Sustainability Requirement Patterns

Kristin Roher, Debra Richardson
Bren School of Information and Computer Science
University of California, Irvine
Irvine, CA 92697, USA
{kroher, djr}@ics.uci.edu

Abstract—In light of the looming threats of global climate change and environmental degradation, the software research community has begun to recognize the need to transition toward environmental sustainability. However, the process of detecting, specifying and evaluating requirements related to environmental sustainability within industry is often neglected or at best performed after the system is built. Most software engineering teams do not include any environmental sustainability expertise to account for the broad range of sustainability concerns related to software systems. To overcome the barriers of incorporating environmental sustainability into the requirements engineering process, this paper suggests the use of sustainability requirement patterns (SRPs), which will provide software engineers with guidance on how to write specific types of sustainability requirements. SRPs contain information on the situations in which the pattern should be used, a starting point for developing sustainability requirements, and any other information needed to develop requirements of a specific type. This research is part of an overarching research project on Software Engineering for Sustainability.

Keywords—sustainability; software engineering; requirements engineering; software requirement patterns; software engineering for sustainability; requirements elicitation

I. INTRODUCTION

Software-intensive systems have led to profound advances in human civilization, but at the same time have contributed significantly to the exploitation of the Earth’s resources. Moreover, software-intensive systems – which interact with other devices, sensors, other systems and people, such as in cell phones, laptops, automobiles – are heavily integrated into day-to-day activities in the industrialized world. As such, they provide a powerful leverage point for enabling sustainability\(^1\) concerns to be brought to bear across a wide range of domains. Developers of these systems, however, generally lack a comprehensive understanding of how to integrate sustainability into their software development processes [1]. If at all, engineers may consider sustainability of the software itself with respect to energy use and fail to consider the multitude of other ways that the software could positively or negatively affect the environment (for example, directly affecting less usage of other resources or indirectly influencing user behavior).

The software engineering research community has begun to recognize the need to transition toward sustainability. For example, the theme of the International Conference on Software Engineering (ICSE) in 2012 was “Sustainable Software for a Sustainable World” and a related workshop was held on “Green and Sustainable Software.” This recognition followed the more generally applicable UN Millennium Development Goals and the related action plan, Agenda 21, which include objectives to reverse the loss of environmental resources and employ sustainable development towards conservation and management of resources [2, 3]. The software industry as a whole, however, has not yet taken action to incorporate environmental sustainability. Considering the three pillars of sustainability (environmental, social, and economic) defined by the 2005 World Summit on Social Development [4], economic and social sustainability offer direct and often immediate benefits for both businesses and users, whereas the benefits of environmental sustainability are much more gradual and often unseen. Therefore, software development organizations often address the economic and social aspects but continue to overlook a system’s affect on environmental sustainability.

It is time to address the effects that software systems have on our environment and its sustainability. We conjecture that there are two primary reasons why such a change has not yet occurred in prevailing software development processes. First, there is often an intellectual barrier that must be overcome since there is non-trivial additional complexity when considering sustainability in software development. Second, this added complexity might increase the financial cost of software development, especially if sustainability experts are hired. Even though the long-term benefits (such as aiding in the effort to decrease global climate disruption) can be abundant, software companies often prioritize time-to-market over product value.

To encourage software developers to address sustainability, we propose the sustainability requirement pattern (SRP), which provide guidance on how to write specific types of sustainability requirements. SRPs contain information on the situations in which the pattern should be used, a starting point for developing sustainability requirements, and any other information needed to develop requirements of this type. Our approach builds on Withall’s software requirement patterns [5].

Section II provides background research on software patterns, while Section III describes the proposed sustainability requirement patterns and their structure. Section IV discusses the process of writing the proposed patterns and section V discusses limitations and future work. Section VI discusses the context of our work, followed by concluding remarks in section VII.

---

\(^1\) For the duration of this paper, sustainability will infer environmental sustainability unless noted otherwise.
II. BACKGROUND: SOFTWARE PATTERNS

Software developers use software patterns at various stages of the life cycle. The Gang-of-Four described object-oriented design patterns through a general document format [6]. Architectural patterns were developed for various domains, such as user interfaces (e.g., model-view-controller pattern) and embedded control applications (e.g., sense-compute-control pattern) among others [7]. Schmidt et al. realized that software patterns can benefit developers by providing solutions to common problems, avoiding costly mistakes, promoting clear communication, and supporting better reasoning [8].

The benefits realized by the use of design and architecture patterns, along with the recognition that requirements are potentially reusable across systems, led to the development of software requirement patterns. Requirement patterns provide a starting point for communication between stakeholders and developers, thus allowing developers to spend their time on novel ideas rather than starting from scratch to solve known problems. Requirements engineering researchers realized that patterns could be used to alleviate the redundancy of defining a new set of requirements for every system. Among those researchers is Withall, who noted that domain-specific requirement patterns could be used to lessen the workload and skill level required to elicit requirements in unfamiliar domains as well as to improve communication between developers and stakeholders [5].

Software patterns have been widely accepted and applied within the area of security. Although most work has been done on applying security patterns to the design phase, security requirement patterns are gaining popularity. Schumacher et al. developed a security pattern format usable within all phases of the software development life cycle and also specified security requirement patterns for accounting, access control, identification, and authentication [9]. By considering the characteristics and boundaries of a security requirement pattern, Slavin et al. adapted Schumacher’s format to be specifically suitable for requirements engineers, thereby reducing specification and elicitation effort [10].

Patterns have successfully supported software developers in specifying security requirements, which led us to believe it may be appropriate to use patterns to encourage and support software developers in the transition to considering sustainability within their systems.

III. CONCEPT: SUSTAINABILITY REQUIREMENT PATTERNS

Based on the software requirement patterns defined by Withall [5], we define a sustainability requirement pattern (SRP) as an artifact that provides guidance on how to write specific types of sustainability requirements. Sustainability requirements may be used to specify system behavior (e.g. requirements that will reduce a system’s energy consumption) as well as to influence the users’ behavior (e.g. the system incentivizes sustainable actions).

Building on Withall’s structure for software requirement patterns, we propose the following SRP format: Summary, Applicability, Content, Archetype (a generalized requirement from which more detailed instances can be derived and tailored), Examples, Discussion, and Related Patterns. The combination of these sections serves to address the necessary characteristics of a pattern that requirements engineers can use to include sustainability in their systems. Each of the sections proposed for a sustainability requirement pattern is described in the following paragraphs.

The Summary section contains basic details about the pattern including but not limited to domain, anticipated frequency (i.e. the number of requirements you can specify from the pattern), and pattern author (to foster trust in the pattern, whereby users may recognize authors they personally know or who have developed patterns that they found useful in the past).

The Applicability section makes clear the situations and context in which this pattern should be applied or not. This section may contain knowledge of why incorporating this aspect of sustainability into a system is important, thus shedding light on the context and meaning behind the pattern and motivating the developer to apply the pattern in the right situations. It may also contain a problem statement that defines exactly what software sustainability issue(s) this pattern is to solve and support the users in fully addressing the issue.

The Content section describes which aspects should be included within a requirement derived from this pattern, clearly defining the attributes, capabilities, characteristics, or qualities of the system that need to be considered to successfully specify a requirement of this type. This section is a starting point for users to identify information that should be contained in a requirement of this type. It is meant to aid the user in beginning to actually build a requirement and to help explain the content of the archetypes and examples in the following sections.

The Archetype section offers guidance for requirements engineers on ways in which a pattern can be instantiated. Archetypes are used here in a new way to describe a generalized requirement from which more detailed instances can be derived and tailored. Archetypes are a convenient way for users to elicit requirements by adapting it to fit the needs of the project, thus enabling a wider range of projects and project domains to be considered for each sustainability requirement pattern. Furthermore, we choose to use archetypes instead of “fill-in-the-blank” templates (as is done in Withall’s requirement patterns [5]) to encourage requirements engineers to think outside the box about the possible requirements they can elicit from the pattern, as the topic of environmental sustainability is inherently broad.

The Examples section provides specific requirements derived by tailoring an archetype to a specific system, thus providing requirements engineers with further clarification on ways in which a pattern and its archetype might be instantiated.

The Discussion section is left open to the author to include other aspects that may need to be explained. This may include, for instance, any consequences to the environment or the software system caused by specifying this type of sustainability requirement, as well as knowledge of any known variants or uses of the pattern and where to look if more information gathering is needed. It may also include a discussion of any competing requirements in which the trade-offs need to be considered (e.g. software cost, time-to-market, complexity).

The Related Patterns section lists any patterns that the requirements engineer should also consider when specifying
further requirements. The list contains patterns frequently used in the same domain or similar domains of the current pattern. This serves as a way to direct users to additional ways they might consider sustainability within their system.

An example of sustainability requirement patterns based upon this work can be found at www.ics.uci.edu/~kroher. This example is not included here due to space limitations.

IV. WRITING SUSTAINABILITY REQUIREMENT PATTERNS

Following the guidelines for pattern writing presented by Riaz et al. [11], three sustainability requirement patterns have been identified and documented based on a particular domain and a set of 18 requirements documents. The process is outlined below along with summaries of each identified pattern and how the patterns can be applied in future projects.

A. Identify the Knowledge Source

There are many potential knowledge sources that can serve to discover sustainability requirement patterns. These include knowledge gained from field research, individual knowledge of the subject, standards and best practices, requirements documents or any other software artifacts. For the research presented in this paper, the authors chose requirements specification documents as the knowledge source. This choice was made based on two factors. First, one of the authors had just completed teaching a requirements engineering class at UCIrvine with 94 students, from which we had easy access to 18 team-developed requirements documents in which sustainability was explicitly considered. Second, using these requirements documents provided information about the requirements of a potential system without being solely influenced by our individual experiences.

B. Formulate Potential Patterns

To identify potential sustainability requirement patterns in a systematic manner, the authors evaluated the 18 requirements documents referenced above. The documents were created for a project that reflects the requirements of an international hotel chain that wishes to encourage their guests to make environmentally sustainable decisions during their stay. For the purposes of this project, the hotel chain has decided that they want to track the guests’ resource consumption on a centralized system so guests can monitor their water consumption, electricity use, food waste, stationary use, and so on. Any additional functionality of the centralized system that could improve the hotels environmental sustainability was encouraged to be included.

For each of the requirements documents, the specified requirements were categorized into types. While any requirement types directly related to environmental sustainability and common among two or more documents were considered to have a high potential for pattern development, the three patterns in Table 1 were present in more than half of the teams’ requirements documents.

C. Document the Patterns

Through the process described above, we identified three sustainability requirement patterns: Communicating Resource Consumption; Incentivizing Minimal Resource Consumption; and Tracking Resource Consumption. Table 1 gives an overview of these three sustainability requirement patterns. Due to space limitations, a brief of each pattern is given by stating the pattern name, the associated archetype, and some representative examples of how the pattern was instantiated as a requirement in the requirements specification documents.

D. Potential Areas for Pattern Application

These patterns should be considered for all systems developed in which consumption of any resource is an issue, and hence reduction (or not) of resource consumption could positively or negatively impact environmental sustainability. It should be noted that these patterns may not be applied to all systems built to reduce resource consumption, however, they should be considered within these systems. Other than a hotel chain, these patterns could be used during requirements elicitation for a wide variety of applications that track resource consumption, such as in a home, fitness center, software company, as well as within an e-commerce system to encourage the user to use sustainable packaging options.

V. LIMITATIONS AND FUTURE WORK

This work is preliminary and there are several limitations worth addressing. Since all of the software requirements documents evaluated were for the same application, the patterns are specific to that application and may only partially translate to other projects within similar application domains. Nevertheless, these patterns can be used as a foundation for developing a generic set of patterns and as a requirements elicitation technique within case studies. Currently, we are working with industrial partners on several case studies on integrating sustainability into existing software systems or systems under development (as well as seeking out future collaborations possibly with consulting agencies that greenify industries). The sustainability requirements written for these case studies will be evaluated to find potential sustainability requirement patterns.

Although our current approach to identifying patterns has not considered all options, we believe that we have demonstrated the utility of sustainability requirement patterns and the potential for further pattern identification through additional knowledge sources. We intend to look into identifying patterns from requirements documents for several additional software systems, as well as any common challenges/recurrences regarding sustainability found while performing case studies.

In the next phase of this research, we will incorporate additional sections within the patterns. For instance, we may create a section entitled “Metrics for Assessment” to provide a sounder basis for evaluation of sustainability. Another supplementary section on testing may be added to provide guidance for software testers, e.g., identifying relevant testing techniques or areas where there are no known testing solutions.

To further help developers identify when to apply a pattern, we will integrate the sustainability requirement patterns into the recommender system described in [12]. During requirements engineering, this system will then recommend sustainability requirement patterns, based on application domain and development locale, that should be considered in a given software system. By recommending patterns, this project aims to increase stakeholder commitment to sustainability and reduce the costs and effort associated with that commitment.
VI. SOFTWARE ENGINEERING FOR SUSTAINABILITY

The work described in this paper is part of a larger project at UCIrvine on Software Engineering for Sustainability (SE4S), where we are working towards a methodology for developing software-intensive systems that meet the functional needs of users while reducing environmental impacts brought about by those systems [13]. Our research is focused on how software engineering can support environmental sustainability in the wide variety of domains in which software is deployed. This goes beyond the aim of many sustainable software efforts that explore how software engineering can contribute to improving the sustainability of the software itself but not its broader, potentially indirect impacts.

SE4S aims to overcome challenges similar to those realized for safety and security in software engineering [14, 15]. At one time, software development practices were not equipped to deal with these emerging critical concerns and were updated accordingly. SE4S aims to make development practices for sustainability just as prevalent as those for safety and security are today, in particular by considering sustainability as a first class quality attribute.

One of the primary topics of the SE4S project is examining how to support sustainability in requirements engineering. This involves developing an adequate set of models, methods, and tools to elicit, document and analyze sustainability requirements for software systems.

VII. CONCLUSION

Sustainability requirement patterns are proposed as an artifact that describes the applicability, content, motivation, archetypes, discussion and any additional related patterns. By supporting requirements engineering with sustainability requirement patterns, we hope to encourage integrating sustainability with software engineering as well as to advance requirements engineers’ ability to produce software systems that have reduced negative impact on the environment. In addition, we hope to see similar benefits to those of other, previously developed software patterns (e.g., increased communication between developers and other stakeholders as well as increased software developer knowledge of sustainability).

The overarching SE4S project provides an avenue for creating awareness of the multitude of potential sustainability impacts present in software systems. This is one way that software engineers will be able to play a key role in reducing the environmental concerns that threaten today’s society.

VIII. ACKNOWLEDGEMENTS

We would like to thank Birgit Penzenstadler and Juliet Norton for their review and contributions to this paper.

REFERENCES


11