Fall of Knight Capital - Buy High, Sell Low. Rinse and Repeat. Retrieved from https://medium.com/dataseries/the-rise-and-fall-of-knight-capital-buy-high-sell-low-rinse-and-repeat-ae17fae780f6

Tate, T. (2019, November 15). How Long Does It Take To Build Custom Software? Retrieved from https://soltech.net/how-long-does-it-take-to-build-custom-software/

Tatnall, A., & Gilding, A. (1999, January). Actor-Network Theory and Information Systems Research. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.1265&rep=rep1&type=pdf

The Worst Computer Bugs in History: Losing $460m in 45 minutes: Bugsnag Blog. (2017, September 14). Retrieved from https://www.bugsnag.com/blog/bug-day-460m-loss