Extrapolating Retention Intricacies of Software Professionals in Ugandan Companies

Maria Kaguhangire-Barifaijo
kaguhangire@gmail.com
Uganda Management Institute

IJOTM
ISSN 2518-8623

Volume 3. Issue II
p. 13, Dec 2018
http://jotm.utamu.ac.ug
email: ijotm@utamu.ac.ug

Abstract

This paper is part of the larger research that investigated the complexity of hiring and retaining Software professionals. This was because software professionals were found to habitually jump into and out of their jobs, which had become an industry-wide concern that caused instability and discontinuity. The Dual Factor of Motivation by Herzberg, 1954, was adopted to explain this challenge. A qualitative investigation augmented with the integrative synthesis was adopted. Subjects were selected purposively from selected companies such as banks and specifically Makerere University – being the dominant producer of software professionals in Uganda. Three questions were advanced to answer retention challenges; (1) What are the reasons software professionals do not keep their jobs for long? (2) What are the implications related to retention challenges in companies today? (3) What can companies do to retain such professionals? A combination of scarcity, particularity and peculiarity of these professionals made them arrogant and a desire to be freelancers, independent and "on call". Yet, most companies are not comfortable with this kind of arrangement, since it affects continuity, sustainability and teamwork. The author concludes that no matter the motivational strategies put in place, these professionals are hard to keep, yet, when they quit - they go with intangible knowledge, especially since they possess undocumented knowledge about the product, the domain, and the designs, essential to the company's competitive advantage. The paper therefore recommends that employers need to engage in honest negotiations and allow software professionals participate satisfactorily, as well as avoiding pigeonhole type of management.

Key words: *: Retention intricacies, Software Developers, Engineers and Professionals*

Introduction

This paper is part of the larger research that explored retention challenges of software professionals, which has become a nightmare not only to companies but to the human resource professionals as well. First, as a consultant who has been engaged in the recruitment and selection processes of software professionals, among other professions; and secondly, as a parent/guardian of software professionals, I have witnessed unprecedented high attrition rates of these professionals. Similar sentiments were shared by Sommerville (2010), whose concern moves beyond retention to career growth and profile development. In his view, the software profession was becoming fluid and exasperating because it was harder to find them. Consequently, retention challenges could put pressure on companies, especially in the process of procuring, acquiring, training and re-training software engineers (Bruegee and Dutoit, 2009). They argue that companies spend lots of funds on training new recruits, but lose them shortly after acquiring them to more competitive companies. Hence, this paper attempts to unravel dynamics in the profession.

The development of Software Engineering

Software engineering is a young field in Uganda, although it has existed for many years - but with different names. For example, programming languages started to appear in the 1950s (Pankaj, 2005; Pressman, 2009; and Sommerville, 2009, 2010), and this was also another major step in abstraction. In the 1960s, major languages were released to deal with scientific, algorithmic and business problems respectively, leading to the introduction of key concepts of modularity and information hiding in 1972. Prior to the mid-1960s, software practitioners called themselves computer programmers or software developers, regardless of their actual jobs (Abran, 2004; Laplante, 2007; Sommerville, 2007 and Matti, 2014). This development helped programmers deal with the ever increasing complexity of software systems (McConnell, 1999), making software engineering more sophisticated and yearned for - which could have brought about the indispensability of software professionals.

Remarkably, the origins of the term "*software engineering*" have been attributed to different sources, but it was used in 1968, where scholars (e.g. Matti, 2014; Pankaj, 2005; Pressman, 2009; and Sommerville, 2010), found that software engineering was created specifically to address poor quality of software, get projects time and budget under control, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and within specification. Consequently, engineering in its own right already addresses all these issues, hence the same principles used in engineering could apply to software, although software is more efficient and adept than the other branches of engineering (Pressman, 2009). In support of such efforts, Watts (1989), affirmed that the widespread of lack of best practices for software at the time was perceived as a software crisis - which was believed to have been minimized with software development endevours. Evidently, today software is considered the most malleable of media (Love, 2013). It is for this reason that software is perceived as a profession that probably leads to self-fulfillment, gives a sense of accomplishment with swift results, which demonstrates visible results and accountability on the part of the professional.

What is Software Engineering?

Software engineering is an elusive term and has been defined differently, by different scholars. For example, it has been defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (Sommerville, 2010); and as "a*n engineering discipline that is concerned with all aspects of software production (Pressman, 2009). On the other hand, it has been defined as* " the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines (McConnell, 1999). Subsequently, Software developers became the creative minds behind computer programs, where some develop the applications that allow people to do specific tasks on a computer or another device; others develop the underlying systems that run the devices or that control networks (Sommerville, op cit). Therefore, software developers have been called upon to analyze users' needs with prescribed design, they then test them and develop software to the satisfaction of the client. In writing software code programmers need clear specification of the designs of the client before they can identify the core functionality that users need from software programs which makes them peculiar (Pankaj, 2005; Pressman, 2009; and Sommerville, 2010 and Wiegers, 2005).

Consequently, the software developers are expected to maintain the software and continue testing and document every aspect of an application or a system as a reference for future maintenance and upgrade to create optimum software (Sommerville, 2010). Specializations of software engineers, in the industry may include; analysts, architects, developers, testers, technical supporters, middleware analysts or systems managers. Yet, in the academia, they include; educators, researchers and consultants (Behson, 2010; Bruegee & Dutoit, 2009 and Deepa & Stella, 2012).

Essentially, almost all critical activities in today's organizations require expertise of software engineers, and as Pankaj (2005) affirms, this indispensability renders companies powerless in retaining software professionals. Moreover, Pressman (2009) argues that because software developers are in charge of the entire development process for a software program, 'they brag and feel that sometimes they own the world'. In fact, software developers determine user requirements that are unstandardized to the functionality of software in different companies, such as; the level of security and performance needs. They design the program and then give instructions to programmers, who write computer code and test it. In his admission of the peculiarity of software engineers, Sommerville (2010) said; "..software engineers come up with just bits - which in actual sense could be thought of 'ignorable', yet, a software engineer can build castles out of thin air. Meaning, that entire businesses, industries and institutions, can be created with nothing physical at all! Hence, software's substrate is the stuff of pure thought... We all know that other engineering professionals are constrained by the surly bonds of the physical world. For example, to design a new plane, the aerospace engineer may spend years designing a model. Yet, a software engineer can go from idea to reality in a week, a day and sometimes hours. As an intellectual pursuit, software has been found to be enormously rewarding".

The Context and Problem Statement

From 2014 to-date, the author has been engaged on nearly a monthly basis to facilitate in the procurement and acquisition of software developers for various companies, among other professions. The companies include mobile Money hubs (FinTech), banks, small and medium consultancy firms, large scale enterprises, freight and air operations; and educational institutions. From a practical perspective, the author got concerned and kept asking "*why don't software developers or computer related professionals keep their jobs?"* Even with such startling cases, very few researchers, especially in Uganda have made attempts to explore causes of such enigmatic behavior of software engineers. It is only a few individuals outside Uganda, such as Behson, 2010; Bisht & Singh, 2012 & Wysocki, 2006; who have shared personal experiences using Wikipedia, but only mentioned how software developers were a rare breed, and possibly an explanation of some of the challenges faced by companies in their retention, although never gave details of the causes. Consequently, the author concurs with some scholars (e.g Behson, 2010; Bruegee & Dutoit, 2009 and Deepa & Stella, 2012), on the dangers of high attrition rates of software professionals, because it poses risks to the business or organizations, due to the human capital and talent; such as skills, training, and knowledge that get lost.

In Uganda, Software Engineering was introduced in 2009, as a teaching discipline, in the then Faculty of Computing in Makerere University. In East Africa, The University of Dar Es Salaam in Tanzania was already offering software engineering since the 1990s. Although some Ugandans, through the Exchange Programme of Inter-University Council of East Africa offered this programme in Tanzania, the profession becamue famous when Makerere had its first graduates on the market. This means that the first Software Engineering graduates from Uganda went into the market around August, 2013, to join a group from University of Dar Es Salaam and elsewhere. Although, research (e.g by Baryamureeba, 2010; Pressman, 2009; and Sommerville, 2010) revealed that retention of these professionals was a mayhem, it was not widely documented because of the small numbers of these professionals. Almost all transactions globally, do not require physical presence of the service providers. These include but not limited to; advertisements, financial transactions, marketing, business venture, educational programmes are all made possible with the competences of software professionals. In this modern day and time, payments are done electronically, including deposits and withdrawals. Business advertisements, reminders and invitations require expertise of software professionals by use of "just" a code. In educational institutions software professionals are hired to perform various functions - that include but not limited to; advertisements, applications, admissions, time-tabling and examination processing. Each of these software applications cost a hefty amount of

money, where even the professionals acquire superior competences, given that every company wants to be unique. These professionals acquire plentiful familiarity of the different software from different companies. Hence, by implication, when these professionals quit the jobs, they depart with all this information - notwithstanding loss of stability, continuity, sustainability and costly hiring activities. This means therefore that acquiring a new developer requires fresh and rigorous attraction and procurement activities, fresh negotiations and fresh training. Although Bisht and Singh (2012) found that turnover was normal, exaggerated turnover has the potential to hurt the company - especially when the company loses highly skilled and "*hard-to-get"* professionals. This is because, when excellent workers quit – they take the company knowledge because they gradually become the "think-tanks" of the company.

To be able to understand the peculiarities and particularities of Software professionals, and why it is difficult to retain them, the following questions were posed:

1. Why is it difficulty to retain software professionals in Ugandan Companies?
2. What are the implications of failing to retain software professionals in Ugandan companies?
3. How have companies attempted to resolve retention challenges of these professionals?

Literature Review

Literature on employee retention has revealed that human resource professionals' world over were faced with retention challenges of all types of employees for various reasons, regardless (Maya and Thamilselvan, 2012), although, in the case of software engineers it was a different case because they job-trotted, and actually, opportunities were quite open for them (McKay, Avery, Tonidandel, Morris, Hernandez and Hebl, 2007). They found specifically retention of software engineers to have more challenges than any other profession in the recent past. Research (e.g. by Behson, 2010; Bruegee & Dutoit, 2009 and Deepa & Stella, 2012), found the profession to be to be enigmatic and complex as well. To comprehend issues of retention, the Dual Factor theory of motivation, postulated by Frederick Herzberg (1954) was adopted. The theory assumes that employee satisfaction is derived from one major component (satisfiers), although the second one is healthy to keep employees on their jobs (Hygiene). Consequently, the theory breaks job satisfaction into two factors; "*hygiene factors*"; such as working conditions, quality of supervision, salary, safety, and company policies, that keep the employees "*stay on the job*" (George & Neerakkal, 2011). Conversely, "*satisfaction factors*" such as achievement, recognition, responsibility, challenging tasks, the work itself, personal growth, and advancement; were perceived to motivate employees (Bisht & Singh, 2012) and stay longer on the job. Judge and Klinger (2007) concur with the propositions of the theory, and argue that whereas, hygiene factors were necessary to ensure employees don't become dissatisfied, the very attributes do not contribute to employee motivation, either. They argue that an employees need for meaningful and personal growth is what keeps people stay on their jobs, yet those satisfiers are intrinsically imbedded and at the same time, incognito. Hence, according to this theory, employees view the job as a little more than a paycheck, much like how software professionals view salary or pay.

In the present-day, where technology has rocked business endeavors, Laplante (2007) contend that software developers are the 'thing' that can work miracles, since every organization is looking for tailor made, cost effective, high quality, scalable customized solutions to function speedily and provide quick solutions for their needs. Astrauskaite, Vaitkevicius & Perminas; 2011 and Mahoney, 2015; advance that one critical reason why software professionals have become more important, is their ability to provide a solution which is as per the company's exact requirement and functionalities customized to a specific company. Considering their creativity, ability to do research, design, program, and test computer software, it makes these professionals distinct and superior (Merx & Norman, 2006; Pressman, 2009). For example, these professionals argue that: "*software professionals code new features and fix bugs and, in a click of a finger, the*

deal is done! These professionals, don't write specs, they don't write automated test cases, they don't help keep the automated build system up to date, they don't help customers work out tough problems, they don't help write documentation, they don't help with testing and they don't even read code. All they do is write new code and get ready with a product - and that makes them distinct".

However, the distinctness may not explain why these professionals never keep their jobs for long. Thirulogasundaram & Senthil Kumar (2012) these professionals to have a complicated relationship with money, because they keep saying they don't care about money – but, that is part of how they see themselves in relationship to their jobs. Laplante (2007) found that the real reason software developers say they don't care about money could be they make so much of it, that they reach a point where money does not matter. This means therefore that companies must resolve the salary issue off the table and be toast. Aside pay, Fethi, et al..2011; Deepa & Stella, 2012; blamed these professional's arrogance on employer's preference, different pay and special allowances by some companies. Yet, once the company pays these professionals below market rates they quit and look for better offers. One other reason software engineers do not retain their jobs is because they do not want to be directed on what to do which according to Pressman (2009) could outweigh all the good. This is because skilled software professionals understand that crossing the value apex often triggers an innate feeling of "time to move on" and after a while, leads towards inevitable resentment and an overall dislike of the job (Pankaj, 2005). Consequently, they always expect consistency, and timely payment independence, complex and challenging tasks, and ability to contribute to what is being done in the company. Therefore, Bruegee & Dutoit (2009) advances that in order to retain your professionals, you must make sure that whatever you commit to, is feasible and that you follow through with the promise.

In this mix of exacerbated attrition, Bruegee and Dutoit (2009) explains how companies have had consequences related to cost, lost company knowledge and reduced morale due to departing colleagues. There have been also contingent facets regarding lost opportunities on the part of the employers (Fethi, et al..2011). It gets even worse with software professionals because of some sophistication-related. Nonetheless, retaining skilled professionals with the competencies to software functions has continued to be a challenge not only to human resource professionals and companies, but also to institutions who give them reference for offers (Morrell, Loan-Clarke, Arnold & Wilkinson, 2008).

Not until there is sufficient pool of these professionals to compete for the jobs and companies devise retention strategies this mayhem will endure Lewis, (2013). Studies by (e.g Podsakoff, LePine & LePine, 2007) have also found that turnover of software professionals negatively affects competitive advantage, profitability, continuity and productivity of business companies. This has been found to harm not only competitive advantage of companies, but also stability. Whether obsession or arrogance, human resource management (HRM) professionals need to step up and address this situation because these professionals are the think-tanks of the companies.

Methodology

A case study, using qualitative investigation was employed. This was augmented with an integrative synthesis were adopted. This approach is widely acceptable as an evidence-based method to summarize the existing research literature, observed behaviour, personal experience and artifacts (Kothari, 2006). An integrative synthesis was used to collect and compare evidence that involved various data collection methods. Kothari (2006) recommends this approach and argues that it is most suitable for investigating patterns across primary and secondary research studies, compensating for single-study weaknesses in research design to improve the internal and external validity of the various research findings. A non-probability sampling technique was adopted. Document analyses, literature search and observations

were espoused. These were supplemented with the in-depth interviews with employers and software professionals. Makerere University, selected banks and companies were selected for this investigation. These software developers included those in formal employment, the freelancers, and the consultants that own companies. Both thematic, content and narrative analyses were employed to generate answers for the research questions. According to Borg, (1994), triangulation of these analytical techniques were suitable for primary, secondary data and storytelling that emerges from experience and observation. Nevertheless, by its nature, qualitative research has an element of subjectivity. Hence, although the author may not have any control to completely eliminate inherent human bias, she made an effort to exercise restraints in order to produce credible results. First and foremost, she moved out with intellectual humility, suppressed her personal beliefs and knowledge and was bound by the set questions. Further, responses were well documented as reported. All respondents were treated with dignity.

Findings and Discussion

This research found that software engineering is a relatively new field of study, where we have not had many on the market. Actually, Matti (2014) also found that even the few that are on the market prefer to work as independent consultants, contractors or work for themselves as freelancers, which in itself poses serious challenges for employers. Therefore, the problem may not only be retention, but also finding them in the first place. Hence, it should be noted that these professionals perform different tasks in the software development process (Pankaj, 2005). Interestingly, software professionals deny being motivated by money and instead claim enjoying coding, although they admit that the profession truly pays well and jobs are easy to find - which in itself is the cherry on top (Behson, 2010).

The first question sought to understand why it was difficult to retain software professionals and the findings were quite controversial and conflicting – ranging from arrogance, obsession, intellectual curiosity, attractive pay; and the peculiar nature of the profession. Whereas the findings revealed high demand for these professionals, Ramakrishnan, et al.. (2013) found a near infinite amount of stuff to learn - every minute, every hour, every day, every month - and all year through. He asserted that because these professionals love to learn and do new things, it keeps them alert and motivated to work. In support of this finding it was also revealed that because of the new languages used, new programming paradigms, learning hardware stuff and their characteristics, science and theory, they never get bored. Because of their creativity and passion, software engineers won't do their best work or stick around in a hot job market if they're being micromanaged. Although this was confirmed, there was much more than being creative, and that was the need for independence and freedom and also to choose their own approaches to solving problems, measuring them on their results rather than how they achieved them.

On the same finding, Rayl (2008) explains how the benefits of allowing developers to explore their ideas will not only attract and retain super employees, but will definitely make an impact on the productivity of the company. Findings further revealed that software professionals desire to take time off to accomplish their goals privately – they need time alone, they need concentration and they need to critically think. Yet, the changing technology has actually exacerbated the situation of developers, they always go for the latest version of anything (Merx & Norman, 2006). They were found to have interest in the latest and greatest tools to play with, is both fun and interesting which gives them bragging rights, with their peers, but also kept them current in a fast-changing technical world. This was affirmed by numerous researches (e.g Behson, 2010; Bruegee & Dutoit, 2009; Deepa & Stella, 2012) on how the latest tools and processes may have higher and immediate retention effects. Although Pressman, 2009; and Sommerville (2010) found that retention was dependent on many factors, companies should be cautious of new software "*addictive"* since it is believed to bring instant gratification of seeing their work in production.

Interestingly, efficiency was what was found to be critical for their motivation. It was found that what excites them was taking the shortest time possible to complete a task or a project. They become fulfilled by accomplishing a task in just weeks, days or even hours to fix the deal and were found to be "go-getters"! They were found to enjoy unrealistic deadlines since they are perceived as a huge part of being set up to succeed. Abran, et al.(2004) also found this interesting work mode, because they want to build software that not only works, but maintainable and praised which gives them a sense of accomplishment. Short of this, these professionals expressed their displeasure and underutilized. Another interesting finding was their need to deliver a specific project on-time, and wanting to do things the right way, and not just the quick way because the considered quality to be an important feature count and budget. Bisht & Singh (2012) are in support of this finding and affirmed how interruption of their work can cost the company their critical professionals.

Although, many professionals denied they quit jobs because of pay, it was discovered they demanded a hefty sum of money as salary. This finding was supported by Kalwarski, Mosher; Paskin & Rosato (2006) who contend that software engineering was one of the highest paid profession. In India, for example, this profession, much as it is highly cherished, retaining them has remained one of the challenges facing companies today (Sivarethinamohan & Aranganathan, 2014). In fact, these professionals charge exorbitant salaries, and as soon as you bring them on board, competitors court them with possibly with sweeter deals. Similarly, Maya & Thamilselvan, 2012; confirmed this finding and affirmed that it wasn't easy to keep these top talents from walking out the door. They doubted they were motivated by just money. But what is it that they want in a job? When asked about whether money was the driving factor, one respondent had this to say:

> To me, money doesn't matter – but of course, that is the reason I went to school. I want a flexible work environment, I want to do things my own way. I want to make decisions in whatever I do. No one can compromise my culture by trying to account for people that might take advantage of the situation. I want to be creative, to be left alone, to think, to meditate, to create a code and fix stuff! When I do my work and where, is up to me. Even if I was paid an "eight-digit salary" and I feel I am stepped on, I will definitely quit! Money does not count, I want my freedom – I want my independence, I love creativity, I love doing things my own way – period.

Although Kalwarski, et al. (2006) found that salary wasn't prime for most developers, it was evidently inextricably linked to happiness and satisfaction as explained by Herzberg (1954). Therefore, by implication, although salary may not be their biggest concern, companies should not take months without paying these professionals. So long as there is an opportunity to work on an important project which is a little complex and a hope of success (Victor Vroom, 1965), a believable vision to achieve it, trust the software professional, he/she will stay with a company not just for pay but for the feeling of accomplishment. Rittinghouse (2003) finding was in support of the current one. Correspondingly, he found constructive and timely feedback to be as critical as the pay cheque itself. Hence, it is imperative that companies provide timely and accurate feedback if they want to retain these professionals. Although it might be hard to believe, many software professionals money or salary was secondary. What they value most is meritocracy. Similarly, Kalwarski, et al..(2006) found that these developers command high salaries. For example, although scholars such as; Furnham, et al. 2009; Judge & Klinger, 2007; Maya & Thamilselvan, 2012; found that money was not a smart strategy to keep software developers on the job, Pressman, (2009); and Sommerville, (2010) found that at least every employee worked for money - to begin with, and should be down on the list of priorities- and must be competitive in order to keep these professionals. One question remains, what is it that makes some companies attract and retain developers while others churn through them like toilet paper? A startling finding was that actually it is not true about

working or not working for money. During the interviews, it was revealed that actually money was key in the employment contract and for that matter, it was a "constant". It was found that aside attractive pay, which was obvious anyway, the rest were critical to make them stay. The high cost of these professionals was due to their willingness to take risks.

George & Neerakkal (2011) reiterate that software engineers actually do not become engineers overnight, and when they eventually prove themselves, they do not for sure, work for money. These enigmatic people, do what they do because the profession touched on a deep inner yearning to tinker, do their best, and impress their friends – in the same profession. Although software engineers claim that they are not motivated by money, majority of workers surely are (Deepa & Stella, 2012).

Surprisingly, unlike other professions, these professionals loved challenging work, they admit finding no time to eat or sleep if they have not fixed the code, yet with no extra pay because they value recognition not only from their clients (employers) but also from their friends within the same profession (Paskin & Rosato, 2006). They advance that even before accepting to sign any contract, they make their demands, and because companies need them, they submit to their demands, only to lose them after a few months. Disappointingly, they still leave, often times, without any notice. Frequently they demand for new technologies, new hardware, want to operate from their homes. Another key informant lamented:

> Last year (2015), we hired two fresh software graduates and sent them out for a six months training in Dubai. Only to come back and tender in their resignation two months after they returned. Surprisingly, they even offered to pay back funds that had been expended on them for the training. It was not little money but they offered to repay. Some competitors are out there, waiting for finished products. It is a real challenge retaining these professionals.

Surprisingly, whereas those who get into the jobs voluntarily get out of them as fast as possible, others with similar qualifications, have failed to get any employment. Part of this has been found to be lack of opportunity to at least, do something new, or challenging task that requires them to scratch their heads and think deeper (Deepa & Stella, 2012). One employer had this explanation.

> It is about motivation, interest and the will! Even as we train them, we realize that some of them could have ended in this profession – either to fulfill their parents' desires, or fate landed them there. Sometimes we find that the extremely brilliant students are passing because they have high IQ, but not necessarily passionate about the profession. Passion is critical and will definitely determine individual's opportunities.

One respondent equated these professionals to footballers:

> There are always so many players, but only a few can be bought out and paid highly for their superior skills, these professionals too will move so long as they have something peculiar to offer, where the receiver too values that peculiar skills. For example, player can play for Arsenal, Man-U, Chelsea, etc.

In support of such analogue, Bisht & Singh (2012) argue that many times, like football players, these professionals lack serious commitment. Not in a sense that they can be manipulated during play time, but can be compromised thereafter.

Surprisingly, these professionals have a social contract with colleagues at work, not just a financial contract with the organizations they work for (Maya & Thamilselvan, 2012), which has an effect on their decision to stay, otherwise the social and relationship aspects of the job may not be enough reason to keep these professionals in the job. Hence, Matti (2014) advises managers to keep good relationship with key

developers in order to retain them. Nonetheless, maintaining challenging and enriching work for these professionals is not easy.

Consequently, it is clear that the two aspects – good money and the peculiarity of the 'coding thing' might be the trick to retain them. I found some bit of pomp, arrogance, obsession and intellectual curiosity while interacting with these professionals. However, what was clear is that software professionals believe they can move mountains. Mahoney (2015) too, found that great engineers don't consider any task to be "beneath" them, since they are always willing to lend a hand where possible. He argues that great engineers help out novice engineers in between doing their own work. If something has to get done, and no one else is able to do it in time, great engineers volunteer to take on the work. About their willingness to help out, one respondent had this to say:

> We do not scoff when asked to help on a project – be it small or menial or even low-profile. We are team-focused and willing to do whatever it takes to help the team. You need to understand that great aren't born, they are made. You need patience excel - to turn around the world – it is not a one stand job. You don't become a great developer over night. How do you think employers identify us? You must prove yourself through your work and how you work in teams. Self-discipline and self-sacrifice are key towards becoming a great software engineer.

Apparently, these professionals have the power to take risks, push boundaries and generate codes that transform operations and enlighten how millions of consumers and small businesses manage their finances in cloud, platform, mobile and SaaS environments (Pankaj, 2005; Pressman, 2009; and Sommerville, 2009; 2010). This prowess was found to make them extremely competitive or indispensable, and could be the driving force for such yearned independence and becoming freelancers. Quite infuriating, though, companies found it extremely difficult to get the right professionals with the requisite skills. Yet those with right attitude, with the will to learn and lacked the competencies required by these companies. This finding was supported by Maya & Thamilselvan (2012). They affirmed that actually failure to find software developers with the right skills, the right attitude, able and willing to work, has left many employers in disarray.

Interestingly though, these professionals always want to know what exactly the client wants and then be left alone to do everything. Further, no amount of pressure can push them into a specific direction, because if you insist on what they do not like, you will drive them away. Hence, software developers prefer being left alone, to concentrate and think, to generate ideas and develop something new and of his/her interest (Zachariah & Roopa, 2012), and if they are to work in teams, then, it must be really a "good team" with comparable brains and moving at the same pace - that's what software professionals consider good teams, as they cannot stand slow learners. Yes, those are the software developers. If they prefer to work this way, it's imperative to keep them together and let them be. If this arrangement is intercepted, they prefer to leave to where they can find freedom and perfect match.

Further, these professionals were found to be in the trenches, and would always be the first ones to know when a system or process is not working. They demand high degree of self-assessment to enable them evaluate their contribution, otherwise, they will get bored and eventually quit (Matti, 2014). They want someone to listen to their problems and actually take them seriously. When a developer speaks, they want someone to listen to them. It cannot be coincidental, when all of them say the same thing, someone should listen and act quickly because ignoring them would be frustrating their efforts and would actually lead to their quitting.

Quite distinct from other disciplines, software professionals have been found to enjoy writing and playing around with their codes and indeed, writing codes was the most precise way for them to express their thought process and creativity (Maya & Thamilselvan, 2012). Actually, they often consider their work as a craft since they can spend their whole lives improving their skills in them. Seemingly, the craft in question has a community of software engineers/developers which is much closely knit, unlike most professions. Suhasini & Naresh-Babu (2013) confirmed their elitist of always organizing meet-ups for every niche of programming, which enable them easily pull fame by making various contributions about their profession and their work. They found that these meet-ups helped them sustain interest in programming and developing new software, while constantly learning new things.

Interestingly, these professionals were found to be more loyal to their career and personal development than the company themselves. This was evidenced by their continuous search for better ways to fix a task and value being crafting a code from mere creativity. Therefore, Pressman (2009), advises employers to give these professionals their space and 'let them be'. Nevertheless, Zachariah & Roopa (2012) doubts whether even with such freedom these professionals can stay if they have found something more interesting to do out there. Consequently, Rittinghouse (2003) concludes, that the stronger the match between the job requirements and the employee's skills, goals and values, the more likely it is for the companies to keep these professionals.

Given their numerous demands, companies have had to grapple with this challenge. For example, projects have stalled, institutions have been sued and money lost after the departure of these professionals. One of the obnoxious scenario was when so many students one institution in Uganda, missed graduation because of the failed system during the process of compiling their results, which took the institutions another two months to recruit a new software expert.

In an attempt to explain such circumstances, Judge & Klinger, 2007 said it was extremely difficult if not impossible, to completely protect a company from competing with other employers. However, Bisht & Singh (2012), found that companies can employ proactive retention strategies so these 'high-demand' professionals are less inclined to entertain offers from other companies. In fact, it is worth investing in because anytime, these professionals will seek employment elsewhere, even for minor extra advantages; be it compensation, advancement opportunities or freedom to be creative. In fact, Deepa & Stella (2012) found that these professionals cannot be stopped from leaving the company if they wanted to and explained how companies have come up with numerous and exciting strategies to keep software professionals forgetting how these professionals have different motivation from other employees, which makes it complex to figure out what exactly they want.

Similarly, Maya & Thamilselvan (2012, found that the best trick was to keep these professionals focus on their interesting projects and letting them continue to develop their skill set, and that to avert high attrition rates, employers should negotiate at the time of entry. Secondly, employers should mind what they say and how they say what they say, although, some of these professionals will make decisions based on company culture and business infrastructure right from day one, which companies may not change. On this, Suhasini & Naresh (2013) recommend work-life-balance programs where these professionals are involved. The advance that in order to keep these professionals they should participate in any health program intended for them, otherwise you may not solve any problem. Much like Herzberg's assumption, recognizing good performance was found to be one great strategy (Thirulogasundaram & Senthil-Kumar, 2012). They advise regular assessment of motivational levels of these professionals and ensuring regular feedback from employees, which might uncover aspects that are of concern and can be corrected before they become big problems.

Similarly, your company's commitment to the environment, friendly work environment, the community and innovation is very important to make these professionals love what they do, although it may not guarantee their stay (Zachariah & Roopa, 2012). Additionally, if you want to succeed in retaining your key players, ensure that your company has a social and environmental conscience and should allow these developers to be innovative, add something new and have a voice. Strangely though, there are other tangible aspects that these professionals value other than the work environment to make them stay.

Conclusion and Recommendations

First and foremost, software - an obsession and addiction, and when they cross the value line, it triggers an innate feeling of "*time-to-move-on*" that leads to inevitable resentment – they want to continue discovering. Consequently, there is no fixed recipe for success - retention will be an ongoing challenge throughout the life of the company. Hence, employers should embrace up-to-date technology, give them space, challenging tasks, allow feedback – Act, and keep their work interesting.

Secondly, competition and rivalry of companies striving to out-perform their competitors have instigated the "*get-up-and-go*", syndrome - where "the means always justifies the goal". Hence, the differential treatment and exorbitant allowances, have made these professionals very indispensable, making retention a nightmare for employers. Therefore, employers should not use tricks or try too hard, otherwise what goes around – comes around.

Thirdly, structural deficit has led to so much dependency, leading to lack of succession planning, and empowerment have. This indispensability has made them enjoy total monopoly. Employers should therefore; task a motivated team to work with these professionals, embrace a sense of trust, develop and maintain team cohesion – have a succession plan and introduce mentorship programs- to avoid losses. Further, they should sign up for intellectual property rights, so that when they quit, they leave behind company ideas, initiatives and software. Similarly, usage of "out-of-range" or "task-related" reward systems – diminishes enthusiasm and group cohesion and motivation. Recognizing one category of staff has generated dysfunctional conflicts, complete dismay, havoc and disharmony among staff, as explained by Adam Stancy' (1965), Equity Theory. Employers should negotiate terms that do not contravene company policy – be very clear. Give them short contracts – mitigate culminating law suits - assign the maintenance work for less problematic professionals

Finally, it should be noted that software development is a tedious business and a difficult task for all involved. As projects near completion, work life is full of meetings, disorganization, overtime, and stress. For a developer, even the standard day is full of heavy thinking, problem solving and frustrating compilation errors. The issue of burnout is a real issue and something that faces nearly every developer at some time in their career and forces them to look elsewhere for a new job thinking it could be better the other side. Hence, the developers' employment is a constant journey that's going somewhere, rather than a Sisyphean situation where they're running out the clock until retirement. Therefore, it can be helpful to identify good developers and find out where they want to be and what they want to be doing, working with them to make them happy and to keep the company moving forward. To curb burnout, develop work-life-balance programs and allow these professionals to have fun, because, a team that has fun together is much more likely to work together well, enjoy working with each other and help each other to lighten the mental drain of a tough day of debugging.

References

Astrauskaite, M., Vaitkevicius, R. & Perminas, A. (2011). Job Satisfaction Survey: A confirmatory factor analysis based on secondary school teachers' sample. *International Journal of Business and Management*, 6(5) pp. 41.

Baryamureeba, V. (2007). ICT as an Engine for Uganda's Economic Growth: The Role of and Opportunities for Makerere University. *International Journal of Computing and ICT Research*, Vol. 1, No. 1, pp. 47 - 57. Retrieved from http:www.ijcir.org/volume1-number1/article6.pdf.

Behson, S.J. (2010). Using relative weights to re-analyse 'settled' areas of organizational behaviour research: The job characteristics model and organizational justice. *International Journal of Management and Information Systems*, 14(5), pp. 43.

Bertrand, M. (2014). The origin of software engineering. In Tedre, M. (2014) *The Science of Computing*. Routledge

Bisht Nidhi S. & Singh. L.K. (2012) "Understanding Antecedents to Attrition for Employees with Varying Levels of Experience in Indian Software Industry Professionals. *Global Business and Management Research: An International Journal*, Vol. 4, No. 1,

Bruegee, B., and Dutoit, A. (2009). *Object-oriented software engineering: using UML, patterns, and Java* (3rd ed.). Prentice Hall.

Deepa, E. & Stella, M. (2012). Employee Turnover in IT Industry with Special Reference to Chennai City-An Exploratory Study", *ZENITH International Journal of Multidisciplinary Research* 2(7)

Ewen, R.B., (1964). Some determinants of job satisfaction: A study of the generalisability of Herzberg's theory', *Journal of Applied Psychology*, 48, 1964, pp. 161.

Fethi, C., Cigdem, A. Gumussoy, Ibrahim Iskin, (2011). Factors affecting intention to quit amongIT professionals in Turkey", *Personnel Review,* Vol. 40 Iss: 4, pp.514 – 533

George A. P. & Neerakkal, J. A. (2011). Turnover Intentions: Perspectives of IT Professionals in Kerala", *IUP Journal of Organizational Behavior*, Vol. 10, No. 1, pp. 18-41.

Hardiman, N. (2014). *A portrait of the modern cloud developer*. Retrieved from https://www.techrepublic.com/article/a-portrait-of-the-modern-cloud-developer/

Judge, T.A. & Klinger, R. (2007). Job Satisfaction: Subjective well-being at work. In M. Eid & R. Larsen (Eds.), *The science of subjective wellbeing* (pp. 393-413). New York: Guildford Publication

Furnham, A., Eracleous, A. & Chamorro-Premuzic, T. (2009). Personality, motivation and job satisfaction: Hertzberg meets the Big Five. *Journal of Managerial Psychology*, 24(8), pp. 765.

Kannan, M. and Vivekanandan, K. (2012). A Study on Attrition among New Entrants in Software Testing Professionals. *International Journal of Computer Applications,* 53(7):23-29.

Kothari C.R. (2006). *Research Methodology*. New Delhi: Vishwa Prakashan.

Kuruuzum A, Cetin E.I & Irmak S (2009). Path analysis of organizational Commitment, job involvement and job satisfaction in the Turkish hospitality industry. *Tourism Review*, 64(1).

Laplante, Phillip (2007). *What every engineer should know about Software Engineering*. Boca Raton: CRC.

Maya. M and Thamilselvan, R. (2012). Employee Perception towards Talent Management Strategies - An Empirical Study with Reference To Software Companies in Chennai City. *International Journal of Management (IJM)*, 3(2), pp. 171 - 176.

Mahoney, M. (2015). The Roots of Software Engineering. *CWI Quarterly* 3 (4): 325–334.

McKay, P.F., Avery, D.R., Tonidandel, S., Morris, M.A., Hernandez, M., & Hebl, M.R. (2007). Racial differences in employee retention: Are diversity climate perceptions the key? *Personnel Psychology,* 60,35–62.

Morrell, K., Loan-Clarke, J., Arnold, J., & Wilkinson, A. (2008). Mapping the decision to quit: A refinement and test of the unfolding model of voluntary turnover. Applied Psychology: *An International Review*, 57,128–150.

Pressman R.S. (2009). *Software Engineering: A Practitioner's Approach* (7th ed.). Boston, Mass: McGraw-Hill.

Ramakrishnan, R., Vijayakumar, B., Shesha, A. and Shaji, J. (2013). Use of Structural Equation Modeling to Empirically Study the Turnover Intentions of Information Technology Professionals in Pune City". *Indian Journal of Science and Technology*, Vol 6(12), 5612-5624.

Rayl, A.J.S. (2008). *NASA Engineers and Scientists-Transforming Dreams Into Reality*. Retrieved from https://www.nasa.gov/50th/50th_magazine/scientists.html

Sivarethinamohan, R. and Aranganathan, P. (2014). An Empirical Study on attrition intention among software professionals in Indian Information Technology organizations. *Asian Journal of Research in Business Economics and Management*, Vol 4(7), pp. 29-36.

Sommerville, I. (2008). *Software Engineering* (7 ed.). Pearson Education.

Sommerville, I. (2010). *Software Engineering* (9th ed.). Harlow, England: Pearson Education.

Suhasini,N. and Naresh Babu, T. (2013). Retention Management: a strategic dimension of Indian IT companies. *International Journal of Management and Social Sciences Research (IJMSSR),* Vol. 2, No. 2.

Tedre, Matti (2014). *Science of Computing: Shaping a Discipline*. CRC Press.

Thirulogasundaram, V.P & Senthil Kumar, S.A (2012). Assessment of Individual and Propel Intention for Job Attrition on Software Industry: Voice from Software Employees In Bangalore city, India", *European Journal of Businessand Management*, Vol 4, No 6, p.48-56.

Wright, T.A., & Bonnett, D.G. (2007). Job satisfaction and psychological well-being as non- additive predictors of workplace turnover. *Journal of Management*, 33,141–160.

Zachariah, M. & Roopa T.N (2012). A study on employee retention factors influencing IT Professionals of Indian IT Companies and multinational companies in India. *Interdisciplinary Journal of Contemporary Research In Business.*