In search for green metrics

Juha Taina
University of Helsinki, Department of Computer Science
Juha.Taina@cs.helsinki.fi

Pietu Pohjalainen
University of Helsinki, Department of Computer Science
Pietu.Pohjalainen@cs.helsinki.fi

1. INTRODUCTION

Software will play an ever-increasing role in our defense against climate change. Since software systems are present in practically every field in the world, also software is present everywhere. From the climate change point of view, good software helps to reduce greenhouse gas emissions, such as carbon dioxide or ozone, while bad software causes to increase the emissions.

In principle the division to good and bad software is easy. First, implement and install software, second execute software, and third measure how much software helped to decrease or caused to increase carbon dioxide emissions. In practice, the division is not at all clear. In our earlier paper [1] it was analysed how large a carbon footprint a software package produces. The results were surprisingly small. The carbon footprint of a software package is negligible in most cases compared to other issues. For example, traveling easily dominates development footprints. Software development is more than just writing code. It requires office space, machinery, traveling, and other resources that are idle most of the time. The idle times need to be included in the development metrics.

The carbon footprint of software usage is not a clear metric either. The primary consumed resource of software is CPU cycles. We can estimate how many cycles a software package requires in its life cycle, and how large a footprint each cycle creates by multiplying the average cost of producing electricity with the average power consumption of a CPU cycle. Some might say that such an estimate is good enough, but when considering the system's overall effect, the estimation is but too meek. Software is more than consumed cycles. For this reason, we need established metrics that measure the indirect effects of software.

Usually software is part of a software system with a goal. The software helps the system to reach the goal. The indirect effects of a software package define two things: 1) how well software helps the system to reach the goal, and 2) how environment-friendly the goal is.

For example, let us have software that helps to reduce 10% of a carbon plant's emissions, without reducing its power output. Obviously this is good software since it helps the system (power plant) to reach its goal with smaller emission cost. However, it is not at all clear that it is green software since the system itself is not green.

On the other hand, let us have software that controls a pack of windmills. The software does its job but is not very efficient. Due to its lack of efficiency the windmills are not always working at maximum efficiency. Obviously this software is not helping the overall system to get its full potential but the system itself is very environment-friendly.

In both of these cases, the system's goal is to produce electricity. In the coal-burning power plant's case, the variable cost in terms on carbon dioxide is high, while in the wind-powered plant the variable cost is close to zero.

In absolute terms of saved carbon dioxide tons, the first software package was clearly better than the second, since it unambiguously reduces the emissions in a running power plant. However, in the terms of absolute number of carbon dioxide tons, the second software system is better than the first one, as the variable cost in emissions is an order of magnitude lower. Overall, it is not at all clear which software of these two was greener.

Currently, we do not have an established framework for estimating or measuring the effects of software systems' effect on climate change. As a large number of software projects are starting or already ongoing with the target to reduce emissions, we need to be able to quantify these projects' expected effects.

Once we have good metrics to measure software greenness, we can decide which software projects are the best for the climate. Such metrics would help to decide which projects deserve funding and perhaps how much funding each accepted project would get. We are still a long way from this goal.

2. REFERENCES

[1] J. Taina. How green is your software? Submitted to the *33rd Software Engineering Workshop*, 2009.