Name: Mark Jomar B. Buo	**Date:** December 12, 2016
Course/Section: CPE506/CPE51FC1	**Instructor:** Engr. Jonathan Taylar
CASE STUDY (PRELIM) – ETHICAL ISSUES CONCERNING ETHICAL DILEMMAS	

Ethical issue 1: OWNERSHIP: What part of an information asset belongs to an organization and what is simply part of an employee's general knowledge?

Information, knowledge, and skills we develop in the course of working on projects can be inextricably intertwined. You're the project manager for an effort to reengineer your company's marketing operations system. You have access to confidential internal memoranda on key organization strategic and procedural information. To build the new system, you and your team have to go for some advanced technical training on the new technology products you'll be using. The new system you build is completely revolutionary in design and execution.

Although there are areas of patent law that cover many such situations, there's not much in the way of case law testing this just yet, and of course laws vary between countries. Clearly, you've built an asset owned by your company, but do you have a legitimate claim to any part of it? Can you take any part of this knowledge or even the design or code itself with you to another employer or for the purpose of starting your own company? Suppose you do strike out on your own and sell your system to other companies. Is the ethical dilemma mitigated by the fact that your original company isn't in the software business? Or that you've sold your product only to noncompeting companies? What if we were talking about a database instead of a system?

Reflection:

This issue might cause a person to go crazy over the copyright. This literally shows that there should be limit to how and where the company and the developers claims on their software product. This issue also tells us what are the risks if we sell a software product that might not be used by client.

Ethical Issue 2: SECURITY: What's the liability exposure of managers and the organization?

Can system owners be held personally liable when security is compromised? When an organization holds stewardship of data on external entities—customers, individuals, other organizations—and that data is compromised, to what extent is the victimized corporation liable to the secondary victims, those whose data was stolen?

Organizations generally have internal policies for dealing with security breaches, but not many yet have specific policies to address this area. Managers who do not secure the systems for which they're responsible, employees who cavalierly use information to which they should not have access, and system users who find shortcuts around established security procedures are dealt with in the same fashion as anyone who doesn't meet the fundamental job requirements, anything from transfer or demotion to termination. Should compromised or ineffective security be held to a higher standard?

Reflection:

This issue is about how companies handle software breaches. This might become a fight between the managers and the employees. Who will be responsible for that security breaches. I think the companies should have policies or law stating who handle those problems and how to counter it.

Ethical Issue 3: Individual actions and functionality

A key ethical issue facing individual software developers and software companies overall relates to "how much of the other guy's functionality should I use?" Even when within perceived, reasonable legal boundaries, individuals cranking out software source code and software companies making strategic software product-direction decisions must weigh the ethical implications of copying another party's product vs. the desire to make a quicker profit.

On the individual level, consider the case of VMI and Autodesk, Inc. This case deals with the ethical implications of an employee jumping ship to a competitor and allegedly using a portion of the prior company's software product in his new company's product. As this case shows, it could be surmised that because of the technical complexity and perhaps "difficult to quantify" nature of software algorithms and because programmers have access to a broad range of technical material, software developers are held on a longer leash. Thus, it can be argued that individual software developers need to be highly accountable from an ethical perspective because there is a greater temptation to deviate from professional norms. On the hand, it can be surmised that individual software developers need some creative latitude in their profession. In other words, as long as software source code is not copied line-for-line, it is acceptable to use an algorithm for use with other products at other companies.

Reflection:

This issue still happening today. Employees shipping to competitor company. I think this might because they think they will not fit in their previous company. Or they bullied in the company. Or because they are not relaxed working in a pressured company. This might end up companies going down because the employees who shipped to competitor company might tell to the managers what kind of company they are in before they shipped.

Ethical Issue 4: Requirements and risks

If the software engineering department agrees to make modifications in a complex system, it should do so with the understanding that the time required will be sufficient to perform a thorough check out. Pressure to accelerate the work was obviously there driven by financial benefits in one instance, and the potential for saving lives in another. But a premature, poorly designed effort could clearly create unforeseen problems that might cost money and lives.

The ethical obligation of the software engineering department is to understand the requirements and the risks, to apply their best technical judgment and to candidly discuss the options with the customer. Agreeing to perform a task poorly simply because a customer requests it is not acceptable. Any organization must establish internal performance standards that supersede other demands whether the customer agrees to assume the risk or not.

Reflection:

This issue clearly tells us about being too proud to oneself. Saying to the client that you can modify any system but the truth is you cannot might end up negative feedback to the company or yourself. You should know where are your limits are and learn how to humbly disagree to any request you cannot do.

Ethical Issue 5: Application of software

"As software takes over more of our lives, the ethical ramifications of decisions made by programmers only become greater. Unfortunately, the tech world has always been long on power and short on thinking about the long-reaching effects of this power. More troubling: While ethics courses have become a staple of physical-world engineering degrees, they remain a begrudging anomaly in computer science pedagogy. Now that our code is in refrigerators, thermostats, smoke alarms, and more, the wrong moves, a lack of foresight, or downright dubious decision-making can haunt humanity everywhere it goes. Peter Wayner offers a look at just a few of the ethical quandaries confronting developers every day. 'Consider this less of a guidebook for making your decisions and more of a starting point for the kind of ethical contemplation we should be doing as a daily part of our jobs.'"

Reflection:

This issue is also still happening today. Having programs embed into the hardware but giving wrong outputs might cause clients go crazy over having a defect system. Like the Samsung company. Their clients go crazy because their Samsung Note 7 explodes while charging their phone. Moral lesson of this story is that you should be testing first the safety of your product before giving it to clients.

Ethical Issue 6: Whether -- and how -- to transform users into products

It's a well-worn adage of the startup era: If you're not paying for a service, you're not a customer; you're the product.

On the Internet, "free" services abound. In fact, the question of where the money will come from is often put off, being off putting. We just build the amazingness, keep an eye on the adoption metrics, and figure someone else will take care of the dirty work of keeping the server lights on. Worst case, there are always ads.

Developers need to be upfront about who will support their work and where the money will come from. Any changes should be communicated to users in a clear, timely fashion to avoid shock and blowback. Transforming people into products is an ethical shift not to be taken lightly. Shady ad deals, shady ad networks -- we need to be careful how we handle the implicit trust of early adopters.

Reflection:

This issue is about how developers get money in their software. Having their clients became their products without their consent is not good. They should be wary of what they are doing in your software. And I think developers should wary about the client’s will on taking your software.

Ethical Issue 7: How free does content really want to be?

A number of businesses depend on serving up content without paying those who create it. Some turn around and sell ads or even charge for access. These businesses often couldn't survive and couldn't price their material as attractively if they had to shoulder their fair share of the development costs. They develop elaborate rationalizations about "sharing" or "fair use" to cover up an ethically shaky decision.

Developers must ask themselves how their code will support everyone in the food chain, from creators to consumers. Do the people creating the content want their work to be distributed this way? Are they happy to work for exposure or attention alone? Are they given a fair share of the revenue?

Not considering these questions amounts to turning a blind eye to piracy. After all, not all information just "wants to be free."

Reflection:

This is like the issue in Ethical Issue 6. But this time, it is about the content of the product. I think we should develop systems that will not just all of us will benefit, but will also we can survive by doing that product. We also should explain to clients that we want the software to be free but they should think how the developers survives.

Ethical Issue 8: How far to defend customers against data requests

If you collect data, it's a safe bet that your organization will someday be caught between serving your customers and serving the government. Requests to deliver data to legal entities are becoming increasingly common, leaving more and more software and services organizations to contemplate to what extent they will betray their customers' privacy before the law. You can scrutinize these requests and even hire your own lawyers to contest whether they are truly lawful, but the reality is that the courts will be debating legalities long after your funding runs out.

There are no easy solutions. Some companies are choosing to leave the business rather than lie to their customers. Others are trying to be more open about requests, which the government often tries to forbid.

Reflection:

This issue is about client’s information privacy. The companies should not be having good time about client’s information privacy. All of us should think that we have our own life. We have our own information we would want to keep to ourselves. All of us should be thinking “What if my password will be broadcasted around the world and I having a private conversation with Hydra to taking up the whole word?”

Ethical Issue 9: How to deal with the international nature of the Internet

The Internet runs everywhere, avoiding many of the traditional barriers at the borders. This can be a recipe for legal headaches when customers A and B are in different countries. That's only the beginning, because servers C and D are often in entirely different countries as well.

This leads to obvious ethical issues. Europe, for instance, has strict laws about retaining personal information and views privacy breaches as ethical failures. Other countries insist on companies keeping copious records on dealings. Whose laws should a company follow when customers are in different countries? When data is in different counties? When data is transferred across international lines?

Keeping up with every legal contingency can be Herculean, leaving many organizations surely tempted to bury their heads in the sand.

Reflection:

This is about culture barrier. What if I create a software based on our countries’ tradition and I want to sell it on other country that having a law that they do not want that tradition? We should be thinking also their traditions and beliefs to have a good software that applies to all of us.

Ethical Issue 10: How much to give back to open source

Everyone knows that open source is free. You don't pay anything and that's what makes it so wonderful and complex. But not everyone contemplates the ethical issues that come with using that free code. All of the open source packages come with licenses and you need to follow them.

Some of the licenses don't require much sacrifice. Licenses like the Apache License or the MIT License require acknowledgement and that's about it. But other licenses, such as the GNU General Public License, ask you to share all your enhancements.

Parsing open sources licenses can present ethical challenges. One manager from a big public company told me, "We don't distribute MySQL, so we don't owe anyone anything." He was keying on the clause, written decades ago, that tied the license's obligations to the act of redistributing software. The company used MySQL for its Web apps, so he felt it could take without giving back.

There are no simple ways to measure the ethical obligations, and many programmers have wasted many keystrokes arguing about what they mean. Still, the entire endeavor will grind to a halt if people stop giving. The good news is that it's often in everyone's best interest to contribute because everyone wants the software to remain compatible with their use of it.

Reflection:

This issue is about having open to all. If we create a software, did we want it to be contributed to the open source community? Having others reprogram your software? When we want it to contribute to open source community, we should limit to how they change our system and also we should think how we will be benefit on that software given in the open source community.

Reference:

http://www.techrepublic.com/article/10-ethical-issues-confronting-it-managers/

http://blog.ip.com/2000/12/legal-ethical-and-managerial-issues-of-software-ownership/

http://csciwww.etsu.edu/gotterbarn/artge2.htm

https://developers.slashdot.org/story/14/04/21/212238/the-ethical-dilemmas-todays-programmers-face

http://www.infoworld.com/article/2607452/application-development/12-ethical-dilemmas-gnawing-at-developers-today.html