Candidate number: HZPM0

**Professional and Ethical Culture**

Throughout the career of engineering moral and social issues will always arise. When dealing with those issues, we must be aware of professional ethics. Professionalism, which means having extensive expertise in certain areas, and maintaining a good relationship with the world under ethical governance. Software engineering is a type of profession, which is governed by moral and ethical guidelines. Profession will help software engineer define their identity and thus establishing a good understanding of the profession and professional culture has a critical role of behaving ethically.

Formally, profession has been defined in the sociological account, social contract account or Davis's definition. According to Davis: *A profession is several individuals in the same occupation voluntarily organized to earn a living by openly serving a moral ideal in a morally permissible way, beyond what law, market, morality, and public opinion would otherwise require.* [1] Davis' definition introduces constraints on professionalism, which is stricter than the legal requirements. Profession usually is a group of people who are willing to conduct with a morally desirable goal. Davis addresses ethics as an important part of professionalism.

Typically there are three stages when developing your professional identity:

1. Independent operator: At this phase, profession is to satisfy fixed and clearly defined external guidelines and expectancy.
2. Team-oriented Idealist: conforming expectation of peer professionals
3. Self-defining/Integrated professional: professionalism has been integrated into oneself’s personal identity. [2]

By now, it is commonly agreed that the primary obligation of engineers is to promote public good instead of encouraging loyalty to their client or employers during early ages. The term ‘*public good’,* means social welfare. Although software engineers need to obey their managers to avoid causing chaos in team management, the problem is that if engineers obey everything they are told to do, there is no need for a code of ethics and a professional culture at all. The company's service, or software engineering in general, would have and already had a huge impact on society. Engineers should use their expertise to serve society, rather than making the society seek help from the engineers. To summarise, to build a strong professional culture, the employees are required to act according to professional ethics and always consider the impacts and value to the society in the first place [2].

Guidance in acting ethically, not only includes what is required by the law but surpasses it much further. The law is the bottom-line of conduct. A software engineer needs to at least be aware of relevant provisions from laws including Intellectual Property(IP) Law, GDPR data protection and legally enforceable contract.

Ethical training is required for all employees to understand how to conduct legally before starting their career. The company is headquartered in London, the understanding of the GDPR is essential. Any project which is inside Europe that is concerning personal data of European citizens is governed by the GDPR. Although the status of post-Brexit is still unclear at the moment, the company should always require the employees to comply with the GDPR to avoid possible troublesome circumstances. The developers of services who work with personal data must be properly trained to understand their responsibilities and the principles of conduct. We must underline that any form of processing (gathering, modifying and analyzing etc. ) of personal data must inform the individuals the purpose and conditions of processing. Software intellectual is protected by IP law. Arguably the open-source nature of computer science contradicts with some regulations and software is hard to patent. But coding could be considered as creativity and even the use of open-soused code is permitted under certain licenses. The decisions are often difficult to make and thus we need to draw close attention to these regulations when operating.

We could have an ethical office whose job is evaluating and assessing the ethical risks of the project and giving advice and guidance based on their assessment. Usually, approval from an external ethical committee is required if personal data is demanded. A project cannot start without ethical approval.

Most of the time, professional ethics are outlined in the code of practice, which often focuses on expectations and prohibitions for a particular profession. Codes of professional practice, for example, British Computer Society(BSC) and ACM. Employees are recommended to look at both frameworks, based on these codes of practices and manage their behaviour accordingly and comply with the codes. The main reason for implementing these frameworks is because professional ethics is different from personal morality or personal restrictions of manners. Both frameworks emphasise on the contribution to the society’s well-being which is also the ultimate goal of the company’s ethical culture[2].

We divide the professional ethical culture of the company into three parts: prohibited actions, preventing harms and aspirational ethics. Whilst fulfilling the obligation to promote the public good, we must be clear about what is being barred. It is more effective when we mention specifically what is prohibited rather than stating what is encouraged. Examples of prohibited actions could be extracted and referenced from the BSC and ACM code of practices or the NSPE provision such as bribery. On top of everything, the most fundamental and essential point is not to cause harm to others and actively seek to prevent harm. The company also should define the aspirational ethics in their ethics culture, which can be considered as helping others to strengthen their ability to improve the well-being of the society. Engineers are conscious of their obligations and non-obligatory and supererogatory actions such as contributing to Engineering Without Boarders[2].

The company expects engineers to be reliable and trustful. As the code of ethics practice requires, they should always be honest and impartial when making professional decisions. Any form of dishonesty is strictly prohibited as it violates the guidelines regarding the code and ethics and it misinforms the others causing them to make decisions without proper informed consent. Concerning utilitarianism, dishonesty would destroy the chemical reaction of the team, threatening the relation and trust that has been built and eventually decreasing the quality and productivity of work.

The importance of confidentiality and intellectual property must be recognised. Disclosure of private information regarding the client or the company would result in serious consequences. Intellectual property involves a wide range of technologies in a complex context, line drawing method can be used in this circumstance. Another point is, as stated in the code of ethics, conflicts of interest must not affect the professional thinking of the engineer and it should not lead to any kind of mistruth and misconduct.

A critical attitude to the technology must be adopted as by all means software engineers would be involved in the development of new technologies and services that are created in the project. It must be acknowledged that technology would introduce both positive and negative effects on society and engineers are responsible for minimising the negative effect as much as possible. In general, we advise the engineers to establish a good understanding of the professionalism and professional ethics by making use of cases in developing necessary skills for ethical practice meanwhile with a deep perception of the company’s culture[2].

Conflicts will arise at the workspace, thus the staff needs to be morally-responsible. The employee should have good moral thinking skills and this often requires the employee to be familiar with a set of ethical methodologies and thinking techniques. Engineers' priority should always be valuing their actions to customers and the community. They should be able to break the moral situation into four categories: factual issue, conceptual issue, application issue and moral issue to aid them in resolving it. An eligible engineer knows how to apply approaches like line-drawing method and creative-middle-way solutions to solve moral issues and a skilful engineer would be able to combine the concept of common morality, utilitarianism, virtue ethics and respect for persons in the most appropriate way when analyzing a complicate moral situation. Line drawing method can be used to determine whether an action is acceptable or not when it lies in the unclear boundary between ethical and unethical. By taking both the paradigm case and anti-paradigm case into account, we are clearer about whether an action is problematic, and we would be able to avoid such behaviours. The creative-middle-way method is another option when there are conflicting moral values drawn from different parties. Finally, when conducting moral evaluation, the intention behind an action can be critically important.

Consider the scenario:

Software engineer, P, has been tasked to obtain consultancy from an external organisation to help the company understand how best to adopt its desired microservice approach to system development. Several firms have submitted bids to provide the consultancy at varying prices, and that all meet the company’s required standards. P must decide which to accept. The lowest cost bid has come from a company at which a member of P’s family works. The family member’s name is not on the submitted bid and they are not part of the proposed consultancy team

Does P has a conflict of interest and should the low-cost bid be accepted? Line drawing method and paradigmatic analysis could help in this case. By coming up with paradigm cases and adding test cases we get:

**Is the family member involved?** No X------------- Yes

**Decision making** sole X------------- team

**bribery** no X------------ yes

**Is there benefits to P?** no -X----------- yes

**cost of bid** low X----------- high

**quality of service** low-----X------ high

From this analysis, as long as the quality of the service is satisfying and the company has legitimate reasons to support their low price we should go with the lowest bid. It seems like the benefits to P is limited. Bribery is unlikely to happen and P is making a professional decision. Thus it should be morally acceptable to choose the company with the lowest bid.

**Team management and Project Management**

Agile methods are currently adopted by the team. To give a brief introduction, agile is a set of values, beliefs and principles that can be used for software team development with the emphasis on the ability to respond rapidly to the changing conditions, team collaboration, iterative processes, continuous planning and incremental delivery. In the early stages of the IT industry, when typically a software is developed under a waterfall approach which consists of a linear life cycle of development where every phase must be fully completed before progressing to the next phase. Agile methods specifically targeted the drawbacks of the traditional method such as the involvement of the customer and shorter work cycles. Researches have shown that agile methods can yield 37% more profit [3]and agile projects generally are 28% more successful [6]. Agile is an umbrella term for all the methods that comply with Agile Manifesto. Popular agile frameworks are, for example, Scrum and Kanban.

As a start-up company with a small team, the benefit of agile is obvious:

* Customer is involved to ensure the product meets the needs
* Shorter cycle team can give frequent feedback to the team. Allowing for necessary modifications and corrections in developing. The amount of wasted effort is reduced and overall the development time should decrease.
* Risk is reduced
* Early release of the software has a more positive economic effect

However, obstacles regarding the cost and capitalism must be brought into account.

The development of software solutions needs to be well-supported and sufficiently funded by the enterprise to deliver a quality service that meets the demand. This may include capital expenses (*CapEx*), and operating expenses (*OpEx*). Capitalization required the project to be feasible and there is a certain level of confidence that the product would be successfully developed and delivered. In the waterfall project, software capitalization practice is well-defined. However, agile development lacks relevant phase gates. In agile programs, requirements and design principles are constantly changing and updating. As a result, it is a difficult job to anticipate the expenses and there isn’t any formal phase gate for official capitalization stage. The process would become unreliable if the finance service fails in accommodating the changes.

The solution could be, associating the development of features that are aligned with the relevant commitment period to potential capitalization to pass the feasibility analysis. As features extend the functionality of the software meanwhile a feature can be easily identified for potential capitalization. Most of the time, user stories are directly linked with a feature. Agile Lifecycle Management(ALM) tooling can support the implementations of the stories. Agile Lifecycle Management(ALM) tooling has query functions can help calculate potential CapEx of the implementation of user stories. Typically mechanisms in calculating CapEx are: via Story Hours, Point and the number of stories. To fully capitalize on agile processes, enterprises need to take advantage of agile methods across the organization so that the entire supply chain is examined and optimized [4].

More potential issues are hidden. For future benefit, we could imagine that as the company grows, it is very likely that we would have larger teams and each team has its own goal. Several points we need to address when scaling the agile development[5]:

1. Although the number of people increases, we still need to keep the size of the team, usually limited to 6 to 7 people to ensure the efficiency of communication. To maintain the momentum and chemical reaction of the team, we should avoid personnel change to the best we could.
2. Each team should consist of members with different specialities rather than the whole team is specialized in one direction and should focus on one feature.
3. Each iteration should have a relatively short length. However, this could be one of the major challenges in a large and complex agile project.

One of the challenges is: multiple teams may have different iteration lengths and different team may progress with different pace. The challenge now is how to synchronize the deliverables of each team? It is unrealistic to expect each team to finish at the same time. But the productivity is hindered when the team is waiting for lagging team to integrate the whole service. If the team decides to continue working on the feature without real-time feedback, the risk is the project might be headed in the wrong direction.

Microservices can be taken as the desired approach to facilitate scaling for the current system architecture. It has enormous benefits in the future. The key idea of microservices is to divide the architecture of a large and complex application into several small services. The benefits would be more evident as the scalability of the project increases. To make full use of it, it is usually accompanied by the idea of continuous delivery, continuous integration and containers. This is favoured by many big companies, for example, Netflix and Amazon. Microservices are required to split the application into small-focused, independent, language-neural and loosely-coupled modular services. We could break our application into different services like booking services and tracking services, driver services and customer services. We could have different services focusing on different platforms: IOS, Android and web app respectively.

In traditional Monolithic architecture pattern, as the scale builds up, it would result in large and complex code with weak modularity and code boundaries. In the company’s context, when customers launched the app, the software is deployed as a single application with the booking service, tracking services etc. There are some conveniences come with it due to its simplicity. However, it becomes much more painful as the scale develops[6]:

* Modularity breaks down over time, which introduces a steeper learning curve for the developers. Thus the quality of code would decrease over time.
* It also goes against the use of continuous development and continuous integration and making the software becoming unscalable and unreliable overall.
* Unreliable: Shutdown of one function might trigger the entire software to go down
* Inflexible: Extremely costly in adopting new technologies

In a microservices architecture, as the software is divided into independent small services, it could hugely benefit the speed and flexibility of the development. Change in one service becomes simplified and would not trigger more bugs in other services at the same time developers do not need to consider other services. Overall the project is benefited from:

* Improved code quality and reduced development time
* During deployment, it is possible to re-deploy one modular service rather than the entire application and further facilitate the idea of continuous development and continuous integration.
* Bugs are located in their domain. It would not produce further problems in other components and debugging becomes easier
* Reliability is strengthened as the entire app would not be taken down if one service is problematic
* Facilitate granular scaling: allows each service to be scale independently rather than scaling the application as a single instance. This can have a large effect on improving customer experiences and use the resources more efficiently.

Microservice does not necessarily reduce the complexity of the project. It is more expensive in terms of interconnected collaboration and team management especially when each team has their own set of tools and technologies. A good microservice should be language-neutral and each service is nicely bounded. It is crucial to have an effective interface to allow communication between each service. As each service would function on its own and the services would also function smoothly when running together. As the company is still in its very early stage, the use of microservice at this point would minimize the cost for culture-changing and reduce the pain of transformation.

As mentioned earlier, software engineering should act in a way that will promote the public good and the company has the responsibility to protect the environment. It is not only an obligation to society, but it also what is expected by the customers and the government.

Sustainable IT refers to the manufactory, management and use, disposal of information technology in a way the minimize the effect to the environment. Estimates showing computers and data centre account for around 2-4 per cent of global carbon emission. Our initial goal is to reduce energy consumption by 50% in 10 years. At the starting point, this could be as simple as switching off the light and the computers and always choosing to use renewable material rather than plastic. We should reduce unnecessary travels and use video conferencing instead of meeting in-person. To further reduce travel emissions, the employee should be allowed to work at home for 2 days per week. Depending on the resulted productivity, we might shift to home-officing if it is viable.

Low energy consumption does not necessarily mean sustainability. We should gradually progress to power the office with 100% renewable energy and use 100% electrical vehicles in our service with cooperation of London electrical vehicle company. However, our primary concern is to reduce energy consumption at the current stage. We could seek help from software consultant companies for their guidance on operation comply with the EU environmental regulations. There is a wide range of frameworks to help the company manage and report the environmental effect of their activities imposed that could aid us in progressing to more sustainable IT. As the scale of the company increases, we might need our own data centers. At the planning stage, we would like to design the data center in a way that is powered with a sustainable source and promote the usage of sustainable materials. If the data collected is legitimate to share with others, we should publish it to save the energy consumption of the other companies. We should also work on developing the algorithm, which takes the carbon emission into account. For example, based on the calculation of Big data, when collecting vehicles we should aim to increase the efficiency of each journey and make maximum use of the capacity of the car.

These procedures will not only benefit the environment but also enhance the company’s image. Similarly, the green reputation could also benefit the company with its employee's retention and recruitment. The direct effect of reducing energy consumption is lower operational cost.

Greenwashing must be prohibited. When we try to move to 100% electrical vehicles, we need to make sure the electricity is produced greenly. Arguable it would generate more pollution than fossil cars when the electricity is generated in a dirty way. We do need to address the issue that being green and sustainable is very likely to increase the cost of production and operation. We need to make compromises and balance it with the company's economically target. However, there is a global trend toward sustainability. Customers are more likely to choose greener services even if it means that they need to pay a bit higher price. However, this initial cost also offset by the benefit it offered, such as lower energy bills.

**Reference**

[1] Michael Davis, ‘‘Is There a Profession of Engineering?’’ Science and Engineering Ethics, 3, no. 4, 1997, p. 417.

[2] Engineering Ethics: Concepts and Cases Charles E. Harris Jr, Michael S. Pritchard, Ray W. James, P.E., Elaine E. Englehardt, Michael J. Rabins, 6th Ed. Cengage, 2019. ISBN 978-1-337-55450-3.

[3] Weill, P. (June 14, 2007). IT Portfolio management and it savvy: Rethinking it investments as a portfolio. Centre for Systems Research (CISR) MIT Sloan School of Management.

[4] Scaled Agile Framework. ‘CapEx and OpEx’. Accessed 21 May 2020. <https://www.scaledagileframework.com/capex-and-opex/>.

[5] Rigby, Darrell K., Jeff Sutherland, and Andy Noble. ‘Agile at Scale’. *Harvard Business Review*, 1 May 2018. https://hbr.org/2018/05/agile-at-scale.

[6] dotCMS, ‘What Are Microservices, And How Do They Aid Agile Development?’, *dotCMS Content Management System*. https://dotcms.com/blog/post/what-are-microservices-and-how-do-they-aid-agile-development- (accessed May 22, 2020).