Position Paper: The Social Dimension of Sustainability in Requirements Engineering

Timo Johann
Department of Informatics, MOBIS
University of Hamburg
Hamburg, Germany
johann@informatik.uni-hamburg.de

Walid Maalej
Department of Informatics, MOBIS
University of Hamburg
Hamburg, Germany
maalej@informatik.uni-hamburg.de

***Abstract*— Nowadays mobile phones and smartphones are common not only in mega cities in developed countries but also in rural areas in emerging and developing countries. Technological progress will enable more and more people from different socio-cultural backgrounds and with different needs to use software services with up-to-date technical devices. End users are increasingly expecting to use software services in their "own way". They expect to customize the functionality, contribute to the content, and share information with others. The requirements of users differ drastically depending on the socio-cultural context. This is why we must raise the following questions: What is the impact of a society or an economy on software and vice versa? Do we need new approaches in requirements engineering to deal with the social dimension of sustainability? In this paper we aim to bringing forth a discussion about social sustainable software.**

Keywords-component; sustainability; requirements engineering; social sustainable software, sustainable informatics

I. The Lack of Social Aspects in Requirements Engineering

Definitions of sustainability are mainly based on three pillars, the economical, ecological, and social: the so-called triple bottom line [1]. Software systems have an influence on each one of the three pillars [2] [3] [4]:

- They changed economical processes and play a central role in our globalized economical system.
- It has been shown that they can have positive or negative impacts on the environment.
- The way we communicate and socialize today is based on software systems.

We claim that Sustainable Informatics research must take into account *all* aspects of sustainability, in particular the social aspect, which has been neglected so far.

Looking at recent publications in the field of Social Informatics, we found that social aspects are only mentioned in passing, while the clear focus has been placed on environmental aspects. The Sustainable Informatics community is developing processes, models, methods, and tools, mainly to minimize resource and energy consumption. Figure 1 depicts the most frequent terms of the proceedings of the last RE4SuSy workshop in order of their occurrence [5]. The 'biggest', i.e., most frequently used term is "energy". Terms such as "social", "user", "community", and "human factors" do not occur at all.

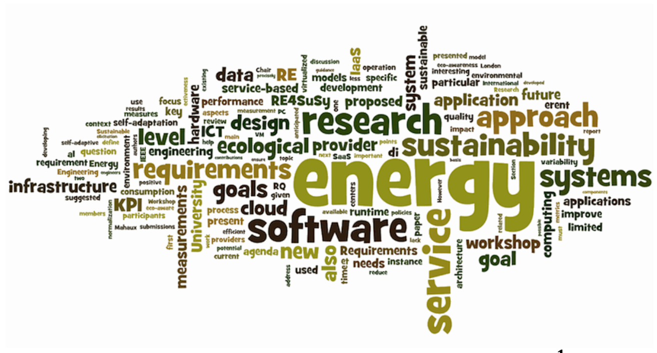

Figure 1: Wordle of the last RE4SuSy Workshop[1]

We think that the inclusion of users and their communities in the engineering processes has a high potential to support sustainable software engineering [6] [7]. The aim of this paper is to initiate a discussion about social sustainability in requirements engineering towards providing a foundation for holistic sustainable software systems.

Requirements engineering for sustainable systems must take into account all sustainability aspects, although these might seem contradicting. When searching for social or sustainable software we will find opposing views, ideas, and definitions. Sustainable software is often understood as software that is easy to maintain or reuse, without any reference to ecological or social aspects, e.g. in [8]. Tomlinson et al. define "*Sustainable software engineering aims to create reliable, long-lasting software that meets the needs of users while reducing environmental impacts*" [9]. In our opinion, this definition of sustainable software is incomplete and can be misleading. Naumann et al. [3] gave a definition that covers all aspects of sustainability in software. They define *Sustainable Software* as software "*whose direct and indirect negative impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software are minimal and/or which has a positive effect on sustainable development.*" This definition distinguishes between two types of sustainable software:

1. Software with positive impact on sustainable development.
2. Software that minimizes its own negative impact on sustainable development.

[1] Taken from http://www4.in.tum.de/~penzenst/re4susy/2013/

The second type of sustainable software should receive particular attention in requirements engineering, since we believe it applies to every software system. Sustainability should become a relevant concern for requirement engineering as a basic non-functional requirement.

Naumann et al. definition also shows that optimizing the impact of software is not enough. Impacts of development and deployment must also be associated with the software sustainability. Consequently the focus of future research in the field of social sustainable software should cover the users as well as the developers, their communities, and the interactions between them.

II. Examples of Social Impacts of Software

Social aspects of software involve many fields of interest that are directly or indirectly linked to social sustainability. In this section, we discuss a few examples of social sustainability in software. These examples can be used to discuss the characteristics for developing a uniform quality model for social sustainable software.

A. *Data Access and Transparency*

Freedom of information is a fundamental human right. 65 countries around the world have passed freedom of information laws and anchored them in their constitutions.

This positive development will facilitate participation in democratic processes. For example, in the field of E-Government, access to governmental data is often provided via software systems. Currently there are two basic problems in this field. First, the means to access these data are heterogeneous and often embedded in complex processes. Second, the access to data is unequal to access to information.

The Infrastructure for Spatial Information in the European Community (INSPIRE) provides a good illustration to these problems. INSPIRE is “an EU initiative to establish an infrastructure for spatial information in Europe that will help to make spatial or geographical information more accessible and interoperable for a wide range of purposes supporting sustainable development” [10].

A portal has being developed to grant access to the data [11]. The data contain, e.g., measures of water quality, air pollution, demographics, and historical water levels. These are stored in various formats such as PDF, CSV, or XML. Only few Europeans know about this project. Users with no expertise in IT or geographical information systems will encounter difficulties in finding specific information among this huge amount of data. A simple question like “Can I go swimming in the lake close to my hometown?” cannot be answered easily. This becomes even worse for questions like: “Where is the next source of clean drinking water?” Applying a system like this in rural areas in developing countries, where people rarely own computers, the requirements change drastically. When users only have mobile phones, they expect direct, location-based information about the surroundings. An excel file with complete historical data about the water quality of a river will not help much. In the current version of the system, users cannot easily add data and make it immediately accessible to others. However, this will be an important feature during natural disasters, when conditions drastically change within a short period of time.

B. *Reliability and Resilience*

Another question about the characteristics of social sustainable software is: How do software systems react to collapsing infrastructures? Tomlinson et al. showed that “understanding how to design ICT and sociotechnical systems […] enable social wellbeing in times of collapse could benefit many” [12]. They proposed the research field *Collapse Informatics.*

A collapse can occur globally or locally. A collapse is a “rapid, significant loss of sociopolitical complexity which in itself constitutes an event whose impacts exceed the responsive capacities of [those] affected” [13]. A collapse can be triggered among others by natural disasters.

Important software systems should be designed in a way that they are still available even in an unreliable infrastructure. Natural disasters create a great need for communication. Companies like Inveneo [2] specialize in quickly building broadband infrastructures, which were used e.g. after the earthquake in Haiti in 2010. With this infrastructure emergency and rescue forces, victims, family members, and friends were able to quickly connect to each other, share, and access important information, through the participation of a huge, globally connected community. Participants quickly organized themselves to support emergency teams on-site. People from all over the world contributed to the *Crisis Mapping* [14]. They traced roads and damaged buildings and entered camps of displaced people into OpenStreetMap. They gathered data from manifold sources and combined them with the help of *OpenStreetMaps*. This case shows that we “should also consider the design context to be a world radically altered by environmental damage. Solutions that fit into today’s lifestyles risk irrelevance” [15]. This case also provides evidence that requirements not only change depending on the socio-cultural context, but also that the socio-cultural context itself can change. Requirements engineering needs to be adjusted to these possible changes.

C. *Civic Participation and Software Socialness*

Civic participation is another aspect of involving social groups, where the influence of software is continuously growing. In recent years social media, like Facebook or Twitter have been used by the youth of different countries to form a protest in order to change the political system. Software can play an important role for enabling democratic processes.

Facebook, Twitter, Wikis, or Blogs are utilized for civic participation in political decision processes. Other software systems such LiquidFeedback [16] explicitly focus on political opinion formation. Every user can make a proper proposal, which can be supported, discussed and extended by others. This process provides a way to create a democratic image that is not distorted by hierarchies, discrepancy in

[2] http://www.inveneo.org/

knowledge or other constraints. The German Pirate Party, with elected members in several German regional parliaments, successfully uses this software. Other parties are currently running trials.

Also companies have an immense interest in user feedback to improve or even invent products and services. The SYNAXON AG is, e.g., a large company that uses LiquidFeedback for internal decision making and customer involvement. Vice versa, users have an interest in communicating their opinions. In requirements engineering we must take up this trend and find ways to build adequate systematic feedback methods [8]. This will help to gather requirements of a heterogeneous group of users and to meet the needs of people that will otherwise be out of scope.

An other essential approach that will support sustainable software is its *socialness*. Maalej and Pagano define the socialness of software systems "*as the degree of involvement of its users and their communities in the software lifecycle*" [6]. The involvement of users in the software life cycle can be an important step towards social sustainable software. This can be archived by *Social Software Engineering*. Social software engineering is "*the application of processes, methods, and tools to enable community-driven creation, management, deployment, and use of software in online environments*" [17]. Sustainable informatics can benefit from the ongoing research in the fields of social software engineering. A recently presented social software engineering process called SNAIL is an example for a mature solution in this field. SNAIL "*thoroughly and continuously involves users by establishing interaction channels and integrating user communities*" [6]. The challenge is to find ways to involve users from different backgrounds. Further analyses of the heterogeneous ways users are able and willing to communicate their requirements and feedback is necessary. Furthermore we have to study approaches to conflate the feedback and make it accessible in a standardized way.

D. Accessibility

Accessibility, especially for handicapped users is an important concern. Many countries have laws that guarantee direct accessibility for everybody. In software engineering there are no uniform rules for whether and how accessibility is implemented. The majority of modern operating systems are equipped with integrated accessibility options like screen readers, display zooms or specialized color settings. The W3C encourages web developers to build barrier-free websites and gives out the *Web Content Accessibility Guidelines* [18]. These are limited to the physical conditions of individuals. It is however important to bear in mind that accessibility requirements differ depending on the social environment, and the social environments are responsible for access conditions.

One major problem is the digital gap between developed countries and emerging and developing countries. In some rural areas, people have very limited access to communication services. Internet is often slow or not available at all. Devices are usually old. An example for the inclusion of specific requirements is *Esoko*, a software that provides market information, mainly from commodities markets. Although this example does not sound innovative at first, it shows how information has been made accessible in specific social environments. Users can sign up via SMS or E-Mail to receive prices of the world markets by SMS. This simple 'innovation' granted thousands of farmers in rural areas in Africa access to important information. This made farmers no longer dependent on resellers. They can make better decisions about selling their goods for the offered prices. A survey conducted by the French National Institute for Agricultural Research (INRA) found that maize, groundnut, and cassava farmers, recorded a 10% increase in revenues after receiving and utilizing the Esoko [19].

E. Privacy, Safety and Security

Privacy is a major concern in the information age. It is part of a worldwide public discussion. Opinions diverge widely and the topic has become crucial for governments, companies, lawyers, and software engineers. Especially when it comes to surveillance by authorities, the issue becomes highly problematic. In many countries police and intelligence use surveillance and monitoring software for crime protection. Most countries adopted strict laws for the use of said software. For example, in Europe a court order is required prior to the utilization of monitoring software. Private companies with commissions of governments develop monitoring software. In the past, this software has also fallen into the hands of repressive regimes. An example is the surveillance software *FinFisher*. The tool was developed and marketed by a German and British subsidiary of the *Gamma Group*. During the Egyptian Revolution of 2011, dissidents discovered a contract with Gamma International for €287,000 for a license to run FinFisher [20]. Last year, FinFisher was found on the computer of human rights activists in Bahrain. Citizen Lab, based at the University of Toronto, confirmed that the Trojan was installed on the laptops of several journalists and activists. Gamma Group denies that they sold the software to these regimes. In February 2013 a consortium of different NGO (Privacy International, European Centre for Constitutional and Human Rights, the Bahrain Center for Human Rights, Bahrain Watch, and Reporters without Borders) officially filed a complaint with the OECD [21]. Even if this example primarily seems to be a political issue, it also demonstrates the importance of the socio-cultural context, when it comes to requirements of software systems.

III. Research Agenda

There is a need to discuss whether requirements engineering for sustainable systems is different from traditional requirements engineering. For this discussion we should first identify the fundamental characteristics of social sustainable software.

Requirements engineers and analysts are still missing the overall scope of social sustainable software. Social aspects are already scattered across many fields of interest that are directly or indirectly linked to social sustainability. We think that there is a lack of the social aspect of sustainability in the current research on Sustainable Informatics and propose a further discussion of the topic.

First the community should define the characteristics that support the development of a **uniform quality model for social sustainable software**.

This requires extracting different examples of software that affect social sustainability. Based on these examples we can derive basic characteristics of social sustainable software. By now functional as well as quality requirements often lack awareness of the socio-cultural context. We need to include the socio-cultural context. This might play an important role for the requirements engineering of sustainable systems.

Software systems are usually developed in the "western world". Software engineers are running the risk of being unaware of the manifold conditions under which their software is used. We think **further studies** are needed to identify ways how **users can participate in requirements and software engineering processes** Therefore we must apply modern approaches that involve users and their communities, e.g., as suggested in [6].

Furthermore, field studies in ongoing and future software projects in unconventional domains represent a way to reveal insights and help us to learn more about requirements in different socio-cultural contexts. The knowledge about possible conditions will also help us in developing a quality model for social sustainable software.

In order to collect more information about sustainability in software systems we propose the development of a database. The purpose of this database is to create an overview of software with positive or negative, direct or indirect impacts on society as well as the context in which these impacts occur. Interested parties can access the data and add entries to the database. We believe that the collected data will help us to refine requirements for sustainable systems over time. As a result we hope to get enhanced insights into the social aspects of software.

A long term future goal is to develop recommender systems for software engineers and analysts. These systems will foster the awareness of possible impacts of software and thereby support the development of more sustainable software.

IV. References

[1] UN, "Report of the World Commission on Environment and Development. Our common future. UN document no. A/42/427," New York, 1987.

[2] L. Hilty, "Information technology and sustainability. Essays on the relationship between ICT and sustainable development.," Books on Demand, Norderstedt, 2008.

[3] S. Naumann, M. Dick, E. Kern and T. Johann, The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering, Sustainable Computing: Informatics and Systems, 2011.

[4] B. Tomlinson, Greening through IT: Information Technology for Environmental Sustainability, MIT Press, 2010.

[5] "Proceedings of the First International Workshop on Requirements Engineering for Sustainable Systems," in Requirements Engineering: Foundation for Software Quality, Essen, 2012.

[6] W. Maalej and D. Pagano, On the Socialness of Software, 21st. IEEE International Requirements Engineering Conference ed., IEEE.

[7] W. Maalej and D. Pagano, "User feedback in the AppStore: An Empirical Study.," in 21st. IEEE International Requirements Engineering Conference, 2013.

[8] K. Tate, Sustainable Software Development, Addison-Wesley , 2005.

[9] N. Amsel, Z. Ibrahim, A. Malik and B. Tomlinson, "Toward sustainable software engineering," in 33rd International Conference on Software Engineering (ICSE), Honolulu, 2011.

[10] K. Benoit, Infrastructure for Spatial Information in the European Community, Cede Publishing, 2011.

[11] "Inspire Geoportal," European Commission, [Online]. Available: http://inspire-geoportal.ec.europa.eu/. [Accessed 25 April 2013].

[12] B. Tomlinson, S. M. Silberman, D. Petterson , Y. Pan and E. Blevis, "Collapse informatics: augmenting the sustainability & ICT4D discourse in HCI," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, 2012.

[13] M. a. T. B. Silberman, "Precarious infrastructure and postapocalyptic computing.," in Examining Appropriation, Re-use, and Maintenance for Sustainability, workshop at CHI 2010, 2010.

[14] P. Meier, "nationalgeographic.com," National Geographic, 2012. [Online]. Available: http://newswatch.nationalgeographic.com/2012/07/02/crisis-mapping-haiti/. [Accessed 24 4 2013].

[15] J. Wong, "Prepare for Descent: Interaction Design in our New Future.," in Defining the Role of HCI in the Challenges of Sustainability, workshop at CHI , 2009.

[16] "liquidfeedback.org," Verein zur Förderung des Einsatzes elektronischer Medien für demokratische Prozesse, [Online]. Available: http://liquidfeedback.org/. [Accessed 20 April 2013].

[17] I. Hammouda, J. Bosch, M. Jazayeri and T. Mikkonen, "1st International Workshop on Social Software Engineering and Applications," in Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering, 2008.

[18] W3C, "Web Content Accessibility Guidelines (WCAG) 2.0," W3C, 2008.

[19] "http://www.esoko.com/," INRA, 15 12 2011. [Online]. Available: http://www.esoko.com/about/news/pressreleases/2011_15_12_Esoko_INRA.pdf.

[20] "f-secure," F-Secure Lab, [Online]. Available: http://www.f-secure.com/weblog/archives/00002114.html. [Accessed 25 April 2013].

[21] "privacyinternational.org," Privacy International, [Online]. Available: https://www.privacyinternational.org/blog/our-oecd-complaint-against-gamma-international-and-trovicor. [Accessed 22 April 2013].