VU VRIJE UNIVERSITEIT AMSTERDAM

VU Research Portal

Software and Sustainability

Lago, P.

2016

document version
Publisher's PDF, also known as Version of record

Link to publication in VU Research Portal

citation for published version (APA)
Lago, P. (2016). *Software and Sustainability*. Vrije Universiteit Amsterdam.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 26. Jan. 2021

Software and Sustainability

ORATIE

Rede uitgesproken bij de aanvaarding van het ambt van
hoogleraar Software and Services
aan de Faculteit der Exacte Wetenschappen
van de Vrije Universiteit Amsterdam
op 15 januari 2016.

door

Patricia Lago

Copyright © 2016, Patricia Lago
All rights reserved unless otherwise stated
Typeset in LaTeX by the author

Software and Sustainability

Patricia Lago

15 January 2015

Rector, ladies and gentlemen.

Sustainability is broadly associated with Mother Nature and ecologic issues. In this lecture I will explore what sustainability means in our times of fast digital evolution and extreme complexity, and how software can help.

Sustainability

The sun is an incredible source of energy. It is the prime example of sustainable energy, that is the energy obtained from inexhaustible resources, and as such it serves the needs of the present without compromising the ability of future generations to meet their needs.

We all have a general understanding of what sustainability means. However, its definition changes depending on the sources we use or the context in which it is considered. In essence, sustainability is the ability to continue a defined behavior indefinitely [1], or at least to continue for a long time [14]. In a business context, it refers to managing the triple bottom line - a process by which companies manage their financial, social and environmental risks, obligations and opportunities. These three impacts are sometimes referred to as profit (economic), people (social) and planet (environmental) [7].

However, this approach does not fully capture the time element that is inherent to sustainability. A more robust definition is that business sustainability represents resiliency over time businesses that can survive shocks because they are intimately connected to healthy economic, social and environmental systems [7]. This time element comes back in all fields. In an ecologic context natural resources should be kept in balance indefinitely. In a financial context, economic sustainability is the ability of an economy to support a defined level of production indefinitely.

And what does this all mean in a technical context? While sustainability has been extensively researched in other fields, its role as related to ICT and

specifically software engineering is very immature. So those that are here for answers will be badly disappointed because I will leave you with even more questions than before.

In fact, Environmental informatics goes back to the early 90s, while fields like sustainable HCI, green computing and Green IT emerged around the year 2007 to 2008. Sustainable software engineering started in the same period. Hence, what sustainability means for green IT and specifically green software is still under debate. However, what we understand so far is twofold:

First, sustainability needs to take a holistic perspective on a big enough system of interest. Second, in all contexts, sustainability influences four common dimensions: environmental, social, economic, and technical [13].

Back to our sun. The ability to harness solar energy brings evident environmental benefits. But solar energy also has the power to influence people health and mood in a positive way, hence playing an important social role. It is also evident that harnessing solar energy requires economic investments as well as it implies access to suitable technical solutions.

For instance, Germany is since many years one of the world leaders in solar power installations and production of renewable energy. This could happen thanks to strong public support and political will. In the 90s, when German politicians passed the Renewable Energy Sources Act, they required utility companies to insert renewable energy producers into the national grid and to buy their power at a fixed rate. This guaranteed a modest return over time.

But more importantly, knowing that the industry was being firmly backed by the German government, investors confidently dove in the market with long-term investments, which helped financially stabilize those companies. The growth rate of solar energy in Germany has nearly tripled in the last 10 years [3], hence giving an incredible incentive to the national smart grid.

This example shows that the notion of sustainability entails the complex interplay of elements in the four dimensions. Being able to serve the needs of the present without compromising future needs poses strong environmental, social, economic and technical demands. While this is intuitively natural, transferring it to the demands of a heavily digital society like ours is not trivial.

Lets think of another example. Today, over half of the worlds population lives in cities. For the first time ever, more people live in urban areas than outside them. And although cities make up less than 5% of the Earths surface, they use up to 75% of its resources. According to a Philips research, 19% of global electricity consumption is used for lighting. We need more and more sustainable lighting solutions like this, where street lamps (called light blossom) use an array of petals with energy efficient LED lights on the bottom and solar panels on top. The blossom harnesses solar energy when the sun shines. When the wind blows it spins, acting as a windmill to further charge itself. The lights stay in a dimmed, low-power mode at night until someone walks by when they will brighten to full intensity to light your way.

Transformational Power

But smart lightning does not just decrease energy consumption. It has the power to influence people health.

In collaboration with Philips, the Municipality of Uppsala in Sweden studied how increased opportunity for outdoor playing affect children well being [4]. They installed a smart lighting solution in a city playground, teamed up with a local kindergarten, and observed amazing effects:

- Children spent on average 37% more of their time playing outdoors.
- As a consequence, 57% of the parents noticed an improvement in their childs mood after playing outside
- 43% also reported a positive impact on their children sleeping pattern.

LED Lighting reduces energy consumption up to 80%. This has a direct positive impact in the economic dimension thanks to a proportional reduction of electricity costs.

But combined with smart lighting-control-software, LED Lighting allows authorities to remotely manage lighting, uncover optimal usage patterns and adapt lighting to changing needs according to a daily or seasonal calendar, to planned citywide events, or (like in Uppsala) to have a positive influence on citizen well being.

This is the transformational power of software: it doesnt just glue features together; it transforms the way we think of our society and change it.

Smart Software

Smart lighting is just one of the many examples of this transformational power of software and Information Technology in our times. The Internet of Things revolution is finally upon us. A recent report by Gartner [6] estimated for 2015 1.1 billion connected things being used in smart cities, a figure which will rise dramatically to 9.7 billion by 2020. 81% of those things will come from sensors in smart homes and smart commercial buildings like concert halls, shopping malls, railway stations, airports, and universities of course.

The concept of a smart city refers to the idea of having sensors across the city in public infrastructure in order to monitor everything from public energy usage to implement energy saving measures, to which streets are most traveled late at night and require lighting, and which don't. The sensors wirelessly feed back information to computer servers (in the cloud) where it can be analyzed to create useful insights that help governments to properly manage cities. At the same time this is creating new business opportunities for commercial companies that can deliver novel or better (software) services, and to citizens that can use them to sharpen their own lifestyle around less expensive consumption patterns and a higher quality of life.

This very complex scenario, however, assumes access to endless computing capacity and high speed connectivity provided by services like cloud computing, broadband networks and data centers, which are and will remain major global energy consumers worldwide. This illustration from the 2014 Greenpeace report [9] confirmed the Netherlands as a major data center country by serving about one-third of the data center services in the European Union. The latest Greenpeace report on Building the Green Internet [8] shows that Internet data is now growing at 20% per year; and the collective electricity consumption of our devices, data centers, and networks will jump from 7.4% of global electricity consumption in 2012 to 12% by 2017.

Thanks to the investments of major IT giants like Apple, Google, and Facebook in adopting renewable energy, the carbon footprint of their data centers is decreasing. However, according to the Global e-Sustainability Initiative [10], the ICT sector is expected to be able to realize a truly sustainable growth, hence holding down its own carbon emissions, only if the following two factors will be realized: first, the trend will continue in replacing large hardware like PCs with more efficient devices like tablets and smart-phones; and second, software services will be able to fulfill the requirements of a smart digital society driven by end-users, which account for nearly half of the ICT sectors emissions.

How can we make software smart and sustainable at the same time? What are the software challenges ahead? I cannot forecast the future, but software certainly plays a pervasive yet silent role in this journey toward sustainability. In fact, we can already see a number of SE challenges we can reflect upon.

Software of the Past and its Unsustainability

With a step back in history. ENIAC was the first electronic general-purpose computer. It was capable of being reprogrammed to solve "a large class of numerical problems. Meaning that it could be programmed to perform complex sequences of operations, including loops, branches, and subroutines. The task of taking a problem and mapping it onto the machine was complex, and usually took weeks. After the program was figured out on paper, the process of getting the program into ENIAC by manipulating its switches and cables could take days. [wikipedia] - a bit funny now, but pretty amazing at the time.

The ENIAC signed the beginning of the software industry. Since then the software engineering profession witnessed important changes, two of which are probably less obvious than others and have been especially relevant for software sustainability or its unsustainability.

From being a highly specialized practice, programming has been purposely transformed into something for anybody: over the years, programming languages, development environments, methodologies and paradigms aimed at becoming simple, intuitive, visual. This often happened to the cost of completeness, costs and efficiency.

This is evident from the fact that, while according to Koomey's law the energy efficiency of hardware has doubled almost every 18 months for the past

60 years or so, Wirth's law showed that software gets slower more rapidly than hardware becomes faster. This is not due to lack of insights or competences. Rather, thanks to powerful and cheaper computing resources, resource optimization became less of a problem, and other, more business-related problems were put in the forefront.

The introduction of the smart phone (that combined telephony with mobile computing) changed the user experience forever. Much later, with the introduction of Voice over IP and the mass adoption of mobile computing and cloud provisioning, the mobile device in our hands has lost its local limitations and became an extension of the wide Internet. Software services and Apps seem to run in our phone, but in fact combine services and resources from anywhere in the world seamlessly.

While new software is being developed directly with this architecture in mind, legacy software needs to be migrated or modernized. In 2013 a study of Berkeley Labs estimated to 87% the potential energy savings of migrating US business applications to the cloud. At the time, however, it was unclear which applications should have been re-engineered to realize this potential, and what features should go in the cloud and what should remain local instead.

A negative effect of this immaturity is that the cloud is assumed to be green, i.e. energy efficient and optimized, while the field still does not offer sound methods to develop energy efficient software, or realize such optimization automatically. The integration of local and cloud-based software gives the impression of endless resources, which in turn further pushes organizations to move their assets to the cloud even more, often with insufficient insight in the real costs, and benefits.

The following digs into the software challenges in each of the four sustainability dimensions.

Technical Sustainability

By extending our smart phone, laptop computer, smart TV, and video-gaming platform with cloud-based services, software is expected to dynamically detect if the allocated resources are insufficient and dynamically scale and adapt. This level of flexibility demands that software is made aware of the changes in its context, which can vary from adding new servers when too many users are playing video-games simultaneously, to adding services on-the-fly, like extending your self-driving car with a digital tourist guide during a holiday trip.

Smart software is required to accommodate all types of changes that were unpredictable at design time, and still function correctly 24/7 while delivering the expected satisfactory quality. Andrew Moore (former vice president for engineering at Google, now Dean of Carnegie Mellon Universitys School of Computer Science) [15] wrote: How do you test and provide quality assurance on systems that are adapting all the time and will be running very differently over time on the basis of accumulated data?. The era of Apple and Google testing new products with hundreds of potential users before launching the best one

on the market is officially over. The key success-factor in any software industry is not anymore about what you deliver but how well you can ensure quality, being this efficiency, security, or usability. Allocating over-capacity is not the answer to this quality-of-service problem anymore, if we pursue sustainability and a greener cloud.

As my colleague Jan Bosch would say, the other key to success is how fast you can deliver. Speed has been a topic for many years, and indeed many companies conquered the market by arriving first, moving faster, and being even faster in changing their vision and providing innovation. This search for speed has been brought to extremes by the speed to which technologies emerge or change. Many companies report that they have to change their software products every six months (or even less) to follow the technology innovation cycle. This phenomenon led to software engineering (SE) methods under the umbrella of 'continuous delivery', and regular updates have evolved from exceptional to normal practice.

By pursuing speed we might be looking at the wrong problem. If technical sustainability is about building "long lasting" software, the real target should be to make software stable and independent from technology changes. Doing that would completely change the way software engineers think and work. This would free incredible resources now wasted in pursuing continuous change.

Smart software is also expected to autonomously detect if some of its parts decrease in quality, and replace them with alternative services from alternative cloud service providers. This is called dynamic discovery and composition, one of the promises of service orientation, which remained for many years impossible mainly due to lack of standardization. Elasticity might be finally for the taking now that cloud software harmonization is becoming possible.

However, while the standard notion of elasticity implies the 'flexible allocation of resources on need', a new opposite definition is emerging, that is 'to make needs (!) flexible, so that resources are optimized. For example, this illustration is about a research on defining smart charging of electric cars, a collaboration between the Amsterdam University of Applied Sciences and the City of Amsterdam. It shows the charging patterns in the city. On a daily basis, people use the charging points in different hours depending on being residents, taxi drivers, commuters, etc. Thanks to urban data analytics, smart software services have the potential to intelligently schedule smart charging to combine needs so that the whole city requires 50% of the foreseen charging infrastructure and electricity peaks are minimized. This means great benefits in terms of environmental, social and economic sustainability, ultimately enabled by smart and technically sustainable software.

Despite the plethora of fascinating software technologies that attract our attention, the 'human remains central as a both contributing and disrupting factor in the success of software developments. The wrong identification of user requirements is still one of the primary reasons for failure. New RE approaches are barely emerging to systematically combine gathering of data directly from the user experience, and using data analyses as requirements for improvement.

Of course this is just possible if we want to improve pre-existing software.

More traditional ways of directly involving the user are the various agile development methods. However, while they seem to accommodate the needs of the developers, they do not seem to help any more effectively in getting the requirements right.

But nowadays humans are not just the source for requirements. They are assuming a totally new role. End-users are increasingly digitally-skilled, have unprecedented access to services that they can combine themselves to create new features, and can share their opinions in a multitude of ways that are often seamless, crowd-sourced, analyzed, and finally used to create new services and innovative business models. Think of Uber for hailing a taxi or Booking.com for finding a hotel room: until short ago it would have been unthinkable to start a taxi company without owning any car, or a hotel reservation company without any hotels. Moreover, companies like Airbnb or Vanderbron are examples of user communities that, to various extents, organize themselves, and have transformed consumers into prosumers with minimal resources and value creation.

Social Sustainability

Software and ICT are already having a great social effect on our society. Only, we do not analyze or estimate it upfront.

For instance, ICT promises to give by 2030 1.6 billion extra people access to healthcare [10]. This will be particularly meaningful to disadvantaged or remote communities where ICT could help pensioners with limited mobility to access healthcare at home, or provide a farmer in rural Kenya with access to global crop, weather and market data, boosting his or her income.

When we develop socio-technical solutions, however, we only consider the 'immediate effects on production and use. Many critical effects play out over time. These effects benefiting or hindering society in the long term translate (or not) into sustainability.

As Melvin Kranzberg said, Technology is neither good nor bad; nor is it neutral. By which he meant that "technical developments frequently have environmental, social, and human consequences that go far beyond the immediate purposes of the technical devices and practices themselves, and the same technology can have quite different results when introduced into different contexts or under different circumstances.

A Bird's Eye View

Next to immediate impacts, Hilty & Aebischer [11] distinguished two additional orders of impact.

'Enabling impacts arise from use over time. This includes the opportunity to consume more (or less) resources, but also shorten their useful life by obsolescence (when we buy a new smart phone just because incompatible with newer applications) or substitution (when e-book readers replace printed books).

'Systemic impacts, in turn, refer to persistent changes observable at the macro level [11]. This includes behavioral change and economic structural change. This may translate into (negative) rebound effects by converting efficiency improvements into additional consumption, or new risks - like our dependence on ICT networks that makes a digital society also vulnerable. Systemic effects can also bring (positive) sustainable patterns of production and consumption.

To understand the long lasting effects of modern software-intensive systems, we must take a 'birds eye view that supports holistic reasoning and decision making involving software, hardware, human, and system elements. We must reason along the 'time dimension to set our sustainability target. And most importantly identify how the software quality characteristics contribute or hinder which of the four sustainability dimensions.

With such an architectural approach, we can determine if the smart lighting software we mentioned before does contribute the expected 80% energy savings and positively influence citizen well being. These are 'immediate effects we expect the software to directly realize. To do so, smart lighting must be configurable, and must scale to serve residential areas or city events of variable sizes.

The ability to quantify the social effect on well being determines how much money the city can save in healthcare, money that can be eventually allocated elsewhere like smart mobility and education. While these aspects are outside the immediate software engineering scope, being aware of their systemic impact supports SA reasoning, and sustainable evolution. Big data analysis and visualization are enabling technologies to monitor the extent to which software makes these goals a reality.

Case studies in industry show that, differently from other domains, software development does not necessarily cover all four dimensions. For example, Netflix, industry leader in streaming video content over the Internet for about 69 Million subscribers worldwide, made in 2009 a strategic decision to move all its customer traffic from their own data center to Amazons public cloud solutions. Nowadays Netflix runs 100% in the cloud. The original motivation was to stop focusing on predicting traffic and continuously re-architecting due to the rapidly increasing scale, and invest instead in building and improving their applications which is their core business.

Scalability became Netflix competitive advantage. Scalability also allowed them extreme elasticity with a corresponding dramatic reduction of their provisioning costs. Of course they could do that, because they did hardly have any legacy. Architecture assessment along the four dimensions and the three types of impacts allow to assess and monitor the benefits and risks of harmonizing old with new software solutions meant to be sustainable.

Environmental Sustainability

Software can contribute to environmental sustainability in various ways. The many believe that the greatest benefits come from its indirect impact, its capacity to support sustainable processes and inject a positive behavioral change outside the ICT sector, like software enabling smart homes, smart grids and smart EV charging.

However, we forget that the ICT sector uses 50% more energy than global aviation (source: Digital Power Group, also [2]), with data centers alone having overtaken aviation as a source of global CO2 emissions. The same data centers effectively exploit between 20% and 80% of their available capacity. Now that cooling is becoming increasingly energy-efficient, and data centers are increasingly powered by renewable energy, it is time to cut these 80% inefficiencies inside the servers.

Baby steps are being taken by making cloud-governance-software smarter. We are becoming better in predicting the needs for computing resources; hence we can better allocate resources for computation and data-storage. However, managing data-center-resources better does not remove the inefficiencies accumulated within the software implementations.

The real elephant in the room, hence, is energy efficient software [12]. In spite of the growing number of publications in the field, most provide only partial (if any) information about e.g. how to develop software that is energy efficient; what metrics, tools or contextual configurations should be considered when measuring or estimating the energy consumption of (certain types of) software systems or applications; and finally what is the real impact of software applications on the energy efficiency of the computing resources they need for execution.

As a result we are witnessing a paradox. From afar, researchers and practitioners have started believing that much has been done already, and the lack of significant adoption in practice would suggest that the impact of software is negligible. This further feeds the myth that major gains are to be searched in hardware optimization, or in using renewable energy resources instead. We are forgetting that computing resources and infrastructures are there because software needs them, and if software is bloat and inefficient any optimization will be just wasted.

We built the Green Lab, a computer lab here at the VU where we measure the energy consumption of any type of software. We dig into the complexities of the software implementation and its planned or observable behavior, and develop techniques to identify and verify the 'energy hotspots, i.e. software elements or properties, at any level of abstraction from architecture design to implementation, that have a measurable and significant impact on energy consumption [16].

But the Green Lab is much more than a lab. It is an accelerator for synergies between education, research and industrial practice. Our students learn the competencies of sustainable software engineering and bring them into the market. Our industrial partners contribute their software or their data, learn the

related energy footprint, and get best practices and innovate their know-how.

Finally, our researchers incrementally generalize their findings and build a consistent knowledge base for energy efficient software. Multidisciplinary collaboration with the partners of the Amsterdam Data Science initiative creates synergies to harness data science and visualization techniques for sustainable SE.

Economic Sustainability: How much does Green Software matter?

Finally, to be able to quantify the energy footprint of a piece of software we must be able to define theories about the relevant software characteristics that influence energy efficiency. We need knowledge about which design and coding decisions result in less energy consumption; and techniques able to handle the multiple layers of technologies that obfuscate the dependency relation from a software application down to the virtualization, infrastructure, and hardware components exploited for its execution. We developed inefficient software for decades. It seems just logical that we can optimize it to great extents.

Everybody asks me, How much? How much energy can we save? How much should we invest? And what is the ROI?.

I can give you a glimpse on few results we have obtained. For smart cloud governance, we built a simple model that predicts capacity management incidents before they occur [17]. In a case study with a commercial cloud provider, we could predict about 63% of the incidents (like insufficient disk space and memory leaks), amounting to 1.9 M Euro cost savings per year.

We could measure that optimizing Web queries can lead to 25% of energy savings, and putting to sleep application servers can reduce energy consumption by 8%. As queries and application servers are at the bulk of our 'connected world, you can do the math of how many queries are issued every second, and how much energy we could save globally with this 25% or 8%.

A survey taken by 650 KMPG executives in 16 different countries [5] found that one-third of the respondents discovered that the costs of cloud migration were higher than expected. Most critical reasons being lack of in-house skills and complexities in integrating with legacy software. Architectural tactics and design patterns can be a powerful instrument to aid professionals in making the right design decisions on which cloud model to adopt, and how to modernize a software system to deliver the same or improved functionality. The cost of bad decisions in cloud migration can be dearly: the European Union Digital Agenda estimates that private and public cloud migration could double the EU cloud sector's current revenues to nearly 80 billion by 2020 [9].

A condition to achieve these goals is the availability of reusable knowledge in terms of verified software practices, along with suitable metrics and reference measures. Only when this knowledge will be built we can speak of green labels and benchmarks that provide concrete guidance.

The software engineers of the past have been trained with the idea that they only had to take into account technological and economic factors: does the product work properly, and is it profitable? But the professionals of today must also be educated with the knowledge of sustainability, so that they can meet the challenges of the future.

Marc Andreessen was the co-author of Mosaic, the first graphical web browser for those that are old enough to remember it. But he is also investor; co-founder of Netscape and of venture capital firm Andreessen-Horowitz, and in the board of directors of Facebook, eBay, and HP, among others.

He said software is eating the world. Sectors of the economy get transformed into coding problems. Today, the world's largest bookseller, Amazon, is a software company. Toyota (not to mention Tesla) sells computers-on-wheels. DHL is a software network that happens to have trucks, planes and distribution hubs attached. Disney had to buy Pixar Animation Studios to survive.

The IoT revolution is turning into the software revolution: it is not anymore about attaching devices to the Internet. High Tech companies are taking over the giants in any other sector, from manufacturing to transportation, and government.

As Andreessen wrote: Over the next 10 years, I expect many more industries to be disrupted by software. Being able to design for sustainable software will determine the success, and longevity, of these industries. To this end, major fundamental research is necessary.

Closing

My inaugural speech is coming to its end. But before I would like to thank the many people that over the years supported me in this journey. Too many to mention them all, so bear with me on this.

When more than 13 years ago I have decided to move to the Netherlands, I could choose any university, any place. I chose the VU and I have never regretted it. Our department combines outstanding research quality, interesting education, innovation, and most importantly holds a culture of collaboration and informal dialogue that creates a force to be reckoned with. This is expressed in many initiatives, like the Network Institute and Amsterdam Data Science, and the growing collaboration with the colleagues and students of the University of Amsterdam.

I extend my gratitude to the management team of the Computer Science department, the Faculty of Exact Sciences, and the board of the VU University, who have given me the opportunity to work in this stimulating environment, and contribute to the research pillar on Science for sustainability for the connected world.

I am vary grateful for the researchers that joined my research group over the past few years: we share the same passion for combining academic research and industrial relevance, and they inspire me every day with their ideas, energy, and warmth.

I would also like to express my gratitude to prof. Jan Bosch (Chalmers University of Technology), prof. Arie van Deursen (TU Delft) and Jannie Minnema (Oracle), who enthusiastically accepted to share their inspiring perspectives in the symposium this afternoon on Software Engineering Reflections for Practice and Research.

The same goes for our students, in particular in the CS Master Track 'Software Engineering and Green IT and Information Sciences: you are our ambassadors in industry; your skills and ideas will shape our futures.

Not to be forgotten are the many industrial partners that have trusted me and my team with their research questions, their time and financial support. Among them, I especially praise the significant collaboration with CGI, IBM, ArchiXL; the partners of the Cluster Green Software, Greening the Cloud, and the upstarting project GreenServe; and the Sustainability Urban Lab collaboration with Omala and Green IT Amsterdam.

In particular, I want to mention two initiatives that are very dear to me. The first is VERSEN, the new Dutch national association for Software Engineering, which has just started and finally brings together our national Software Engineering community. The second is VHTO (the Dutch National Expert Organization on Women and Science/Technology) and Digivita, a programme introducing girls to computer science. I want to thank everybody that contributed to this programme in the occasion of this inauguration.

I also want to thank few people in particular. The first (in order of appearance) is professor Letizia Jaccheri, my PhD supervisor de-facto, a colleague but most of all a friend. We share the same passion for research. I miss our long discussions during our many games of cards, or in front of bread, *salame* and a good glass of Chianti.

The second is professor emeritus Hans van Vliet: approaching him almost 13 years ago has been one of the best professional decisions I have ever made. The things he taught me are countless, and the more evident to me now that he is enjoying his well-deserved retirement.

Steve Jobs once said: "The only way to do great work is to love what you do. But a rewarding career is nothing if you cannot share it with the ones you love. So, my final words are for my family.

My parents (who taught me the freedom to choose my own path), and especially my mother (a role model in a generation when working mothers were the exception). My Dutch family, that welcomed me as their own, and helped me understand and integrate in this Country. But most of all, my greatest supporters, Hans and Peter, who give me energy, every day.

Ik heb gezegd.

References

[1] Definition of Environmental Sustainability. http://www.thwink.org/sustain/glossary/EnvironmentalSustainability.htm.

[2] Digital Agenda for Europe: Cloud Computing. http://ec.europa.eu/digital-agenda/en/cloud.

[3] Environmental sustainability definition. http://www.triplepundit.com/2015/08/germany-became-solar-superpower.

[4] Light Up the Dark – Uppsala. http://www.philips.com/a-w/innovationandyou/article/extended-story/uppsala.html.

[5] The Cloud Takes Shape. www.kpmg.com/FR/fr/IssuesAndInsights/ArticlesPublications/Documents/the-cloud-takes-shape.pdf.

[6] Smart Cities Will Include 10 Billion Things by 2020. https://www.gartner.com/doc/3004417/smart-cities-include--billion, 2015.

[7] Financial Times Lexicon. Definition of Business Sustainability. http://lexicon.ft.com/Term?term=business-sustainability.

[8] Gary Cook And. *Clicking Clean: A Guide to Building the Green Internet*. Greenpeace, May 2015.

[9] Gary Cook and Tom Dowdall and David Pomerantz and Yifei Wang. *Clicking Clean: How Companies are Creating the Green Internet*. Greenpeace, April 2014.

[10] GeSI: Global e-Sustainability Initiative. SMARTer2030 – ICT solutions for 21st century challenges. page 134, 2015.

[11] Lorenz M Hilty and Bernard Aebischer. Ict for sustainability: An emerging research field. In *ICT Innovations for Sustainability*, pages 3–36. Springer, 2015.

[12] Patricia Lago. Challenges and opportunities for sustainable software. In *Proceedings of the Fifth International Workshop on Product LinE Approaches in Software Engineering. ICSE Companion*, pages 1–2. IEEE Press, 2015.

[13] Patricia Lago, Sedef Akinli Kocak, Ivica Crnkovic, and Birgit Penzenstadler. Framing sustainability as a software quality property. *CACM*, 58(10):70–78, October 2015.

[14] Merriam-Webster. Definition of sustainability. http://www.merriam-webster.com/dictionary/sustainable.

[15] Andrew Moore, Tim O'Reilly, Paul D. Nielsen, and Kevin Fall. Four thought leaders on where the industry is headed. *Software, IEEE*, 33(1):36–39, Jan 2016.

[16] G Procaccianti, P Lago, A Vetrò, D Méndez Fernández, and RJ Wieringa. The green lab: Experimentation in software energy efficiency. In *Proceedings of the 37th International Conference on Software Engineering (ICSE)*, 2015.

[17] Jack Schilder. Improving Capacity Management. Master's thesis, VU University Amsterdam, the Netherlands, 2010.