The Precautionary Principle in Computer Programming

Brian Batinchok

December 8th, 2015

PHI3626 0W61

Dr. Jones

**The Precautionary Principle in Computer Programming**

Computer programs are developed to make the lives of users easier. The intent of a developer is to write code that accomplishes a task, whether that be to open a file or to initiate a transaction from a financial database. However, end users often do not know the scale or the complexity of the code written to process a task for a computer nor will that user know the possible ethical breaches that could occur when developing the program for use. Precautionary principles, which state that actions should not be taken if a consequence could be ruled dangerous, are put into place to protect developers, companies, and end users from unwanted or undesirable outcomes. Developers hold a great deal of power when programming and can be both ethically and legally bound for the code that is developed.

Illustrating the issues that arise during software development will be easier to understand with a working example. Let’s say that a major bank is trying to create a web program that allows for financial statements and transactions to occur at a user’s home. Sensitive information passes through both the Internet and the software that is developed, ranging from account numbers, credit card numbers, social security numbers, addresses, and bank statements. Numbers can be updated and moved over the program that allow for payments and withdraws. Failure to properly protect this information would have disastrous results for both the bank and its users because of the power of the information contained in the program.

With this example there are three major points of possible security flaws: passing data to the Internet, passing data from the Internet to the bank, and the back-end code developed to process the passing of information. Developers take on the ethical burden of developing code that will not violate confidentiality and avoid harm to its users. Because this is a risky area for development it is important that the quality of code be very high; this prevents any outside threats, such as hackers and penetrators, from getting the information of its users. The code should also be safe for its users and also not incorporate any illegal financial practices like double billing or bribery. After the code has been developed, it must also go through rigorous testing and debugging to make sure it functions as intended before being deployed for release.

This is a large undertaking for any development company to go through. On top of the stringent coding safety requirements the project also requires a great deal of ethical oversight to prevent any financial or legal issues. However, because this is high risk does not mean that the product should not be developed. Home banking and bill-paying is an incredibly useful service offered for users and some would accept the risk for the convenience of not having to go to the bank. This is a strong precaution in software development; there is heavy regulation on the quality of code and programming practices that create the software utilized by banking and financial institutions.

Over time, developers created a series of principles called The Code of Ethics and Professional Conduct that is maintained by the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers to both help and hold developers accountable for code that is written in situations like this (ACM). Principles like confidentiality, software safety, and individual responsibility are core components to these codes of ethics and are highly encouraged to be followed by all developers (IEEE). Furthermore, universities are starting to place emphasis on developing code that fits best practices, both in syntax of the language and the quality and intent of the code itself (Cornell). Processes, such as cryptography, are also put into place to obscure information from outside sources.

Although the guidelines and standards are available for developers to view not all developers are bound by them. Following these standards is more of a moral obligation of a company or on a person-by-person level. There is always the potential for a bad company or developer to slack on these ethical boundaries, leading to low-quality code or code that is unsafe for enterprise use. Unfortunately code can sometimes be malicious, whether it be a virus or something more serious that steals personal and financial information from a user. These occurrences are in the minority of situations but happen often enough that users should be wary of any program that asks for sensitive information.

Like the previous bank example, this should not be a reason to adopt a weak stance and not develop code at all. Rather, code that runs such sensitive processes should be heavily regulated by companies and developers. Although this is more work in the long run and adds time and cost to development the result is a safer product. This translates to less potential risk incurred by the companies in charge of developing delicate code. In the case of the example, users will receive the largest benefit: a convenient service that has a large amount of information safety built in.

References:

Association for Computing Machinery. (n.d.). Retrieved December 8, 2015, from <http://www.acm.org/about/code-of-ethics>

Association for Computing Machinery. (n.d.). Retrieved December 8, 2015, from <http://www.acm.org/about/se-code>

Cornell Security Goals and Requirements. (n.d.). Retrieved December 8, 2015, from <http://www.cs.cornell.edu/~clarkson/courses/csci6545/2013fa/notes/goals.php>

Institute of Electrical and Electronics Engineers. (n.d.). Retrieved December 8, 2015, from <https://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers>

Precautionary Principle. (n.d.). Retrieved December 8, 2015, from <https://en.wikipedia.org/wiki/Precautionary_principle#Strong_vs._weak>

Programming Ethics. (n.d.). Retrieved December 8, 2015, from <https://en.wikipedia.org/wiki/Programming_ethics>

Security Requirements Engineering: A Framework for Representation and Analysis. (n.d.). Retrieved December 8, 2015, from <http://oro.open.ac.uk/10058/>