Timothy Moll

# **Empathetic Software Engineering**

Software engineering is an interaction between people. This may seem abnormal to say, as software engineering is perceived as a one-way relationship between someone and a computer, but this is not the case. Software engineering is ultimately an interaction between the writer of the software and the user of the software. This misconception that software is not a social activity has led to the idea that software engineering does not need empathy to succeed at its tasks. After my time at St. Thomas, I now know this is false, and that software engineering is actually in dire need of more empathetic practices to allow for the industry to become more welcoming and to allow for more engagement of software with more people. This increased empathy will make for more useful software, increased diversity among software engineers, and a better world because of it.

Empathy is integral to software development. Software development's goal is to make a product that completes a task for someone. Now to be able to complete any task you must understand that task. When creating software that completes a task for someone else you must understand that person’s point of view to fully understand the problem they need to solve with your software. Even when hearing the description of the objective facts on the problem, this is not enough to solve a problem. In software engineering, there is never just one way to complete tasks, and this openness to problem-solving means that for different users with the same problem you will have to solve it in different ways to complete that task for a person. If your software development lacks empathy you can fall into many pitfalls that will leave your project being suboptimal for the user, possibly even making their initial issues worse, as a misunderstanding of a problem can lead to adding to it.

If empathy is so crucial, why do many software engineering projects lack it? One of the factors of lack of empathy in engineering projects comes from development time crunches. Teams cannot always consider the impacts of their product when barely given enough time to create it. Along with this lack of time to develop there is also a lack of time given for interaction between users and developers. Many developers make products without ever interacting with the end-user of their product. Usually, this interaction is done by sales or management and not by the team creating the product. This lack of interaction means that many details can be misunderstood or ignored by management and the development teams can misunderstand the end-users needs and lives. The largest factor of lack of empathy in software engineering comes from the lack of diversity in software development teams. If a team is comprised of people who have no similarities to their end-user this can lead to many projects being made without the end user's different background being recognized. Development teams without representation may not have the life experiences needed to understand many of their potential userbases.

Now with that understanding of the importance of empathy, how my projects are connected to this issue. The Jekyll and Hyde essay highlights the double-sidedness of technology we use. Specifically the era of social media we are currently in. Social media has been known to negatively impact the mental health of many users. This mental health toll is not a goal software engineering teams should be shooting for. This toll is happening due to a lack of empathy given to users of the products over the potential profits that can be gained. This double-sidedness even comes out when looking at the social media companies themselves, they are created to entertain people and make for some enjoyment in users, but in reality, this same company is also taking a mental toll on that userbase. They want to be perceived as Dr. Jekyll but will act like Mr. Hyde to please shareholders.

The next comparison is in the machine learning project and the importance of responsible machine learning. Machine learning is likely one of the largest spots of non-empathetic software engineering, as engineers who do not understand it can accidentally hurt many. Machine learning's goal is to predict some outcome based on input. This may seem great, but it can be used in bad ways as well. Machine learning projects where data used is historical data can negatively impact people of different communities, as historical data can cause current machine learning models to uphold biases of the past. This misuse and mistake, when paired with a non-empathetic engineering team, can lead to many negative issues for communities, as loans and job applications can use these models and hurt growth. This can lead to many negative feedback loops that continually hurt the people and community.

What can a negative feedback loop look like? Depending on the machine learning model itself it can be predicting something like the probability a certain person will default on a loan. So the model will take in all data points it has and will output that number. Some data points may be, age, location, salary, and credit score. Well, this can mean that a person in a community that has historically been inhabited by oppressed people can have a predicted lower probability of paying off a loan than someone in the next neighborhood with the same values for all other variables. This is bad because then the model would give this person a lower probability of paying off the loan, and likely would lead to a higher APR loan. This higher APR loan can lead to this person who normally would have paid off this loan to have to default. This model would see that it predicted correctly and may put even more weight on that neighborhood being bad for loans, making it even harder for the next person from that neighborhood to get a good loan. This is an example of the negative feedback loop that non-empathetic machine learning models can have.

Throughout my years at St. Thomas, one main factor of action for the common good was the act of communication with others whose lives and opinions differ from your own. This is exactly what empathetic software engineering must do. Engage in interactions with people who differ from you. Software engineers must interact with non-technical people who need the product. I think this interaction is the first step software engineering teams must enact to have more empathetic software engineering practices. This interaction will also only help everyone involved. Software teams will likely have their understanding of software products they make broadened by seeing others' perspectives of it. It will also broaden the perspective of the person the software team interacts with as they will gain the perspective of the software team as well. The teams will also gain insight that will make the outright product better and the user will be happy along with the team. The other thing that should happen is the software teams themselves must also differ, the makeup of a software development team cannot consist of people who have the same life experiences as this will hold back the backgrounds a team will be able to fully understand.

Now, this may sound good, but this seems to be only helping people who are interacting with the development of a project. What can this empathetic software engineering do for people outside of this interaction and their communities? Well for starters, a person who is not involved with a project's development but uses the final product will be more likely to have been acknowledged. This means that this person who may have had their needs ignored will now have more opportunities for their uses and needs added to the development process, leading to a much more enjoyable experience and more representation. The next aspect that can have a great impact is this positive experience with software engineering along with more diverse hiring practices that can lead this person to pursue a career in the field. This is one of the ultimate goals of empathetic software engineering. Opening the doors for more people to join the industry. This opening of doors will allow for more brilliant engineers to join in and bring their experiences and background to help diversify software and make its end products even better for everybody to use.

In conclusion, I believe my time at St. Thomas showed me the importance of the opportunities to interact with others and pursue the common good. In software engineering, the interaction with more people will allow for more empathetic engineering practices to occur. These engineering practices and hiring processes will allow for more representation in the industry and better software it. All in all, software engineering is a form of human interaction, and to make human interaction more strong we must allow for more of those humans to engage in that interaction.