Looking Towards A Future Where Software Is Controlled By The Public and Not The Other Way Round

Magiel Bruntink, Jurgen Vinju

To cite this version:

Magiel Bruntink, Jurgen Vinju. Looking Towards A Future Where Software Is Controlled By The Public and Not The Other Way Round. ERCIM News, ERCIM, 2014, 99, pp.1. hal-01110831

HAL Id: hal-01110831
https://hal.inria.fr/hal-01110831

Submitted on 29 Jan 2015

Looking Towards a Future where Software is Controlled by the Public (and not the other way round)

by Magiel Bruntink and Jurgen Vinju

Nowadays, software has a ubiquitous presence in everyday life and this phenomenon gives rise to a range challenges that affect both individuals and society as a whole. In this article we argue that in the future, the domain of software should no longer belong to technical experts and system integrators alone. Instead it should transition to a firmly engaged public domain, similar to city planning, social welfare and security. The challenge that lies at the heart of this problem is the ability to understand, on a technical level, what all the different software actually is and what it does with our information.

Software is intrinsically linked to many of the challenges currently facing society. The most obvious of these challenges is data security and user privacy. Much of the software currently in use collects data. This data comes from a wide range of sources including recreational activities, personal health, messaging, street surveillance, financial transactions and international communications. Software is not only deployed on personal (mobile) computing devices but also through far-reaching government programs. In all cases, it is the software that tells each computing device how to participate in the act of data collection and process the activities that bring benefits. Whether these benefits are for the greater good of society, however, is not always clear cut. This prompts questions such as ‘Who is aware of the exact data collected by their smartphone, or where (on the internet) it will be stored?’, ‘What servers hold the contents of your software-supported tax return and in which countries are they located?’ or ‘Is there a database somewhere that somehow stores a picture of you linked to a crime scene?’.

Besides the obvious political and social aspects of these questions, there are more fundamental problems that still need to be addressed by software researchers and practitioners. The core problem that exacerbates the issues of software security and privacy is that software is not sufficiently well understood at a technical level, especially at the scales at which it is now being developed and deployed. All too often, software is so complex it can’t even be handled by the most experienced software engineers or explained by the most advanced theories in computer science, and too big to be summarised accurately by the automated tools created for that purpose. How then are policy makers or the general public supposed to be able to make software-related decisions that are based on facts and insight?

Given that software complexity is still an untamed problem, what consequences exist for data security and privacy? There are numerous examples of incidents in which the security of key systems in many (public) organizations have been breached. Recently, a serious vulnerability dubbed the 'Heartbleed bug' was exposed in a software (OpenSSL) that is supposed to secure vast numbers of Internet servers. It isn’t at all clear what happened to the data stored on those systems that were compromised by these vulnerabilities.

Heartbleed, in particular, provides an interesting illustration of the level of software complexity we are dealing with. The bug itself consists of only two lines of code, whereas the entire OpenSSL software package contains 450,000 lines of code [1]. Industry research into the existence of bugs or defects suggests a wide range in the bug ratio, from 0.1 to 100 bugs per 1,000 lines of code [2,3]. This ratio is strongly related to how well the software was developed and tested. Clearly, our current understanding of software does not allow us to develop software without bugs and is just one of the consequences of software complexity. Other considerations include the high cost and lack of performance.

Considering all this, we feel that the future of software should involve a radical change, best summarised as follows:

> *Software complexity should become a public problem, instead of simply remaining just a problem for the public. In our view, the current situation in which software is too complex to be handled properly should transition to a situation where software-related decisions can feasibly be made by non-experts, in particular policy makers and citizens.*

In our view, a positive development has been the installation of the (temporary) committee on ICT by the Dutch House of Representatives which is tasked with investigating several problematic e-government projects. We envision a similar public status for software as given to law making, city planning, social security, etc.. While all these social priorities still require a certain level of technical expertise, public debate determines their direction. There is a long road ahead to reach a point where software can join this list, We feel the following directions are essential to making this journey:

- investment in research that creates more accessible software technologies, for instance, domain-specific (programming) languages that achieve a better fit to societal problems and reduce software complexity;
- investment in empirical research that considers the current state-of-the-art practices in dealing with software complexity with a view to scientifically establishing good and bad practices, methods and technologies;
- the introduction of software and computing curriculum at the primary, secondary and higher levels of education to increase general software literacy and ultimately, foster a better public understanding of software complexity; and
- contributions to the public debate on the nature of software and its impacts on society, for instance, by arguing that society-critical software should transition to open-source models, enabling public control and contribution.

In conclusion, to arrive at a future where software is something we can all understand and control, as opposed to us being controlled by software and its complexities, a strong focus on software will be required in both research and education. Therefore, it is high time to generate public engagement on the complexities of software and the challenges that creates.

Link:

[1] Blackduck Open Hub: https://www.openhub.net/p/openssl

References:

[2] Coverity Scan: 2013 Open Source Report, http://softwareintegrity.coverity.com/rs/coverity/images/2013-Coverity-Scan-Report.pdf
[3] Watts S. Humphrey, The Quality Attitude, SEI, 2004, http://www.sei.cmu.edu/library/abstracts/news-at-sei/wattsnew20043.cfm

Please contact:

Magiel Bruntink
University of Amsterdam, The Netherlands
E-mail: M.Bruntink@uva.nl
or
Jurgen Vinju
CWI, The Netherlands
E-mail: Jurgen.Vinju@cwi.nl

Picture caption text

Figure 1: The core problem that exacerbates the issues of software security and privacy is that software is not well enough understood on a technical level, especially at the scale at which it is now being developed and deployed.