Chapter 12
The Impact of Human Factors on Software Sustainability

Asif Imran and Tevfik Kosar

Abstract Software engineering is a constantly evolving subject area that faces new challenges every day as it tries to automate newer business processes. One of the key challenges to the success of a software solution is attaining sustainability. The inability of numerous software to sustain for the desired time length is caused by limited consideration given to sustainability during the stages of software development. This chapter presents a detailed and inclusive study covering human factor-related challenges of and approaches to software sustainability. Sustainability can be achieved by conducting specific activities at the human, environmental, and economic level. Human factors include critical social activities such as leadership and communication. This chapter groups the existing research efforts based on the above aspects. Next, how those aspects affect software sustainability is studied via a survey of software practitioners. Based on the findings, it was observed that human sustainability aspects are important, and that taking one into consideration and ignoring the other factors will threaten the sustainability of software products. Despite the noteworthy advantages of making a software sustainable, the research community has presented only a limited number of approaches that contribute to improving the human factors to achieve sustainability. To the best of our knowledge, these representations require further research. In this regard, an organized, structured, and detailed study is required on existing human factor-related sustainability approaches which will serve as a one-stop-service for researchers and software engineers who are willing to learn about these.

12.1 Introduction

Software sustainability is an important area of software engineering research today. The goal of software sustainability engineering is to ensure that software continues to achieve its goals despite updates, modifications, and evolution [1]. We consider

A. Imran (✉) · T. Kosar
University at Buffalo, Buffalo, NY, USA
e-mail: asifimra@buffalo.edu; tkosar@buffalo.edu

© Springer Nature Switzerland AG 2021

C. Calero et al. (eds.), *Software Sustainability*,
https://doi.org/10.1007/978-3-030-69970-3_12

the definition of sustainability provided by the Software Sustainability Institute, which states, "software you use today will be available—and continue to be improved and supported in the future" [2]. Other definitions of software sustainability consider the age of software and social aspects. Software sustainability can help us achieve a number of useful goals. Some notable goals of software sustainability are mentioned below:

- ***Operational efficiency***: Sustainability of software used both in industries and by individuals should be a natural part of the overall performance management practice [3]. If the software possess the capability to sustain for a long time, there is no need to train researchers on new type of software [4]. Researchers will become more efficient if they use the same software for a long time, thereby increasing their operational efficiency [4]. Also, an individual using a software for a significant amount of time is likely to stick to that software rather than move to a new one.
- ***Desirable reputation of software product***: To remain competitive, companies need to make innovation their top priority [5]. For example, if the software developed by a company is sustainable from human, environment, and economic perspectives, they can state that their software are long lasting and ensure high-quality output [6]. Hence, consumers will find the software reliable and have more confidence in using it. This, in turn, will provide the company with the capacity to build a desirable reputation.
- ***Reduced cost***: If a software used by an industry or an individual for day-to-day activities is technologically sustainable, then that industry or individual does not need to invest in a new software in the near future, and so their capital expenditure is reduced, unless a new software is procured which offers increased benefits and better fits the business needs [7]. On the other hand, if the software is not sustainable and it needs to be replaced within a short time, the users (both industry and individual) need to spend on procuring a new software, installing it on the computers, arranging for training on the use of the new software, etc. Hence, both the capital and current expenditures will rise due to the lack of sustainability of software [8]. From a business perspective, investing in a software which is sustainable will guarantee cost reduction and profit increase in the long run [8, 9].
- ***Accelerated progress of scientific software***: The influence of digital technology in modern research is manifold, where data and publications are being produced, shared, analyzed, and stored using various types of scientific software [10]. Although research software plays an important role in the field of science, engineering, and other areas, in most cases they are not developed in a sustainable way [11]. The researchers who develop them may be well versed in their own discipline; however, they may not have the required knowledge on the best practices of software maintainability and sustainability which are needed for reproducibility of simulation results [11]. As stated by the US Research Software Sustainability Institute (URSSI), there is a need for a strategic plan that will conduct the necessary activities of training, prototyping, and implementation,

with a goal to create improved and more sustainable software [10]. This software in turn will accelerate the progress of science.

There are different kinds of activities involved in software sustainability engineering. Depending on their complexity and applications, conceptually, sustainability can be divided into three broad levels: human, environmental, and economic. Human sustainability encompasses the development of skills and human capacity to support the functions and sustainable software development of an organization and to promote sustainable software development and usage practices. In this chapter we focus on the areas of concern regarding human sustainability, identify what is being currently done, determine the pillars of human sustainability in software, and conduct a human-factor-based study on the impact on sustainability.

A previous survey paper by Penzenstadler et al. [12] covered several low-level components for understanding software sustainability, focusing mostly on the environmental and economic attributes. However, there is a need to identify the human factors as well to provide a holistic viewpoint for software engineers and researchers. Their extended review on sustainability [13] focused on the sustainable design of software. They stated that secured software design and testing are important to develop sustainable software. They limited their work to specific programming components such as commenting code and following coding standards. However, the authors did not analyze the effect of important aspects like requirement prioritization, code smell detection, change management, etc., which equally play a role in ensuring sustainability. Calero et al. [14] provided a review on software sustainability which is primarily based on environmental friendliness of software. However, human and economic aspects like documentation skills, sustainability manifestos, funding, and leadership skills of the project manager were not considered.

The objective of this chapter is to provide a systematic and comprehensive overview of how stakeholders view human factors to be impacting sustainability. We discuss various types of human activities which aim to make software sustainable. We compare and contrast how these approaches apply from the perspective of software engineers.

Our findings show that software practitioners view certain human factors as critical to sustainability. However, most of the sustainability techniques are rarely applied due to lack of knowledge of the software community [10]. Currently, organizations like the US Research Software Sustainability Institute (URSSI) [10] and The Software Sustainability Institute [2] are investigating to address those issues. Hence extensive research is required to solve the impact of human factors on sustainability.

Based on our findings, we argue that under present circumstances there is still room for improvement in the field of sustainable software development regarding human factors. The open issues emerging from this study will provide input to researchers who are willing to develop improved techniques for addressing human factors to achieve software sustainability. We conclude that to achieve sustenance, human, environmental, and economic factors need to be considered simultaneously.

Considering one and ignoring the others will not provide long-term sustenance of software.

Based on the above information, the major contribution of this chapter can be stated as follows:

- Integrate software practitioner's feedback to identify the negative impact of the smells on architectural debt.

The rest of the chapter proceeds as follows. Section 12.2 illustrates the research questions and identifies the survey questionnaire to collect expert opinions. Section 12.3 describes the implementation of tools for architectural smell detection and recording. It also describes how Spectral clustering is used to group smells. Section 12.4 provides the obtained results and analyzes them. Section 12.5 discusses related work, and Sect. 12.6 concludes the chapter and discusses future research directions.

12.2 Empirical Study Setup

The workflow of the chapter proceeds as follows. Figure 12.1 shows the alignment of the various human factors of this chapter to the core elements of sustainability [15]. As seen in the figure, we identified six human factors, which are related to software sustainability. We took expert opinion via a survey based on the assumption that the impact of the human factors on sustainability for one software can be applied to other, similar software [16]. We studied the feedback of software practitioners to analyze impact since this added intelligence cannot be obtained from software (source code, design documents, blogs, and QA reports).

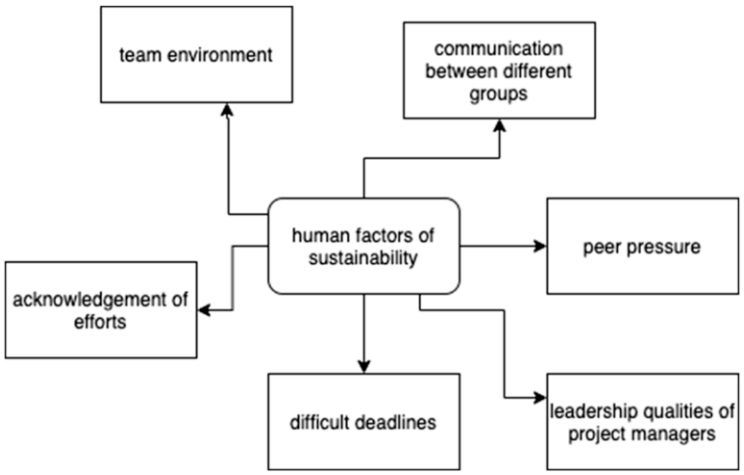

Fig. 12.1 Representation of human factors affecting sustainability

This study focuses on detecting the important human aspects to sustainability. Next, their negative impact is analyzed via feedback from software practitioners. Specifically, the following human factors are identified:

1. Team environment: The environment of the team should be such that all members should own the project and believe in its scope, schedule, and chances of success. If the teammates are not motivated to work towards success of the project, that severely affects the sustainability as well [17].
2. Communication: Lack of effective communication can break the sustainability of a software project. Even if all other aspects of the team are ideal, when communication is lacking, you will have sub-par sustainability. Effective communication can also allow teams to overcome many less-than-ideal circumstances. Here, it must be noted that for sustainability the power of the team members to listen carefully to each other is critically important.
3. Leadership qualities: Leadership includes the skills through which a project manager handles change management, timeline management, cost management, employee turnover, etc. [14].
4. Difficult deadlines: These are stressful for the humans involved in the software project, and if there are more than one, then it can be overwhelming. This leads to software engineers trying to complete the software development rather than focusing on the sustainability manifestos.
5. Peer pressure: Similar to difficult deadlines, peer pressure can create unnecessary stress on software developers which can lead to the developers trying to finish the project early to get into superiors' good books, bypassing the sustainability benchmarks of the software.
6. Acknowledgment of efforts: The appreciation of one's hard work to follow sustainability benchmarks while designing and developing software systems can go a long way to ensure that the critical attributes of sustainability are preserved.

12.2.1 Research Question

As software evolves from initial to matured phases, its sustainability may incur significant interest. Hence, we should be able to identify which human factors have a higher negative impact on sustainability based on expert opinion. This will help to prioritize smells for refactoring. More specifically, the major research question is as follows:

RQ: Which human factors have a greater impact on sustainability according to software practitioners?

For impact determination, we gave the developers an option to determine whether the impact on sustainability is high, low, or no impact. We provided a questionnaire to the developers seeking answers for their selected impact for a human factor. The questionnaire asked about the factors which influenced the developers to assign a

specific impact to a smell. These factors included the *impact of mental pressure to meet deadlines [18], the causal effect, and the context dependency of a factor, which resulted in more factors [19], and aspects focusing on team environment, and communication issues [17]*. Hence, the human factor and the presence of specific community contexts were considered for impact on sustainability. These factors are described in greater detail in the following section. The answers to these questions can help team leaders address community issues which threaten sustainability.

12.2.2 Survey Structure

We designed an online survey questionnaire with a minimal number of questions. The goal was to reduce the cognitive complexity and at the same time obtain the required information [20]. The questionnaire aimed to define what is meant by software sustainability and provide the human factors impacting sustainability to the developer and, at the same time, ask them to tag the adverse impact of a factor as high, low, or no impact. Also, we tried to obtain justification behind the tagging, trying to identify the context and any human factor if present. To ensure that the respondents are well aware of what is meant by sustainability, we conducted a training session where we remotely presented the definitions, examples, and scenarios where human factors threatening sustainability may occur and tried to determine the impact caused by them. After the training, we provided them the questionnaire. Then we identified the following questions based on motivation from Palomba et al. [17].

Q1 Are you aware of the identified human factors? If yes, are those handled at your company?
Q2 Will you tag the factor as having high, moderate negative impact, or no impact on sustainability?
Q3 Was your answer to Q2 based on a specific software context, time, or effort?
Q4 Did any friction in your development community affect impact determination? Explain.
Q5 How does the negative affect on the mindset of software engineers due to COVID-19 impact sustainability of the software?

After detection of the issues, we present the factors to the software practitioners together with the questionnaire. The software practitioners who worked the most on the critical parts were asked the questions [17]. This was determined from the Git commits. The respondents included system architect and senior developers in the team who worked with the critical software components and were under stress to meet deadlines. If there were multiple developers who worked equally on a class, we identified the developer who worked solely on that class and no other classes. This is based on the assumption that the developer who worked only on that class is possibly the owner of the class and knows it in depth, and will be the best individual to know whether any kind of human factors threatened the sustainability of the class.

12.3 Survey Exercise

This section describes the implementation of the survey mechanism to determine human factors and their impact on software sustainability. There is a need to group sustainability factors based on human activities. The exercise has been applied to a real-life software on a testing phase and feedback was collected from the developers.

As a result, we need to introduce context awareness. This is done via questionnaire tagging, and we have an additional option where the developer can flag a smell as "having no negative impact" based on the context. We trust the software practitioner's knowledge and expert opinion in this regard. For clustering, we obtained the refactored dataset and conducted the analysis.

For our impact analysis of human factors on software sustainability, we chose OneDataShare [21] since we have full access to the developers of this software to get the required response in the survey. OneDataShare is an open-source software which started in 2016. This software have been studied earlier to analyze impacts in terms of performance [22]. However the effect of human components on the sustainability of the software has not been studied earlier. In this chapter, we aim to fill that gap.

Altogether, we collected data from 10 software practitioners who worked in the OneDataShare project. In total, there were 14 developers, yielding a response rate of 71.43%. OneDataShare is a research software for fast data transfer, which is a flagship project, hence easy access could be obtained to the respondents. Next, the 10 software practitioners who were surveyed worked the most with the classes suffering from smells, and they were the most concerned, hence readily responded.

12.4 Results

In order to reliably state that the smells detected by the tool negatively impact Architectural Technical Debt (ATD), we first need to establish reliance on the tool, followed by taking community feedback to analyze the impact of smells. First, it is highly important to justify that the tool is capable of accurately detecting smells, then to compare the smell detection output of the tool by applying it to the software with a predefined set of smells that are frequently used for experimental purposes in the existing literature. Also, it is equally important to show that the tool can be generalized and used to detect smells in software applications other than OneDataShare. We answer the identified research questions which will generalize our findings.

12.4.1 Answer to RQs

This section describes the results obtained for each of the survey questions to analyze the impact on ATD.

1. *Are you aware of the identified human factors? If yes, are those handled at your company? [17]*

The following awareness issues could be determined by analyzing the responses.

(a) **Lack of awareness:** The surveyed software practitioners were not aware of the impact of certain factors described, such as peer pressure and acknowledgment of efforts, on sustainability. They developed the software using asynchronous event-driven network application framework which enabled quick and easy development and did not consider other factors related to human behavior to be significant catalysts for maintaining sustainability. Besides the lack of knowledge of existing human resources, shorter time to market causes excess pressure from management to deliver the software on-time which is another reason for not paying attention to the human factors.
(b) **Awareness of human factors:** For certain software companies, the respondents were well aware of the remaining factors. They rated that leadership of project managers is the most important aspect of ensuring that the projects sustain. They stated that many of those factors had been addressed from earlier stages of the development life cycle. The respondents stated that regular team meetings, team coffee sessions, sharing of ideas, and sharing a common goal of providing sustainable software services are critical to eliminating the negative impacts of the mentioned human factors in sustainability. During COVID-19, the technical leads were responsible for conducting virtual sessions with team members, learn their thoughts, and eliminate such factors. Even top-level directors tried to communicate regularly with junior employees and interns to motivate them towards building sustainable software.

2. *Will you tag the factor as having high, moderate negative impact, or no impact on sustainability?*

The impact of the identified human factors on sustainability was determined based on the opinions from experts. The response from various respondents regarding the score given to each factor is shown in Fig. 12.2. We summarize the findings as follows:

(a) **Factors with a high negative impact on sustainability**: We see that the software practitioners have flagged the factor called "Leadership" as having a high negative impact on sustainability. "Communication" has been identified as an important factor as well. Many teams who responded were not aware of human factors "peer pressure" and "acknowledgment of efforts." Hence this survey was an eye opener for them to address these issues.
(b) **Factors with a moderate negative impact on ATD**: The survey reported that two types of smells called "Team environment" has a low impact on sustainability according to numerous respondents. The reason was that this factor did not affect a large number of modules of the sustainability software and required relatively lesser effort to solve, given there is good leadership. We received comments from software practitioners, such as, "In some cases like COVID-19, it is the way the engineers are forced to work from home and during such times

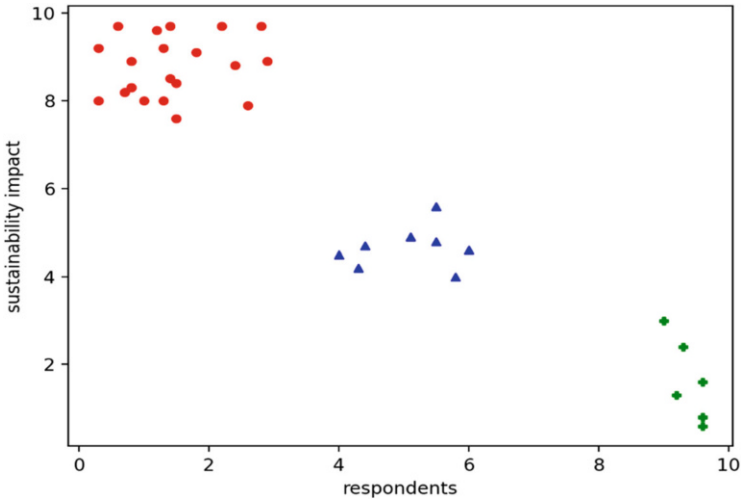

Fig. 12.2 Representation of human factors affecting sustainability

good leadership and continuous communication is more important. Team environment is not important given good leadership since that happens automatically if good leadership is present, and during COVID-19 teams are unable to meet physically for the environment factor is absent. In other cases, we find that effective communication is more important to result in good team environment." The respondents felt that given the short time to market and the COVID-19 situation when this survey was conducted, these factors should be given lower priority compared to the others with a higher negative impact on sustainability.

(c) **Factors with no negative impact:** None of the identified factors were found to have negligible impact on sustainability. In fact, the respondents said that other human factors like gender-based discrimination and racial discrimination should be included in future studies as other important human factors which affect sustainability.

3. *Was your answer to Q2 based on a specific software context, time, or effort?*

During the survey, the following answers were obtained:

(a) **Time and effort**: In line with the survey the responses were motivated by the time and effort required to solve them. Also, lack of awareness was another aspect in this regard. For "difficult deadlines," the deciding factor was time as well.

(b) **Software context**: For the "Leadership qualities" factor, the experts primarily based their answer on software context, stating that for large software, such factors can have tremendous negative impact on sustainability. The reason being that mostly this factor occurs when many employees work on a big project and all of them are not clearly aware of its sustainability goals.

4. *Did any friction in your development community affect impact determination? Explain.*

   For this question, all responses were the same, as discussed below:

(a) For all the human factors which were presented to the respondents, they responded saying they did not have any communication issues or team friction. Capable leadership and good communication may be a reason for such a response. At the same time, the teams which were interviewed worked under the same roof, and they were not affected by any communication gap which happens during a collaboration between remote teams. Also, the hierarchy was flat, which ensured swift decisions. However, studying the impacts of community smells [17] on sustainability with respect to this type of software can be an interesting topic.

12.4.2 Implications of Results

The analyses provided in this chapter can be applied to software researchers and practitioners.

- **Usefulness to researchers:** Researchers can use this information to identify which human factors occur specifically and cause a greater adverse impact on sustainability. This will provide them a useful lead on which type of human factors to focus their research on based on the type of software. Addressing human factors of sustainability opens a new doorway of research in this area as it can reduce the challenges to make a software sustainable.
- **Usefulness to software practitioners:** Firstly, managers in software companies can apply the results on their projects, collect the data, and analyze results to improve their sustainability practices. Also, by keeping the results of the chapter in mind, attention can be paid to specific factors so as to incur minimum impact on sustainability.

By surveying the impact of human factors on sustainability, this chapter allows engineers and researchers to refactor only a subset of factors at a time.

12.5 Related Work

A model depicting the collective efforts required to detect architectural smells is proposed based on the study of related literature [23]. The process of prioritizing architectural smells based on the impact of those on ATD has been studied by Martini et al. [24]. The authors analyzed how users perceive the smells to affect architectural debt. Although the model considers important aspects of architectural

smells, it does not study how those smells affect sustainability. Also, the human factors which influence those smells were not considered.

Supervised machine learning approaches have been explored for code and architectural smells detection [25, 26]. However, the authors debated that exiting research using such techniques deal with biased datasets for the training of smells. They stated the need for further research with more realistic datasets to obtain the actual performance of the tool. This leads to the need for further research using unsupervised techniques for smell detection.

The effect of the developer's seniority, frequency of commits, and interval of commits on reducing architectural debts in software were evaluated by Alfayez et al. [27]. The authors determined that seniority and frequency of commits are negatively correlated with reducing architectural debt, whereas the interval of commits is positively correlated. The authors used statistical analysis tests to validate the effects of developer behavior on architectural smells. However, use of unsupervised machine learning for grouping smells was not explored.

Existing tools for detecting architectural smells include *Decor* [28], *Arcan* [29], and *Designite* [30]. Most of these focus on software written in languages other than Java. *AnaConDebt* [31] is used to assess technical debt. Further research is required to include community feedback for the smells detected by those software to prioritize for ATD.

Polomba et al. [32] was able to detect code smell which could not be differentiated via their structures. As a result, they made software suffering from those types of code smells more sustainable by improving detection of smelly code which ultimately set up its removal. The impact of community smells on software was studied based on surveys from software engineers [17]. This motivates us to study how community factors affect sustainability of the software as well.

Using open-source components in design and implementation of research software guarantees its sustainability [33]. The researchers have cited that Mozilla Firefox is a successful and sustainable application, mainly because of its open-source structure. They stated that Mozilla still continues to support, train, and research open-source software [34]. The authors highlighted that based on their years of experience in mentoring open source software projects, there are primarily three areas to consider in order to make an open-source project sustainable and ensure its growth. They are training, peer support, and availability of financial and computational resources. How effective training can be ensured for research-based software is not identified by the authors. Also, they highlighted community support for a largely used software like Mozilla, but how community support can be achieved for a research software which serves a small group of researchers have not been stated.

The importance of studying the experiences of the stewards who developed and used a wide range of open-source software tools to identify and focus on open-source software sustainability has been addressed in [35]. The Hierarchy Data Format (HDF) group has been working with the research community for 30 years, building open-source tools to provide them platforms for data storage, easy access, and analysis. It has analyzed Github and found that nearly 1000 repositories are

based on its open-source codes. At the same time several broadly successful open-source systems are centered around HDF. The authors shared their experience of *PyTables* that whenever an existing code is reused or refactored, there is a high probability that it will present unforeseen issues which require correction and addressing, thus reducing the time to market. However, effective refactoring techniques for a fast data sharing software like OneDataShare have not been addressed.

Best practices in software usability and user experience can have a significant effect on software sustainability [36]. By following software usability best practices, failures in the software can be fixed at a lower cost, and performance can be enhanced [37]. The importance of user experience has been emphasized in the academia by designing course works in Human-Computer Interaction (HCI). However, most of the HCI courses are non-major and considered to be esoteric by researchers coming from scientific backgrounds. However, the dynamic nature of scientific software has made user experience an important criterion for its sustainability [36]. As a result, the authors combined heuristic studies, participant-driven interviews and surveys, usability observations, and evaluations to improve user experience of scientific applications. These experiences have been used to develop User Interfaces data exploration and analysis, create workflow models, and design and build data management tools. However, using such tools for a research software that transfers a high volume of data within a desirable time frame has not been addressed.

12.6 Conclusion and Future Work

This chapter identifies and analyzes the impact of human factors in sustainability via a survey. We detected six types of factors and evaluated the impact of those by interviewing experts in the field of software engineering. We proceeded to analyze how software practitioners saw the negative impact of those factors. We provided a questionnaire to the software practitioners to gather their viewpoints related to the adverse effects of the human factors on sustainability. Results show that the "Leadership" and "Communication" factors were rated by the practitioners to have a high impact on sustainability. On the other hand, the factor called *team environment* had less negative impact.

In the future, it may also be helpful to perform a longitudinal study that detects the precision of the survey results by increasing the sample size. Another interesting area would be to extend the list of human factors to include gender inequality and racial discrimination, which also negatively impact sustainability.

References

1. I. of Electrical and E. Engineers. Defining software sustainability. [Online]. https://ieeexplore.ieee.org/Xplore/home.jsp. Accessed 14 Sept 2019

2. Software Sustainability Institute (2016) https://www.software.ac.uk/case-studies. Accessed 10 Sept 2019
3. Albertao F, Xiao J, Tian C, Lu Y, Zhang KQ, Liu C (2010) Measuring the sustainability performance of software projects. In: 2010 IEEE 7th International Conference on E-Business Engineering. IEEE, pp 369–373.
4. Penzenstadler B, Fleischmann A (2011) Teach sustainability in software engineering? In: 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T). IEEE, pp 454–458
5. Dagli CH, Kilicay-Ergin N (2008) System of systems architecting. In: System of Systems Engineering: Innovations for the 21st Century. pp 77–100
6. Durdik Z, Klatt B, Koziolek H, Krogmann K, Stammel J, Weiss R (2012) Sustainability guidelines for long-living software systems. In: Software Maintenance (ICSM), 2012 28th IEEE International Conference on. IEEE, pp 517–526
7. Stewart CA, Barnett WK, Wernert EA, Wernert JA, Welch V, Knepper R (2015) Sustained software for cyberinfrastructure: analyses of successful efforts with a focus on nsf-funded software. In: Proceedings of the 1st Workshop on The Science of Cyberinfrastructure: Research, Experience, Applications and Models, ser. SCREAM '15. ACM, New York, NY, pp 63–72. [Online]. http://doi.acm.org/10.1145/2753524.2753533
8. Seacord RC, Elm J, Goethert W, Lewis GA, Plakosh D, Robert J, Wrage L, Lindvall M (2003) Measuring software sustainability. In: International Conference on Software Maintenance (ICSM). IEEE, p 450
9. Chitchyan R, Becker C, Betz S, Duboc L, Penzenstadler B, Seyff N, Venters CC (2016) Sustainability design in requirements engineering: state of practice. In: Proceedings of the 38th International Conference on Software Engineering Companion, ser. ICSE '16. ACM, New York, NY, pp 533–542. [Online]. http://doi.acm.org/10.1145/2889160.2889217
10. URSSI. Developing a pathway to research software sustainability. http://urssi.us/. Accessed 14 Sept 2019
11. Carver JC, Gesing S, Katz DS, Ram K, Weber N (2018) Conceptualization of a us research software sustainability institute (urssi). Comput Sci Eng 20(3):4–9
12. Penzenstadler B, Bauer V, Calero C, Franch X (2012) Sustainability in software engineering: a systematic literature review
13. Penzenstadler B, Raturi A, Richardson D, Tomlinson B (2014) Safety, security, now sustainability: the non-functional requirement for the 21st century. IEEE Softw 1:1
14. Calero C, Bertoa MF, Moraga MÁ (2013) A systematic literature review for software sustainability measures. In: Proceedings of the 2nd International Workshop on Green and Sustainable Software. IEEE Press, pp 46–53
15. Imran A, Kosar T (2019) Software sustainability: a systematic literature review and comprehensive analysis. arXiv preprint arXiv:1910.06109
16. Tockey S (2014) Aspects of software valuation. In: Economics-Driven Software Architecture. Elsevier, pp 37–58
17. Palomba F, Tamburri DAA, Fontana FA, Oliveto R, Zaidman A, Serebrenik A (2018) Beyond technical aspects: How do community smells influence the intensity of code smells?. IEEE Trans Softw Eng
18. de Andrade HS, Almeida E, Crnkovic I (2014) Architectural bad smells in software product lines: an exploratory study. In: Proceedings of the WICSA 2014 Companion Volume. ACM, p 12
19. Zazworka N, Shaw MA, Shull F, Seaman C (2011) Investigating the impact of design debt on software quality. In: Proceedings of the 2nd Workshop on Managing Technical Debt, ser. MTD '11. ACM, New York, NY, pp 17–23. [Online]. http://doi.acm.org/10.1145/1985362.1985366
20. Dillman DA (2011) Mail and Internet surveys: the tailored design method–2007 Update with new Internet, visual, and mixed-mode guide. Wiley

21. Imran A, Nine MS, Guner K, Kosar T (2018) Onedatashare-a vision for cloud-hosted data transfer scheduling and optimization as a service. In: Proceedings of the 8th International Conference on Cloud Computing and Services Science, vol 1
22. Imran A, Kosar T (2020) The impact of auto-refactoring code smells on the resource utilization of cloud software. In: García-Castro R (ed) The 32nd International Conference on Software Engineering and Knowledge Engineering, SEKE 2020, KSIR Virtual Conference Center, USA, 9–19 July 2020. KSI Research Inc., pp 299–304. [Online]. https://doi.org/10.18293/SEKE2020-138
23. Besker T, Martini A, Bosch J (2016) A systematic literature review and a unified model of ATD. In: 2016 42nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2016, pp 189–197
24. Martini A, Fontana FA, Biaggi A, Roveda R (2018) Identifying and prioritizing architectural debt through architectural smells: a case study in a large software company. In: European Conference on Software Architecture. Springer, pp 320–335
25. Caram FL, Rodrigues BRDO, Campanelli AS, Parreiras FS (2019) Machine learning techniques for code smells detection: a systematic mapping study. Int J Softw Eng Knowl Eng 29 (02):285–316
26. Fontana FA, Mäntylä MV, Zanoni M, Marino A (2016) Comparing and experimenting machine learning techniques for code smell detection. Empirical Softw Eng 21(3):1143–1191
27. Alfayez R, Behnamghader P, Srisopha K, Boehm B (2018) An exploratory study on the influence of developers in technical debt. In: Proceedings of the 2018 International Conference on Technical Debt, ser. TechDebt '18. ACM, New York, NY, pp 1–10. [Online]. http://doi.acm.org/10.1145/3194164.3194165
28. Moha N, Guéhéneuc Y-G, Le Meur A-F, Duchien L, Tiberghien A (2010) From a domain analysis to the specification and detection of code and design smells. Formal Aspects Comput 22(3–4):345–361
29. Fontana FA, Pigazzini I, Roveda R, Tamburri D, Zanoni M, Di Nitto E (2017) Arcan: a tool for architectural smells detection. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). IEEE, pp 282–285
30. Suryanarayana G, Samarthyam G, Sharma T (2014) Refactoring for software design smells: managing technical debt. Morgan Kaufmann
31. Martini A (2018) Anacondebt: a tool to assess and track technical debt. In: 2018 IEEE/ACM International Conference on Technical Debt (TechDebt). IEEE, pp 55–56
32. Palomba F (2015) Textual analysis for code smell detection. In: Proceedings of the 37th International Conference on Software Engineering – vol 2, ser. ICSE '15. IEEE Press, Piscataway, NJ, pp 769–771. [Online]. http://dl.acm.org/citation.cfm?id=2819009.2819162
33. Cabunoc A (2018) Supporting research software by growing a culture of openness in academia
34. Brown AW, Booch G (2002) Reusing open-source software and practices: the impact of opensource on commercial vendors. In: International Conference on Software Reuse. Springer, pp 123–136
35. Haebermann T (2018) Sustainable open source tools for sharing and understanding data. In: USRRI 1st Workshop on Software Sustainability. USRRI, pp 1561–1570
36. Kitzes J, Turek D, Deniz F (2017) The practice of reproducible research: case studies and lessons from the data-intensive sciences. University of California Press
37. Gilb T, Finzi S (1988) Principles of software engineering management, vol 11. Addison-Wesley, Reading, MA