Jul 1st, 12:00 AM

Design and Development of S.O.L.E. Software

Daryl H. Hepting

International Environmental Modelling and Software Society (iEMSs)
2010 International Congress on Environmental Modelling and Software
Modelling for Environment's Sake, Fifth Biennial Meeting, Ottawa, Canada
David A. Swayne, Wanhong Yang, A. A. Voinov, A. Rizzoli, T. Filatova (Eds.)
http://www.iemss.org/iemss2010/index.php?n=Main.Proceedings

Design and Development of S.O.L.E. Software

Daryl H. Hepting [a]

[a]Department of Computer Science
University of Regina
Regina, Saskatchewan
S4S 0A2, CANADA

Abstract: The acronym SOLE (Sustainable Organic Local Ethical) has been applied to food, for which it signifies the highest standard. In an age when sustainability has become critically important in so many human endeavours, this paper examines the question of whether SOLE can be meaningfully used to signify software of the highest standard. This paper examines how it might be possible to think of software as sustainable, organic, local, and ethical. Should environmental software, in particular, strive for this high standard? Design and development of SOLE software are also considered.

Keywords: sustainability; ethics; software development

1 INTRODUCTION

The S.O.L.E. (Sustainable Organic Local Ethical) acronym was coined as a means to articulate a higher standard for food. Food production must be sustainable, if we are to meet the needs of a growing global population. In 1987, the Brundtland Commission defined sustainability (S) as that which "meets the needs of the present without compromising the ability of future generations to meet their own needs." Reliance on petroleum-based inputs for food production is not sustainable in this era of "peak oil." Organic (O), or ecological, production is preferred over industrial, or intensive, production. Mindful of the impact of transportation of over large distances, food should be produced for local consumption (L). Finally, it is also important to consider the ethical (E) treatment of farmers and farm workers - which has been expressed in campaigns such as fair trade [Hira and Ferrie, 2006]. These concepts are intertwined, and together they provide a fuller definition of sustainability, as well as a prescription for moving towards it.

For food production, S.O.L.E. may embody the ecological approaches to agriculture which "seek not so much to increase output or yield as to identify and moderate production processes that are optimal within a set of geographical and historical constraints" [Lyson, 2005]. There are ecological approaches to software, as well. For example, ecological interface design [Vicente, 2002] seeks to "encourage the use of skill- and rule-based behaviour (to save on scarce cognitive resources) while providing support for otherwise more effortful and error-prone knowledge-based behaviour (to cope with unfamiliar and unanticipated situations requiring adaptive problem solving)." In both cases, the optimal focuses on people in their local context.

Does S.O.L.E. apply to production or use, process or end-product? Some may choose S.O.L.E. food because of its freshness [Svenfelta and Carlsson-Kanyamab, 2010], but that does not devalue the other properties of the food. Similarly, if one chooses S.O.L.E. software because of its ease of use, the methods of software design and development are still important. Can something produced in a S.O.L.E.-ful way be used for ends which are not? Collins et al. [1994] talks about the 1988 downing of an Iranian airline. Could combat control system software, like that involved in this tragic inciden, ever be considered as S.O.L.E. software?

Not all software is created equal, just like one egg may not be just like any other. The process, the lifecycle, is important. A usable software interface cannot be created just before the product is delivered, nor can vitamins be added to food as an afterthought.

S.O.L.E. does not exist in a vacuum: just as food must be sold, so too must software consider economics [Boehm and Sullivan, 2000] and the concept of value. Yet food and software are not directly comparable. Food is material whereas software is not. Food production is dependent on location: oranges cannot be cultivated everywhere, although modern food distribution systems seek to hide this fact. Software production is not dependent on location per se, but locality is important.

Biodiversity in food crop varieties is being sacrificed for the convenience of mass production. Cultural differences, embodied in localized processes, can be lost in a rush to standardization. Conventional and organic agriculture distribute knowledge differently: the former moves information towards input suppliers latter moves information towards the farmers Morgan and Murdoch [2000], which can help to repair cultural losses.

S.O.L.E. is complimentary to McDonough and Braungart's [2002] concept of the triple top line. In contrast to Elkington's [1999] triple bottom line which strives for eco-efficiency in order to minimize harm, the triple top line seeks eco-effectiveness in order to maximize benefit. That which is not sustainable, organic, local, nor ethical should only be tolerated until replacements with those properties emerge.

The rest of this paper is organized as follows. Section 2 discusses how the adjectives sustainable, organic, local, and ethical may be applied to software. Section 3 discusses how these ideas can be applied to design. Section 4 discussed how these ideas can be applied to development. Section 5 presents these ideas in the context of other relevant statements about software.

2 S.O.L.E. SOFTWARE

In terms of software, practitioners can look to the ACM/IEEE-CS Software Engineering Code of Ethics and Professional Practice[1] for guidance about what should be created and how to achieve it [Gotterbarn and Miller, 2009]. It contains the following 8 principles:

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

[1] http://www.acm.org/about/se-code

Public interest is of paramount importance in this code, yet it may not be so easy to discern in practical situations. It is hard to imagine an issue of greater public interest than addressing the challenges of a world with 9 billion inhabitants [Dearden et al., 2010].

It is difficult to infer how the code should be manifested in software design and development. Business ethics education has benefitted from the use of exemplars [DeVries, 1986]. Yet, Giacalone and Thompson [2006] point out that it is not enough to teach ethics if it is done within a framework that discourages ethical behaviour. In Computer Science, a class on ethics has become a necessary part of the curriculum, but are ethics supported throughout the program with an appropriate worldview? Consider the example of blippy.com, which "is a fun and easy way to see and discuss what everyone is buying" [2]. Since its inception and through the recent incident where credit card numbers of subscribers became available via internet search [3], the discussion of ethics around such an enterprise has been absent. Computer Science students may still cling to the idea that ethics is optional: if they do not belong to the ACM or the IEEE, they do not need to be bound by ethical standards.

Even without reference to the ACM/IEEE-CS Software Engineering Code of Ethics and Professional Practice, it is clear that it is possible to think of software in terms of ethics. However, sustainable, organic, and local equally applicable when describing software. Each of them are explored further here. The concepts are intertwined.

2.1 Sustainable software

Sustainable software might support unsustainable behaviours. If the software in question is "free", Chopra and Dexter [2009] point out that the Free Software Definition does not prescribe the uses of software. The Free Software Definition encompasses four freedoms:

0. The freedom to run the program, for any purpose.
1. The freedom to study how the program works, and adapt it to your needs . Access to the source code is a precondition for this.
2. The freedom to redistribute copies so you can help your neighbour.
3. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.

In one sense, sustainable software is maintainable software [Aggarwal et al., 2002]. More generally, we might consider 4 spheres of sustainability: environmental, economic, social and political [Maxim et al., 2009]. Liu et al. [2008] describe how the need for the effective integration of science and decision-making is ubiquitous in environmental management. Software that does not meet the needs of its users, by making information available in an effective and actionable way, is not sustainable. Sustainability science and engineering [Mihelcic et al., 2003; Komiyama and Takeuchi, 2006] will rely more and more on software. Software used to support sustainable behaviours is certainly important, and this software should be sustainable. But in keeping with the Free Software Definition, sustainable software need not be used in service of sustainability. For many years, computer science has downplayed the importance of power management but this is beginning to change [Ranganathan, 2010].

2.2 Organic software

The internet browser Firefox[4] is billed as 100% organic software:

[2] http://blippy.com
[3] http://mashable.com/2010/04/23/blippy-credit-card-numbers/
[4] `http://www.mozilla.com/firefox`

> What is organic software?
>
> As software companies go, we're a little unusual. We use the term organic software to sum up the various ways we're different from the other guys:
>
> Our most well-known product, Firefox, is created by an international movement of thousands, only a small percentage of whom are actual employees.
>
> Were motivated by our mission of promoting openness, innovation and opportunity on the web rather than business concerns like profits or the price of our stock (guess what: we dont even have stock).
>
> And as a non-profit, public benefit organization, we define success in terms of building communities and enriching peoples lives. We believe in the power and potential of the Internet and want to see it thrive for everyone, everywhere.

We may also talk about Organic Computing [Würtz, 2010], which holds that "it is scientifically fruitful to interpret complex systems of interacting processes as computational systems and to study them as such."

To follow on the mozilla definition, organic software is grown from the grassroots, bottom up rather than top down. The term 'ecology' also appears in this discussion [Healy and Schussman, 2003]. Therefore, it may also be local.

2.3 Local software

Software is written for users, who may have local knowledge. The preservation of that local knowledge, through appropriate software supports, is vitally important. The processes of internationalization and localization require more effort than simple language changes.

2.4 Ethical software

The key focus of the ACM/IEEE-CS code of ethics and professional practice for software engineers is the public good. Information is a good whose value increases with sharing [Kubiszewski et al., 2010]. This concept is well-aligned with the Free Software Definition. Yet software may also facilitate ethical behaviour, as in the case of information transparency [Turilli and Floridi, 2009].

Collins et al. [1994] discuss obligations of 4 stakeholder groups in terms of software construction and use: provider, buyer, user, and the penumbra (those affected by the software's use). To the penumbra, all other groups are obliged to safeguard against harm and reduce risks. To the user (for example), the software buyer must be prudent about automation. To return to the airliner tragedy, S.O.L.E. software would have minimized the risk of such a serious error and not automated so much of the workflow that the user was not certain about the exact nature of the threat that was faced.

The imprudent use of automation is seen more abstractly in the case of food: because more food is provided to the consumer without need of extensive preparation, that knowledge for users is diminished.

3 DESIGN

The ACM/IEEE-CS code contains the following detailed statement: "3.03. Identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects." This, and others, may be much easier said than done. Economic, legal, and environmental issues relate to the sustainable property; cultural issues may relate to organic and local properties; and

ethical relates to the ethical property. Participatory design that is culturally sensitive may be employed [Bidwell and Winschiers-Theophilus, 2010; Winschiers-Theophilus, 2010; Winschiers and Paterson, 2004], but is enough consideration paid to the standardization of processes that would be imposed? Although one might like to think about designing solutions that are localized, there is pressure to create solutions that can be mass produced and widely distributed with minimal customization, perhaps in the name of 'best practices' [Wagner et al., 2006].

4 DEVELOPMENT

Herman Daly [1997] talks about development in terms of refinement and not just growth. Is it possible to think of software development in the same way? Whereas it has been commonplace to get bigger and better servers to handle increased loads, relatively little effort has been spent developing algorithms and systems that use more power-aware approaches and that augment, rather than automate away, the human's contributions. McConnell [2004] provides an important foundation for various strategies for dealing with inevitable change within software. Yet, he does not deal with the important elements of openness and freedom that are important to S.O.L.E. software.

As a more inclusive model for development, we may consider Fischer's Seed-Evolve-Reseed (SER) approach which attempts to value input from community members while balancing a consistent product [Fischer, 2005].

5 DISCUSSION

The ACM/IEEE-CS Code of Ethics and the Free Software Definition both provide direction in terms of what software should be, and how it should be produced. Perhaps a more effective means of communicating elements of practice is to use cases involving real software. Just as with food, there are likely very few ways to reach the highest standard.

"Web Search for a Planet" [Barroso et al., 2003] describes google's pragmatic approach to efficiency. Can we work towards 'software development for a planet'?

There is the opportunity, within software development, to enable a higher standard of ethical behaviour. A recommendation following the theft of e-mails from East Anglia CRU called for greater transparency. S.O.L.E. software can meet this call, for example: sustainable software is designed for change so that it can be updated easily and it empowers its users to understand the information they receive; organic software is developed methodically from the bottom up; local software provides appropriate localized information, in the appropriate ways; and ethical software makes the information and computational processes transparent.

Why might environmental software be held to a higher standard? Returning to Collins et al. [1994], consider the size of the penumbra cast by environmental software: it includes all inhabitants of the planet. Towards the penumbra, the software developer is obliged to be open about the software development process and the limits of correctness; the software buyer is obliged to be open about the software capabilities and limitations; and the user is obliged to conscientious in reducing risks to the public and encourage reasonable expectations about the software capabilities and limitations.

S.O.L.E. can be appropriately applied to software. It may be an effective shorthand for describing desired outcomes of software design and development.

ACKNOWLEDGMENTS

The author gratefully acknowledges the support of the Centre for Sustainable Communities at the University of Regina and the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

Aggarwal, K. K., Y. Singh, and J. K. Chhabra. An integrated measure of software maintainability. *Proceedings of the Reliability and Maintainability Symposium*, pages 235–241, Feb 2002.

Barroso, L. A., J. Dean, and U. Hölzle. Web search for a planet: the google cluster architecture. *IEEE Micro*, 23(2):22–28, Jul 2003.

Bidwell, N. and H. Winschiers-Theophilus. Under development beyond the benjamins: toward an african interaction design. *interactions*, 17(1):32–35, 2010.

Boehm, B. and K. Sullivan. Software economics: a roadmap. *Proceedings of the conference on The future of Software engineering*, page 343, 2000.

Chopra, S. and S. Dexter. The freedoms of software and its ethical uses. *Ethics Inf Technol*, 11 (4):287–297, Dec 2009.

Collins, W., K. Miller, B. Spielman, and P. Wherry. How good is good enough?: an ethical analysis of software construction and use. *Communications of the ACM*, 37(1):91, 1994.

Daly, H. *Beyond Growth: The Economics of Sustainable Development*. Beacon Press, 1997.

Dearden, A., A. Light, R. Heeks, and G. Marsden. Innovation everywhere: Computing for 9 billion people, Feb 2010.

DeVries, P. The discovery of excellence: The assets of exemplars in business ethics. *Journal of Business Ethics*, 5(3):193–201, 1986.

Elkington, J. Triple bottom line: Implications for the oil industry. *Oil and Gas Journal*, Jan 1999.

Fischer, G. Distances and diversity: Sources for social creativity. *Proceedings of the 5th conference on Creativity & cognition*, pages 128–136, Jan 2005.

Giacalone, R. and K. Thompson. Business ethics and social responsibility education: Shifting the worldview. *Academy of Management Learning and education*, 5(3):266–277, 2006.

Gotterbarn, D. and K. W. Miller. The public is the priority: Making decisions using the software engineering code of ethics. *IEEE Computer*, (June):66–73, Aug 2009.

Healy, K. and A. Schussman. The ecology of open-source software development. *Unpublished manuscript, January*, 29:2003, 2003.

Hira, A. and J. Ferrie. Fair trade: Three key challenges for reaching the mainstream. *Journal of Business Ethics*, Jan 2006.

Komiyama, H. and K. Takeuchi. Sustainability science: building a new discipline. *Sustainability Science*, 1(1):1–6, 2006.

Kubiszewski, I., J. Farley, and R. Costanza. The production and allocation of information as a good that is enhanced with increased use. *Ecological Economics*, Jan 2010.

Liu, Y., H. Gupta, E. Springer, and T. Wagener. Linking science with environmental decision making: Experiences from an integrated modeling approach to supporting sustainable water resources management. *Environmental Modelling & Software*, 23(7):846–858, 2008.

Lyson, T. A. Civic agriculture and community problem solving. *Culture and Agriculture*, 27(2): 92–98, Nov 2005.

Maxim, L., J. Spangenberg, and M. O'Connor. An analysis of risks for biodiversity under the dpsir framework. *Ecological Economics*, 69:12–23, Jan 2009.

McConnell, S. *Code Complete*. Microsoft Press, 2nd edition, 2004.

McDonough, W. and M. Braungart. Design for the triple top line: new tools for sustainable commerce. *Corporate Environmental Strategy*, 9(3):251–258, 2002.

Mihelcic, J., J. Crittenden, M. Small, D. Shonnard, D. Hokanson, Q. Zhang, H. Chen, S. Sorby, V. James, and J. Sutherland. Sustainability science and engineering: The emergence of a new metadiscipline. *Environ. Sci. Technol*, 37(23):5314–5324, 2003.

Morgan, K. and J. Murdoch. Organic vs. conventional agriculture: knowledge, power and innovation in the food chain. *Geoforum*, 31:159–173, Feb 2000.

Ranganathan, P. Recipe for efficiency: principles of power-aware computing. *Communications of the ACM*, Jan 2010.

Svenfelta, A. and A. Carlsson-Kanyamab. Farmers' markets - linking food consumption and the ecology of food production? *Local Environment*, 15(5):453 – 465, May 2010.

Turilli, M. and L. Floridi. The ethics of information transparency. *Ethics and Information Technology*, 11(2):105–112, 2009.

Vicente, K. Ecological interface design: Progress and challenges. *Human factors*, 44(1):62, 2002.

Wagner, E., S. Scott, and R. Galliers. The creation of 'best practice' software: Myth, reality and ethics. *Information and Organization*, 16(3):251–275, 2006.

Winschiers, H. and B. Paterson. Sustainable software development. *Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 274–278, 2004.

Winschiers-Theophilus, H. The art of cross-cultural design for usability. *Universal Access in Human-Computer Interaction. Addressing Diversity*, pages 665–671, 2010.

Würtz, R. Introduction: Organic computing. *Organic Computing*, pages 1–6, 2010.