Conceptual modeling of goals and metrics to visualize the measurement of sustainability

Birgit Penzenstadler
Department of Computer Engineering
and Computer Science
California State University Long Beach
Long Beach, California, USA
birgit.penzenstadler@csulb.edu

***Abstract*—This keynote first presents a couple of approaches from past and current work in visualizing and breaking down high-level sustainability goals into measurable goals and how to relate them to indicators and metrics, from both the field of software engineering as well as sustainable development. Then we will take a dive into other models that have been used to structure and visualize the relation from high-level goals and all the way down to detailed metrics and sketch out a few opportunities to make increased use of those in a research agenda for MegSuS.**

Keywords—sustainability, sustainability measures, software systems for sustainability, visualization, metrics.

I. Flourishing Instead of Fixing

According to the Oxford dictionary, sustainability is the ability to be maintained at a certain rate or level, as well as the avoidance of the depletion of natural resources in order to maintain an ecological balance.

Ehrenfeld [1] proposes to take that further, where sustainability (as an end) should be understood as the ability to flourish indefinitely (means). He points out that our current problem is that we are trying to reduce unsustainability, which is not creating sustainability. He adds that as a society, we seem to be addicted to solving our problems through a reductionist framework, which manifests itself in technology fixes that keep us trapped in the wrong path. We try to apply a technology fix to create more eco-efficiency, and due to rebound effects, unconsciousness, and even addiction (to our technology fixes), we remain in a state of unsustainability – instead of taking on a new paradigm and new mindset that would allow us (with a certain delay) to be able to transition onto a sustainable path. To get towards flourishing, Ehrenfeld proposes four steps:

1. Take ethical decisions based on **values**,
2. Develop collective **visions** of the future (outside of the old circular patterns),
3. Replace old structures and **strategies**, and
4. Live inside the **question**.

While all four steps require a big shift in thinking and major changes, they promise to lead us towards manifesting a transition that none of our technology fix routes has been able to achieve.

As researchers, we are in a position to lead the way for living inside the question, and we need to step into that responsibility. That means to not buy into the technology fix path, which is very tempting in our field of research. So the question is how we can go beyond this limited perception of making an IT solution more eco-efficient than it currently is, and instead contemplate a bigger picture. How can we, as software engineering researchers, be the facilitators of a larger joint vision of the future?

II. Scoping Sustainability

Tainter [2] proposes that in order to analyze the sustainability of something, we need to get very clear on the scoping, and answer the questions of:

1. What to sustain? What is the purpose of the system, or the mission behind it?
2. For whom? Who are the stakeholders? Are some of them maybe beyond organizational reach, being impacted but not considered during development?
3. For how long? A decade? A generation? A century? Can we think beyond standard business plan terms?
4. At what cost? What is the return on investment, and what are the environmental and social impacts?

Applying those four questions to a specific software system under consideration, we note that we can answer the first question on several levels, on a conceptual level or on a technical level. If we are to respond on a technical level, we go back to the technology fix loop. If we can respond on a conceptual level, we focus on the functionality or service that the system under consideration is supposed to support. The next question is how we can integrate that into the development of software-intensive systems.

III. Designing for Sustainability

The Karlskrona Manifesto on Sustainability Design was one of the first answers proposed on a conceptual level, before considering a specific system purpose and optimizing a technical solution. The manifesto makes observations about common misconceptions around sustainability and development towards sustainability and proposes a set of principles and commitments. These principles are [3]:

1. **Sustainability is systemic**. Sustainability is never an isolated property. Systems thinking has to be the starting point for the transdisciplinary common ground of sustainability.
2. **Sustainability has multiple dimensions**. We have to include those dimensions into our analysis if we are to understand the nature of sustainability in any given situation.
3. **Sustainability transcends multiple disciplines**. Working in sustainability means working with people from across many disciplines, addressing the challenges from multiple perspectives.
4. **Sustainability is a concern independent of the purpose of the system**. Sustainability has to be

considered even if the primary focus of the system under design is not sustainability.

5. **Sustainability applies to both a system and its wider contexts**. There are at least two spheres to consider in system design: the sustainability of the system itself and how it affects sustainability of the wider system of which it will be part.
6. **Sustainability requires action on multiple levels**. Some interventions have more leverage on a system than others. Whenever we take action towards sustainability, we should consider opportunity costs: action at other levels may offer more effective forms of intervention.
7. **System visibility is a necessary precondition and enabler for sustainability design**. The status of the system and its context should be visible at different levels of abstraction and perspectives to enable participation and informed responsible choice.
8. **Sustainability requires long-term thinking**. We should assess benefits and impacts on multiple timescales and include longer-term indicators in assessment and decisions.
9. **It is possible to meet the needs of future generations without sacrificing the prosperity of the current generation**. Innovation in sustainability can play out as decoupling present and future needs. By moving away from the language of conflict and the trade-off mindset, we can identify and enact choices that benefit both present and future.

If software engineers are to take those principles account while engineering the requirements and design of their systems, we will see a significant shift towards more sustainability in software-intensive systems development. The work on the manifesto includes developing methods that make their application straight-forward, and we have evaluated these methods in first industry studies [4].

IV. Transformation Mindset

The next question that arises is where our current solutions are in terms of maturity. Mann, Bates, and Maher [5] have analyzed the maturity of ICT (information and communication technology) solutions for sustainability and found that most solutions are in the area of compliance, e.g. to standards, and use in terms of user behavior. Instead, the authors argue, we need a sustainability-based transformation mindset. The transformation mindset includes ten propositions [6]:

1. Socioecological restoration **over economic justification**
2. Transformative system change **over small steps to keep business as usual**
3. Holistic perspectives **over narrow focus**
4. Equity and diversity **over homogeneity**
5. Respectful, collaborative responsibility **over selfish othering**
6. Action in the face of fear **over paralysis or willful ignorance**
7. Values change **over behavior modification**
8. Empowering engagement **over imposed solutions**
9. Living positive futures **over bleak predictions**
10. Humility and desire to learn **over fixed knowledge sets**

When faced with the challenge of how to bring such a mindset into practice, requirements engineering is the key activity within software-intensive systems development to affect change [6]. As proposed in several pieces of related work [7], an artifact-based approach to requirements engineering with a focus on sustainability as a first-citizen objective makes the contribution by and impacts of ICT for sustainability better tangible and visible.

V. Requirements Engineering for Sustainability

Requirements Engineering for Sustainability (RE4S) helps elicit and document requirements with a focus on analyzing the different dimensions of sustainability in the wider system context.

The first artifact breaking down the sustainability goals of the system is a dedicated goal model [8], organized according to different dimensions of sustainability, namely individual, social, environmental, economic, and technical. Then we analyze the potential impacts of the system for the dimensions and for the orthogonal three orders of effect [9], namely direct effects, enabling effects, and systemic effects. We can visualize a summary of such an analysis in a Sustainability Analysis Diagram (SusAD), as illustrated for a procurement system in [6], and for a resilient smart garden system in [10]. Figure 1 shows the empty template for a sustainability analysis diagram along with guidance for a first draft. Both the goal model and the sustainability analysis diagram allow for a tie-in with metrics to assess chosen interventions.

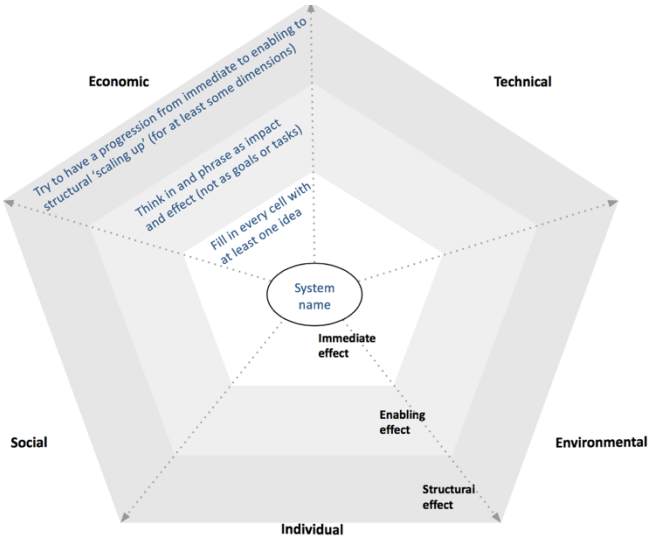

Fig. 1: Template for a sustainability analysis diagram.

VI. Conclusion

The complexity of sustainability asks for effective visualization of conflicts & contradictions. We can visualize goals to identify metrics and analyze effects to measure sustainability. Thereby, goal modeling helps visualize the vision, and a sustainability analysis diagram helps visualize the impacts and effects. Future work is under way towards best practice patterns.

References

1. Ehrenfeld, John R., and Andrew J. Hoffman. Flourishing: A frank conversation about sustainability. Stanford University Press, 2013.
2. Tainter, Joseph A. "Social complexity and sustainability." ecological complexity 3.2 (2006): 91-103.
3. Becker, C., Chitchyan, R., Duboc, L., Easterbrook, S., Penzenstadler, B., Seyff, N., & Venters, C. C. (2015, May). Sustainability design and software: The karlskrona manifesto. In Proceedings of the 37th International Conference on Software Engineering-Volume 2 (pp. 467-476). IEEE Press.
4. Seyff, N., Betz, S., Duboc, L., Venters, C., Becker, C., Chitchyan, R., Penzenstadler, B. and Nöbauer, M., 2018, August. Tailoring Requirements Negotiation to Sustainability. In 2018 IEEE 26th International Requirements Engineering Conference (RE) (pp. 304-314). IEEE.
5. Mann, Samuel, Oliver Bates, and Raymond Maher. "Shifting the maturity needle of ICT for Sustainability." Proceedings of the 5th Intl. Conference on Information and Communication Technology for Sustainability ICT4S (2018)
6. Becker, C., Betz, S., Chitchyan, R., Duboc, L., Easterbrook, S.M., Penzenstadler, B., Seyff, N. and Venters, C.C., 2016. Requirements: The key to sustainability. IEEE Software, 33(1), pp.56-65.
7. Penzenstadler, Birgit. "Infusing Green: Requirements Engineering for Green In and Through Software Systems." 3rd Intl. Workshop on Requirements Engineering for Sustainable Systems, RE4SuSy@ RE. 2014.
8. Penzenstadler, B., & Femmer, H. (2013, March). A generic model for sustainability with process-and product-specific instances. In Proceedings of the 2013 workshop on Green in/by software engineering (pp. 3-8). ACM.
9. Hilty, Lorenz M., and Bernard Aebischer. “ICT for sustainability: An emerging research field." ICT Innovations for Sustainability. Springer, Cham, 2015. 3-36.
10. Penzenstadler, Birgit, Jayden Khakurel, Carl Plojo, Marinela Sanchez, Ruben Marin, and Lam Tran. "Resilient Smart Gardens—Exploration of a Blueprint." Sustainability 10, no. 8 (2018): 2654.