Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 1 : Introduction to Software Sustainability.” Essay. In Software Sustainability. Cham: Springer, 2021.
11th February 2023

Introduction

Calero et al.'s "Software Sustainability" consists of 16 chapters discussing the connection of sustainability and ICT beyond the 'common' environmental aspect but covering all green ICT dimensions.

In this review, the author will summarize the first chapter that focuses on the connection between ICT, Sustainability and Business.

Body

ICT may have a multi-sides impact on sustainability, and it may be defined differently according to how its role is played. When ICT research aims to decrease its footprint, it will be called "sustainability by IT". Meanwhile, it will be called "sustainability in IT" when the aim is to decrease other areas' footprints.

There are two fundamental pillars of sustainability: "The capacity of something to last a long time" and "the resources used." In order to fulfil these pillars, all dimensions of ICT should improve themselves. Even though hardware long starts to decrease its power consumption, the software area still needs to catch up. In order to catch this, software engineer starts to improve sustainability in the whole software deployment process, especially on: software systems, software products, web applications, data centres, Etc. Along with this, developers try to find creative ways to use ICT in business processes to deliver sustainability benefits across the enterprise and beyond. As this, books defined sustainable software is the software coded for code being sustainable, agnostic of purpose and/or being able to support sustainability goals.

Green- IT also has a big role when seen from a business perspective. A business that fails to make sustainable development one of its top priorities could receive considerable public criticism and subsequently lose market legitimacy. Therefore, ISO 26000 standard for Corporate Social Responsibility (CSR) was made to protect the environment, promote greater responsibility, and encourage the adoption of more environmentally-friendly ICT.

As ICT is always intertwined with the real-life process instead of intruding, sustainable ICT should consider its impact on humans, the economy and the environment, which are the UN's three sustainability dimensions. First, the software deployment should consider the sociological and psychological aspects of the software development industry. Second, software lifecycle processes protect stakeholders' investments, ensure benefits, reduce risks, and maintain assets. And finally, how software development should consider its impact on resources (such as energy consumption).

Conclusion

Sustainability is a major deciding factor in the modern economic market. As ICT is a critical aspect of business, it should start aligning with this new concept to keep its standing in the market. However, the implementation of software sustainability is slower than its hardware counterparts. Moreover, the research focuses still mainly on the environmental dimension. Therefore, the author hoped that this book would help developers understand how software deployment protects all sustainability dimensions and conforms to 17 SDGs.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 2 : Criteria for Sustainable Software Products: Analyzing Software, Informing Users, and Politics” Essay. In Software Sustainability. Cham: Springer, 2021.
11th February 2023

Introduction

The second chapter of Calero et al.'s "Software Sustainability" focus on how modern software development goes beyond the actual programming and, therefore, a large number of factors must be taken into account, particularly with regard to energy consumption in software.

Body

Dick et al. defined Sustainable software as software with minimal negative impacts and/or positive impacts on the economy, society, human beings, and environment are minimal on sustainable development. Quantitative measurements are needed to manage these impacts. Therefore, it is reasonable to develop validation criteria to label sustainable software products.

The need for these criteria led to the invention of Blue Angel Certification in Germany. Due to the accountability of energy requirements over the year, this certification is usually awarded for a time period of several years. The aim of the Blue Angel for software products is to reduce the overall energy consumption of ICT and to increase resource efficiency. In order to get this certification, software companies should improve the resource and energy efficiency of their products, potential hardware operating life, and transparent manufacturing process. Thus, the following criteria were selected:

- Required minimum system requirements
- Hardware utilization and electrical power consumption when idle
- Hardware usage and energy requirements when running a standard usage scenario
- Support of the energy management

As Penzenstadler put emphasis on the point of energy efficiency while talking about "sustainable software," the next parts of this chapter mostly discuss how to quantitatively measure how sustainable software is. However, since there is a large variety of software products regarding usage options, software architecture, and of course, purpose, it is complicated to count software resource and energy calculation.

Therefore, Guldner et.al. broke down the components necessary for the assessment of software sustainability before building, while coding, while testing when deploying the system, and after-sales.

- Choosing a suitable programming language
- Wise placement of critical section to prevent increasing power consumption
- Energy-efficient software testing
- Wise usage of container and/or cloud,

and these are the after-sales components:

- Well-documented user manual
- Offline capability
- Encourage users to deactivate standby modes in their system
- No external ads
- Downward compatibility, to make sure the system can be supported by old hardware devices
- User autonomy
- Modular installation

However, it is needed to be noted that in order to produce repeatable experiments, this workload should be automated. So, the energy consumption and hardware usage of a software product can be recorded. This, in turn, can be used for comparisons between software products that perform similar tasks across releases of one software product or for different features of one software product.

Conclusion

Software development is a major issue of sustainable ICT. Thus, it is important for all connected actors to have clear criteria for sustainable ICT in all parts of the software life cycle.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 3 : GSMP: Green Software Measurement Process. Cham: Springer, 2021.
11th February 2023

Introduction

The third chapter of Calero et al.’s “Software Sustainability” aims to define the framework, standard, and activities for analyzing software energy efficiency. A clear methodology is important to obtain greater control over the measurements performed, ensuring the reliability and consistency of the information obtained regarding energy efficiency. The methodology that becomes the focus of this chapter is called Green Software Measurement Process (GSMP).

Body

It is important to be aware of how efficient software is from an energy point of view when running to improve software sustainability. A well-defined and established process allows greater control over the measurements performed, ensuring their reliability and consistency.

Several software measurement standards provide guidelines for efficiently carrying out the software measurement process, such as the Goal/Question/Metric (GQM) method, Practical Software Measurement (PSM) method, and the ISO/IEC/IEEE 15939 standard. However, even though these standards provide guidelines for efficiently carrying out software measurement processes, they are not directly connected to energy measurement.

Therefore, to measure software energy consumption, Hindle proposed a methodology called Green Mining Methodology. The Green Mining methodology describes measuring, extracting, and analyzing energy consumption information from running software. The main defect in this methodology is that it does not provide any valid and reliable protocol to carry out the measurement. For this reason, Jagroep et al. present a measurement protocol in which an extension of “Green Mining” is performed, detailing the specific measurement tasks to be carried out.

However, researchers still lack a specific replicate able and comparable method to analyze the energy efficiency of the software. So, Mancebo et al. defined a process known as the Green Software Measurement Process (GSMP). GSMP integrates all of the activities that must be carried out to measure and analyze the energy consumption of the software being evaluated.

The GSMP comprises seven phases covering the main steps to be performed, from the measurement of energy consumption to the analysis of the results. The GSMP is intended to be performed iteratively, so the phases are closely related. The initial phase focuses primarily on the definition of the requirements and the software system to be evaluated. The next two phases focus on the configuration and preparation of the measurement environment. In the fourth phase, energy consumption measurement activities are carried out. Finally, the last phases are analyzing the data obtained and reporting.

These are the phases of GSMP:

1. Scope Definition
   The main goal of this phase is to obtain a complete specification of the requirements for the evaluation of energy efficiency
2. Measurement Environment Setting
   The second phase’s purpose is to define the measurement environment that will be used to satisfy the goal established in the first phase.
3. Measurement Environment Preparation
   This phase focuses on the preparation of the energy consumption measurements to be performed and on the configuration of the measurement environment that was defined in the second phase
4. Measurement Performance
   During this phase, energy consumption measurements will be carried out.
5. Test Case Data Analysis
   From this phase onwards, the analysis of the energy consumption data obtained by the measuring instrument begins.
6. Software Entity Data Analysis
   Once we have analyzed the energy consumption data of the test cases, we will be able to determine how much energy was consumed when the software entity was executed in the DUT.
7. Reporting the Results
   Finally, the last phase involves documenting the study performed, describing the entire process followed, and the results on the energy consumption of the software extracted. The product of this phase is called the laboratory package, a replicate able and versatile sharing knowledge instrument (documentation) in the form of information and materials.

The later parts of the chapter provide examples of the usage of GSMP. These examples prove that GSMP serves as a guide in the activities that must be followed to measure and analyze the software’s energy consumption, regardless of the measuring instrument used.

Conclusion

In conclusion, the Green Software Measurement Process (GSMP) methodology details all the activities and roles which are necessary to carry out the measurement and analysis of the energy consumption of the software executed. The GSMP ensures the measurements’ reliability and consistency and allows the repetition and comparison of the studies carried out.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 4 : FEETINGS: Framework for Energy Efficiency Testing to Improve eNvironmental Goals of the Software. Cham: Springer, 2021.
11th February 2023

Introduction

The objective of the fourth chapter of Calero et al.’s “Software Sustainability” is to increase awareness of energy consumption by means of establishing a framework that provides a solution to the lack of a single and agreed terminology, a process that helps researchers evaluate the energy efficiency of the software, and a technology environment that allows for accurate measurements of energy consumed. The framework that becomes the focus of this chapter is called FEETINGS (Framework for Energy Efficiency Testing to Improve eNvironmental Goals of the Software).

Body

Energy used for global information and communications technology (ICT) could exceed 20% of total energy and emit up to 5.5% of the world’s carbon emissions by 2025. Although the hardware is generally seen as the main culprit for ICT energy usage, the software also significantly impacts the energy consumed. Unfortunately, little attention has been given to this topic by the ICT community. Therefore, increasing the energy awareness of software developers and end users is necessary.

The starting point of raising public awareness for managing their consumption is to define the clear measurement of software energy consumption. However, there is currently a lack of both knowledge and tools to reliably and accurately analyze software energy consumption. This situation mainly comes from inconsistencies due to a lack of generally agreed-on methodology, terminology, and instruments.

Therefore, Mancebo et al. introduced FEETINGS, a framework to promote more reliable capture and analysis of software energy consumption data. This framework comprises three main components, classified according to their nature as conceptual, methodological, and technological components.

First, the conceptual part of FEETINGS seeks to solve the lack of a unique and agreed terminology, as the common problems of terminology were caused by the confusion and inconsistencies of the main concepts used in software energy assessment. The ontology proposed by FEETINGS is known as “Green Software Measurement Ontology” (GSMO), and its purpose is to provide precise definitions of all terms related to software energy measurement.

Second, the methodological component is the process of measuring and analyzing the energy efficiency of the software. This process is known as the “Green Software Measurement Process” (GSMP). GSMP guides researchers and practitioners in carrying out measurements of software energy consumption. The GSMP ensures greater control over the measurements, improving their reliability, consistency, and coherence. It also ensures that the results obtained are comparable with other studies and facilitates the replicability of the analyses performed.

Finally, the aims of the technological component of the FEETINGS framework is to perform more realistic measurements of the energy consumed by the software and to use these results to analyze the consumption of the software, compare the behavior of different version of the software, identify the consumption patterns, and to give out the recommendation of how to improve energy efficiency. There are two artifacts of this component called EET (Energy Efficiency Tester) and ELLIOT. EET is a measuring instrument that accurately captures the computer’s energy consumption (DUT) on which the software is running. Meanwhile, ELLIOT is a software tool for processing the data collected by EET.

Conclusion

In conclusion, in order to get a precise measurement of energy consumption for raising stakeholders’ awareness of software energy consumption, FEETINGS was introduced. FEETINGS serves two purposes: raising awareness among ICT developers to develop energy-efficient software and showing end users how much energy is required to run their daily-usage software.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 5 : Patterns and Energy Consumption: Design, Implementation, Studies, and Stories” Essay. In Software Sustainability. Cham: Springer, 2021.
11th February 2023

Introduction

The fifth chapters of Calero et al.’s “Software Sustainability” discuss patterns at the code and design level and address energy efficiency not only as the main concern of patterns but also as a side effect of patterns that were not originally intended to deal with this problem.

Body

The growing energy demand associated with ICT usage has become a recurrent concern. Therefore researchers compile these problems and solutions into software patterns. The patterns that address energy consumption as the main concern is called energy patterns.

With the growth of cloud computing and mobile device users, energy consumption’s issue is a critical topic, especially for the massive energy footprint of data/computation centers and battery-powered devices. Therefore, the software optimizations that have been discussed at the source code, design, and architecture level for this problem were compiled into energy patterns to solve similar future issues.

Refactoring is the process of identifying code-level patterns that facilitate their transformation into more efficient alternatives. Using tools to locate code fragments that can automatically improve, replace, and document will maximize the refactoring process.

The authors of this chapter focus on energy-greedy patterns that have been automatically refactored in a large-scale empirical study involving 600+ applications. The problem and solution for each pattern are provided by clear descriptions. Additionally, concrete instances are given for each pattern, and snippets are oriented toward Android application developers. Discussed software patterns were split into 3 big categories, which were viewed by source code, design, and architecture level.

First is the code-level pattern. There are several similar problems, which solutions can be summarized as :

- Using static variables
- Always close used method
- Always release the power state of the device
- Prevent standby
- Remove useless parameter
- Reuse data
- Never call unnecessary a method

The second pattern tries to provide solutions to the design issues of energy efficiency. There are several similar problems, which solutions can be summarized as :

- Use a dark UI color theme
- Increase the waiting time for failed re-connection.
- For minimalization, give users independence to choose what data, resources, services, and features they need.
- Start the resource just before starting the applications.
- User battery-saving mode
- Use push notifications to get updates
- Shut down inactive resources and service

- Transmit and save only critical data
- WiFi over cellular connection
- Combine multiple operations in a single one to optimize tail energy consumption
- Caching to lower the data transmission

Lastly, the patterns to summarize best-practice for Object-Oriented. The pattern that was studied is the Gang of Four patterns. However, design pattern instances (like any design) have effects on quality attributes. Moreover, the study results for this pattern showed that the non-pattern solutions might consume less energy in some testing and more in other testing attempts.

Parts of a system are rarely islands, isolated from each other, and rather comprise a network of interconnected components. Therefore, in order to get the best result, multiple patterns should be used simultaneously. Developers should be aware of energy-efficiency strategies to decide which pattern they need to use when developing sustainable software systems.

Conclusion

With the advancement of mobile systems, developers face the need for patterns to solve frequent issues. This chapter answers the basis of developers choosing the pattern that suits them best.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 6 : Small Changes, Big Impacts: Leveraging Diversity to Improve Energy Efficiency” Essay. In Software Sustainability. Cham: Springer, 2021.
11th February 2023

Introduction

The sixth chapter of Calero et al.’s “Software Sustainability” discuss about to use of software diversity as a tool to build more energy-efficient software. Software diversity can expand the design and implementation options available for developers.

Body

ICT was estimated to be responsible for 4.7% of the world’s electrical energy consumption. Although the usage is mostly for lowering the energy consumption of another field, this number still can not be ignored. Along with the growth of the mobile application market, mobile battery become a new focus for energy efficiency.

When talking about software energy consumption, we should pay attention to the hardware platform, the context of the computation, and the time spent, as these three factors parallelly affect the number of energy consumed by the software.

There are two approaches to gauging energy consumption at different levels of granularity. The first group of techniques uses power measurement hardware to obtain power samples. Even though this technique provides a dependable number, the number presented is high-level and does not detail which software consumes how much energy. The second one uses software-based techniques to predict how much energy an application will consume at run time, called the energy estimation technique. This technique provides clearer and more dependable detail than the previous technique. However, to acquire the number, programmers need deep knowledge of how to use these low-level registers.

With previous approaches, Oliveira and the team created the so-called energy profiles and attempted to make recommendations by leveraging these profiles. An energy profile is a number that can be used to compare similar constructs under the same circumstances. Energy profiles for collections can be produced by executing several micro-benchmarks on different collection operations, aiming to gather information about the energy behavior of these programming constructs in an application-independent way.

The study found that different thread management approaches, e.g., per-core threads, thread pools, and work-stealing, have diverse, significant, and hard-to-predict impacts on energy consumption.

Conclusion

This chapter advocates developers to implement software diversity for making energy-efficient software systems. When developing a system, diversity can be reached by alternating between readily available, diversely designed pieces of software implemented by third parties.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 7 : Tool Support for Green Android Development” Essay. In Software Sustainability. Cham: Springer, 2021.
11th February 2023

Introduction

Anwar et al. conducted a systematic mapping study to the green Android development researchers’ states of the art and to identify further research opportunities. The outcome of this chapter will be a set of knowledge about what resources are already available and what is still needed to support green Android development.

Body

This chapter mostly focuses on the literature study of green software. It summarizes the tool support available to improve the green-ability of Android apps in the development and maintenance phases. The term “Android development” refers to the development of applications developed to operate on devices running the Android operating system. These applications can be developed in various languages; however, in this chapter, we focus on Android development in Java.

However, due to the difference in architecture, the support tools used in developing traditional Java-based applications are not so useful in Android application development and maintenance. Therefore, many specialized support tools have been developed to improve the quality of Android apps with regard to maintainability, performance, security, and/or energy.

In order to build effective Android-specific support tools to aid green Android development, we first need to understand what is already available, what is still needed, and how the problems in existing tools can be overcome. Based on published literature, we outlined an explorative analysis of support tools available to either (1) optimize code in Android applications through code smell detection/refactoring and/or (2) optimize reusable code in Android applications through the detection/migration of third-party libraries.

Anwar et al. formulated research questions and then generated two general search queries to be used on the publishers’ websites. After that, they removed duplicate results using Zotero and manually defined inclusion, exclusion, and quality criteria for further screening search results. Finally, they analyzed and reported the findings from the previous works.

The first finding of the literature study is that most tools in the “Detector” and “Optimizer” categories used static source code analysis, which indicates that these tools do not cover dynamic issues such as those related to asynchronous tasks. For the development of better support tools, hybrid techniques encompassing both dynamic and static analysis could be used.

Furthermore, the second finding is that none of the support tools in the “Identifier,” “Migrator,” and “Controller” categories offer any support to developers to aid green Android development. They neither support IDE nor have open-source licenses.

Conclusion

The analysis produced identified tools for detecting/refactoring code smells/energy bugs, which were classified into three categories: Profiler, Detector,” and Optimizer. Additionally, tools for detecting/ migrating third-party libraries in Android applications have been identified and classified into Identifier, Migrator, and Controller. However, this finding still has a drawback, as there is no perfect one-solution for each category.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 8 : Architecting Green Mobile Cloud Apps : Key Considerations for Implementation and Evaluation of Mobile Cloud Apps” Essay. In Software Sustainability. Cham: Springer, 2021.
10th March 2023

Introduction

With the restrictive resource of mobile, several techniques have been proposed by the green computing research community to delegate heavy tasks to the cloud, which has a surplus of resources. However, many of the offloading techniques developed so far are inefficient or only suitable for older, pre-existing mobile applications, because they were designed to address problems specific to those applications. Therefore, the eight chapters of Calero et al.’s “Software Sustainability” proposed solution that builds on the benefits of already existing software engineering concepts. It first examined existing approaches to highlight key sources of overhead in the current methods of MCA implementation and evaluation. And, then proposed minimizing overhead-key architectural considerations.

Body

ICT hardware and software have direct impact on energy efficiencies and indirect impact on carbon emission. “Green software” is therefore used to refer to software applications that efficiently monitor, manage, and utilize underlying resources with minimal or controlled impact on the environment. Green software optimization usually targets two main artifacts: the process and code.

Mobile cloud computing (MCC) is a well-known technique used to address the challenges commonly faced by mobile devices by use of cloud computing as a surrogate. MCC knows as green software approach, due to metrics which MCC uses for mobile devices, such as energy efficiency.

Software systems are often presented in terms of functional and non-functional requirements. Hence, this functionality become the basis of green software research varied themes: performance (response or execution time) and energy efficiency, optimal accessibility and resource efficiency, energy-efficient secure systems, development efficiency and energy efficiency, etc. There are some strategies to enhance software qualities, that will be discussed in next paragraphs.

Algorithmic approaches are techniques that directly apply to or make changes to the software code. These include (a) refactoring for efficient resource usage, (b) use of energy-aware custom runtimes which manipulate the program’s execution or codebase, and (c) green compilers or IDEs.

Augmentation approaches are techniques which employ the strengths of successful computing paradigms—and can be a hybrid of conceptual and algorithmic approaches—to solve a problem in a domain.
(a) Cloud domain : The domain problem which green IT aims to solve within the cloud is overutilization (runtime bloat) or underutilization (runtime waste) of resources.
(b) Mobile domain : To address the mobile domain problem, such as battery life.

Mobile Cloud Applications (MCA) are applications that leverage the MCC technique. The mobile tier of MCA is composed of the mobile device, whereas the cloud tier is popularly implemented as clouds or fogs (cloudlets). In order to convert a mobile application into a Mobile Cloud Application (MCA), it is essential to identify the tasks that can be offloaded or delegated to the cloud. Identification can be achieved either manually or automatically.

To execute the offload-able tasks remotely, the cloud tier can be set up either as a clone of the mobile device (i.e., cloning) or as independent components executed remotely (i.e., partitioning). Cloning techniques are more expensive and consume a lot of cloud resource as they require virtual mobile devices in the cloud. Partitioning would involve having only a copy of the offload-able code in the cloud.

The decision-making feature (decision maker) is a feature in offloading schemes which is used to decide when to offload or when not to offload. Decision making and offloading mechanism should be carefully considered.

Offloading schemes makes decisions by monitoring and learning from the environmental factors of MCA:

- Mobile CPU Availability : Mobile CPU availability is measured in percent and is of particular importance for computation-intensive tasks.
- Mobile Memory Availability : Mobile memory availability is measured in percent and is of particular importance for data-intensive tasks.
- Cloud CPU Availability : Cloud CPU availability is measured in percent and is of particular importance for computation-intensive tasks
- Cloud Memory Availability : Cloud memory availability is measured in percent and is of particular importance for data-intensive tasks.
- Network Bandwidth : Network bandwidth is the average rate of a successful data transfer through a network communication path.
- Network Latency : Network latency is the time interval or delay between request and response over a network communication path.
- Data Size : Data size is the size of the data transmitted over the communication network. It is measured in series of bytes (i.e., B, KB, and MB) and can be achieved programmatically by checking the byte size of the request packet prior to client socket transmission.

Following the green software objective, the green metrics are energy and resource efficiency. In the context of MCAs, the core investigated green metric is mobile energy efficiency, given that the focus of MCA offloading schemes is the optimization of mobile application. However, MCA is two tiered, i.e., also involving the cloud tier, and thus this research presents cloud resource efficiency as another relevant green metric for MCA.

- Mobile Performance : performance is how long it takes an application to respond to an event. The key driver of the advancement in mobile computing is the portability of mobile devices, which is defined by fluidity and ease of operation
- Mobile Energy : energy efficiency is the ratio of useful work done to used energy. It is the amount of energy incurred for executing a task. Energy efficiency is derived from three quantities: power, time, and work done
- Cloud Resource : Achieving cloud resource efficiency in a mobile cloud environment requires care so as not to compromise mobile performance. Resource efficiency in servers (the cloud) is often achieved through load-balancing.
- Software Availability : availability is the probability that a system will be operational when it is needed

Application taxonomy is a classification system used to categorize software applications based on their purpose, features, and characteristics. There are three taxonomies identified from the MCA literature:

- Computation-intensive applications : Computation-intensive applications are a class of mobile applications that are highly or significantly dependent on the computing power
- Data-intensive applications : Data-intensive applications are a class of mobile applications that devote most of their processing time to I/O and data manipulation. Due to the focus on manipulation of data, these applications make more use of memory than the processing power.
- Hybrid applications : Hybrid applications are applications that are composed of different offload-able tasks with at least one computation-intensive task and at least one data-intensive task.

Conclusion

With the advancement of mobile systems, there are more and more strategy for mobile resource efficiency. However, developer should understand how to make decision, in order to fulfilling Sustainability of Software, which is trade-off capability of the software product to meet the current needs of a set (required) functionality, without compromising the ability to meet future needs

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 9 : Sustainability: Delivering Agility’s Promise” Essay. In Software Sustainability. Cham: Springer, 2021.
09th March 2023

Introduction

The ninth chapters of Calero et al.’s “Software Sustainability” explores promise of Sustainability by agile development, as it is part of both the Agile Alliance’s and the Scrum Alliance’s vision. However, not much has been delivered on this promise. Thus, this chapter explores the Agile Manifesto and points out how agility could contribute to sustainability in its three dimensions – social, economic, and environmental.

Body

Information and Communication Technologies (ICT) to address sustainability issues has been investigated in a number of interdisciplinary fields that combine ICT with environmental and/or social sciences. Among these are : environmental informatics, computational sustainability, sustainable human-computer interaction, green IT/ICT, ICT for sustainability, or sustainable ICT for business.

At the organizational level, there are concrete movements aiming to answer the question of how an organization can be more balanced across the three sustainability pillars. Even business world that practicing capitalism, is starting to run into fundamental structural problems.

The two major agile organizations, the Agile Alliance and the Scrum Alliance, both promise in their vision statements that sustainability is one of their core goals:

- Agile Alliance is a nonprofit organization committed to supporting people who explore and apply Agile values, principles, and practices to make building software solutions more effective, humane, and sustainable.
- Scrum Alliance® is a nonprofit organization that is guiding and inspiring individuals, leaders, and organizations with agile practices, principles, and values to help create workplaces that are joyful, prosperous, and sustainable.

Principles of the Agile Manifesto (to understand how these principles can contribute to or guide sustainability) :

1. Our Highest Priority Is to Satisfy the Customer Through Early and Continuous Delivery of Valuable Software
   At the core of this very first principle is continuous learning by focusing constantly on the customer.
2. Welcome Changing Requirements, Even Late in Development: Agile Processes Harness Change for the Customer’s Competitive Advantage
   Focusing on the customer to deliver satisfy-able solution, and broadening the view to be inclusive for customer.
3. Deliver Working Software Frequently, from a Couple of Weeks to a Couple of Months, with a Preference to the Shorter Timescale
   Establishing a short sprint and regular cadence for feedback, and then focus on the learning about the changes in the system from previous feedbacks.
4. Business People and Developers Must Work Together Daily Throughout the Project
   Continuous collaboration enables self-organization around the system created and allows to discover and actually meet the market needs.

5. Build Projects Around Motivated Individuals.
   Give Them the Environment and Support They Need, and Trust Them to Get the Job Done
6. The Most Efficient and Effective Method of Conveying Information to and Within Development Team Is Face-to-Face Conversation
   Face-to-face conversation provides transparency particularly from the social and economic point of view. This method will build community, where everyone is involved in the process of making the product sustainable.
7. Working Software Is the Primary Measure of Progress
   Precautionary principle (Regularly examine the system for unintended consequences)
8. Agile Processes Promote Sustainable Development. The Sponsors, Developers, and Users Should Be Able to Maintain a Constant Pace Indefinitely
   Developers should have a constant pace in mind. When they are developing the product or service, it should not lead to burnout of the developers (social) or to overspending financially (economic), but should reduce resources (environment).
9. Continuous Attention to Technical Excellence and Good Design Enhances Agility
   Keeping technologically up-to-date and taking into account new learning and development that enable good design: now good in the sense of sustainability.
10. Simplicity—the Art of Maximizing the Amount of Work Not Done—Is Essential
    Clear transparency and wise contemplation of features that are needed (and used) and those that are not.
11. The Best Architectures, Requirements, and Designs Emerge from Self-Organizing Teams
    All team members will be expected to give equal contribution for developing the system and will get the same fair chance for progressing their skills and career.
12. At Regular Intervals, the Team Reflects on How to Become More Effective, Then Tunes and Adjusts Its Behavior Accordingly
    Team will be expected to do regular retrospectives, with taking sustainability as one of the topic.

Conclusion

This chapter examined that agility can contribute to sustainability. As we have seen, the Agile Manifesto in general and the principles in particular can provide guidance to sustainability. Because, taking all three dimensions of sustainability into account leads to higher complexity, so an agile approach also comes in handy for addressing this complexity by breaking down the problems and using an inspect and adapt approach for making them simpler.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 10 : Governance and Management of Green IT” Essay. In Software Sustainability. Cham: Springer, 2021.
09th March 2023

Introduction

Through what is known as Green IT, organizations are implementing measures to reduce the environmental impact of their IT. However, organizations are conducting these Green IT implementations at their own discretion, due to the lack of guidelines, standards, or frameworks in this regard.

With the objective of helping organizations, the tenth chapter of Calero et al.’s “Software Sustainability” presents a framework that guides the way in which organizations should properly govern and manage Green IT. The framework is called Governance and Management Framework for Green IT (GMGIT)

Body

The GMGIT is divided into four main sections that address all the characteristics for the implementation, assessment, and improvement of Green IT in organizations:

- Section I. This first section includes the conceptual basis needed to understand the context of the framework.
- Section II. This section is the main part of the framework, through which the elements to implement a correct governance and management of Green IT are defined and established.
- Section III. This third section proposes a Green IT audit framework, which includes the steps that an auditor must follow, as well as the aspects that must be considered to audit Green IT.
- Section IV. The fourth and final section includes a maturity model for Green IT based on the ISO/IEC 33000 family of standards

GMGIT has identified nine principles organized into three groups:

- Give support to the business.
  - Give quality and value to the stakeholders.
  - Comply with relevant legal requirements and regulations.
  - Provide convenient and precise information on the functioning of Green IT.
  - Evaluate current and future IT capabilities.
  - Promote ongoing improvement in Green IT.
- Reduce the environmental impact.
  - Adopt a strategy that is based on the efficient use of IT resources.
  - Develop the systems in a sustainable way.
- Foster responsible behavior toward and in favor of the environment.
  - Act professionally and ethically.
  - Foster a positive culture of Green IT.

GMGIT has identified the following six policies that should be considered, adapted, and implemented in Green IT :

- Policy of Green IT.
- Policy of acquisition, development, and maintenance of IT systems.
- Policy of resource management.
- Policy of compliance.
- Policy of conduct.

- Policy of asset management.

GMGIT we have identified two main roles that must be established to be in charge of the Green IT functions:

- Chief Sustainability Officer (CSO). Its role is being the main representative of Green IT in the organization. The CSO has the responsibility of controlling and supervising the management of Green IT.
- Sustainability Steering Committee (SSC). This committee should be responsible for verifying that Green IT performs correctly, as well as that the policies, plan, strategy, and other governance aspects in this regard are applied and followed effectively and efficiently.

GMGIT has defined that the people who are in charge of/responsible for Green IT should have these skills and competencies :

- Governance of Green IT.
- Strategy of Green IT.
- Architecture of Green IT.
- Operations of Green IT.
- Evaluation, tests, and compliance of Green IT.

Behavior is the key pillar, because it determines the culture of the organization itself and the approach to Green IT. For this area of Green IT, in the GMGIT we have defined eight behaviors that organizations should include in their culture:

- Behavior 1. Green IT is put into practice in day-to-day operations.
- Behavior 2. The importance of the policies and principles of Green IT is respected.
- Behavior 3. The members and stakeholders are provided with enough detailed guidelines on Green IT, and compliance with these is encouraged.
- Behavior 4. The members and stakeholders of the organization are responsible for the proper use of Green IT.
- Behavior 5. The members and stakeholders of the organization identify and communicate new Green IT needs.
- Behavior 6. The members and stakeholders of the organization are receptive when identifying and managing new Green IT challenges.
- Behavior 7. The organization is committed to, and aligned with, Green IT.
- Behavior 8. The organization acknowledges the value brought to it by Green IT.

In order to conduct an appropriate decision-making in the implementation, operation, and maintenance of green-IT, organization should standardize how the information are shared:

- Policies and principles of Green IT.
- Plan and strategy of Green IT.
- Requirements of Green IT.
- Budget of Green IT.
- Awareness material of Green IT.
- Review reports of Green IT.
- Scorecard of Green IT.

The services, infrastructure, and applications are translated into a set of service capacities in order to provide a proper functioning of Green IT in the organization. the following activities or services that should be provided for Green IT to perform correctly have been identified in the GMGIT:

- Provide an architecture of Green IT that is appropriate to the needs and capabilities of the organization.
- Provide awareness of, and training in, Green IT.
- Provide evaluations and tests of Green IT.

Additionally, GMGIT gives proper guidelines for software development process and auditing too.

Conclusion

Even though organizations, and governments around the world to wake up and come together to defend a sustainable development, these efforts are mostly still futile. Thus, contribution from all areas of knowledge to guide organizations compulsory to achieve the agreed sustainable goals. Therefore, GMGIT advocates to help organizations to establish the governance and management bases that will support the implementation, assessment, and improvement of sustainable practices in and by IT (well-known as Green IT).

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 11 : Sustainable Software Engineering:Curriculum Development Based on ACM/IEEE Guidelines” Essay. In Software Sustainability. Cham: Springer, 2021.
10th March 2023

Introduction

The eleventh chapters of Calero et al.’s “Software Sustainability” discusses how sustainability can be included in various courses of the Software Engineering (SE) curriculum by considering ACM/IEEE curriculum guidelines for the SE curriculum, so that SE students can attain knowledge on sustainable software engineering.

Body

The current industrial growth and the increasing adoption of ICT threaten the future of sustainability and cause environmental issues. Sustainable software is defined as the software whose direct and indirect negative influence on economy, society, human beings, and environment are minimum. Therefore, Sustainable software engineering is developing software through a sustainable software engineering process which satisfies the purpose of sustainability.

Green software suggests more efficient algorithms will take less time to execute and thus will lead to sustainability. However, sustainability is mostly still considered as an additional feature, one of not important non-functional requirements. Which in turn, make the skill sustainable software developing rarely taught as basic course in software engineering classes.

For incorporating sustainability to SE classes, there are several challenges. The major challenge come from the field of teaching, because this field needs a clear key competences list for each course. Therefore, identifying key competences in sustainability may be the first step towards sustainability inclusion in higher education.

Sustainability knowledge should be integrated in a curriculum by linking the concept of sustainability to a particular field of study rather than offering separate courses on sustainability, so that students can get sufficient exposure to different components of sustainability issues in the software development life cycle.

These are examples of how sustainability can be integrated to current SE courses and process.

- Fundamental Concepts of Sustainability
  - Sustainability Theory : Understanding the concept of sustainability in software and its various parts so as to be able to apply it in different stages of software develop ment and deployment and operations stages in the organization.
  - Sustainability Analysis : Understanding of rigorous analyses of sustainability issues in software development
- Core SE Courses
  - Software Requirements Engineering : Sustainability inclusion in requirements elicitation and analysis process is crucial. Therefore, it is important to understand how to include sustainability during requirements elicitation process.
  - Software Architecture and Design : How to apply sustainability in different kinds of software architecture and design issues
  - Human-Computer Interaction Design : Human-Computer Interaction (HCI) is part of many information technology and software applications.
  - Software Modelling and Analysis : It is important that system modelling for complex software systems should be done from a sustainability perspective by using available

tools, because these tools can assist in understanding how to incorporate sustainability into stakeholders' requirement scenarios.

- Software Process : Software process improvement should include sustainable software engineering processes along with Agile and DevOps approaches
- Software Verification and Validation : An optimized approach in ensuring sustainability in software engineering is the software verification and validation process
- Software Quality Assurance : Sustainability issue should be part of software process improvement and quality assurance process
- Software Project Management : It includes the planning and controlling phases of sustainability activities along with sustainability policies to ensure an efficient process
- Software Construction and Evolution : Software evolution is a continuous refactoring process.Therefore, this study goals is for ensuring sustainability in software construction and evolution.
- Software Security : Security and safety during the development of complex software systems is crucial. So security and safety are now an integral part of the set of nonfunctional requirements which lead to software quality.

- Technical Elective Courses
  - Internet of Things (IoT) : IoT has the ability to combat climate change towards green environment.
  - Cloud Computing : Cloud computing provides more efficient use of computing power and is advantageous for environmental sustainability
  - Web and Mobile : Systems Sustainability and page speed are correlated. When your website runs more efficiently it consumes less processing power thus less energy and leaves a lower carbon footprint. Moreover, sustainable design will make the produced software more efficient, accessible and inclusive.
  - Sustainable Data Center Green data centers or sustainable data centers help in reducing carbon footprint, design and deployment of data store, and applications to operate in energy-efficient ways.
  - Tools for Software Sustainability : Tools must be introduced to assist different stages of software development
- Nontechnical Elective Courses
  - Global Professional Practice/Social Responsibility : Students should be aware how carbon footprint, CO2 emissions, global warming is a matter of concern
- Project-Based Courses and Industrial Practice/Internships

  Sustainability can be included as a learning outcome for project-based courses, such as internship, final thesis, or mini projects.

Conclusion

Sustainable software engineering is an emerging paradigm and significant for society in terms of the environment. Therefore, proper feedback system is required between educators and university administrators to show the value and significance of the integration of sustainability. Moreover, professional practices should be part of the SE curriculum, which can include a sustainability component. Lastly, the proposed curriculum development should be easily customized and introduced as part of an undergraduate or graduate-level software engineering curriculum. These steps need to be taken, in order for each SE professional produces by the courses can understand that sustainability is a key factor, not an additional feature, and have sufficient skillsets to integrate sustainability to their works.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 12: The Impact of Human Factors on Software Sustainability” Essay. In Software Sustainability. Cham: Springer, 2021.
3rd April 2023

Introduction

One of key challenge in software solution for business automation is attaining sustainability. Consideration of human factor has big impact on this area. Therefore, the objective of this chapter is to provide a systematic and comprehensive overview of how stakeholders view human factors to be impacting sustainability..

Body

Human factors include critical social activities such as leadership and communication. Meanwhile, software sustainability means "software you use today will be available—and continue to be improved and supported in the future".
Software sustainability can help us achieve a number of useful goals, such as:

- Operational efficiency: Sustainability of software used both in industries and by individuals should be a natural part of the overall performance management practice.
- Desirable reputation of software product: To remain competitive, companies need to make innovation their top priority.
- Reduced cost: From a business perspective, investing in a software which is sustainable will guarantee cost reduction and profit increase in the long run
- Accelerated progress of scientific software: Strategic plan to conduct the necessary activities of training, prototyping, and implementation with a goal to create improved and more sustainable software.

From previous studies, the authors of this chapter found that even though software practitioners view some human factors as critical to sustainability. However, most of the sustainability techniques are rarely applied due to lack of knowledge of the software community.

Therefore, the study that have been done by the author of this chapter focuses on detecting the important human aspects to sustainability. Next, their negative impact is analyzed via feedback from software practitioners, using smell detection tool. To calibrate Architectural Technical Debt (ATD) detection tool, the authors establish reliance on the tool, and followed by taking community feedback to analyze the impact of smells. After that the result was compared with predefined set of smells.
The following human factors are identified:

- 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.
- Communication: Lack of effective communication can break the sustainability of a software project.
- Leadership qualities: Leadership includes the skills through which a project manager handles change management, timeline management, cost management, employee turnover, etc.
- Difficult deadlines: This stressful factor may turn software engineers for trying to complete the software development rather than focusing on the sustainability manifestos.
- Peer pressure: This stressful factor may turn software developers for trying to finish the project early to get into superiors’ good books, and bypassing the sustainability benchmarks of the software.
- 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.

After the survey, 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.

Conclusion

The authors of this chapter discuss various types of human activities which aim to make software sustain able. They compare and contrast how these approaches apply from the perspective of software engineers. At the end, they conclude that to achieve sustenance, human, environmental, and economic factors need to be considered simultaneously. Integrate software practitioner’s feedback to identify the negative impact of the smells on architectural debt is mandatory.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 13 Social Sustainability in the e-Health Domain via Personalized and Self-Adaptive Mobile Apps” Essay. In Software Sustainability. Cham: Springer, 2021.
3rd April 2023

Introduction

Social sustainability is the dimension of sustainability that focuses on the support of current and future generations to “have the same or greater access to social resources by pursuing social equity.” An important domain that strives to achieve social sustainability is e-Health (mobile apps). E-Health mobile apps is available for many purposes, such as lifestyle improvement and mental coaching. However for now this kind of apps are rarely tailored toward individuals’ needs. Therefore this chapter will discuss about how to build an e-Health system that have self-adaptive feature.

Body

Better innovation, such as combining personalization and software self-adaptation to provide users of mobile e-Health apps with a more engaging and effective experience may invoke user interested in maintaining engagement with the app, and therefore also help the user achieve better physical and mental conditions.
The main design drivers that make the proposed reference architecture unique are:

- The combination of multiple Monitor–Analyse–Plan–Execute (MAPE) loops
- A dedicated goal model for representing health-related goals via a descriptive concise language accessible by healthcare professionals
- Online clustering algorithm for efficiently managing the evolution of the behaviour of users as multiple time series evolving over time.

Goals model have been used in many areas of computer science for a long time, for example in AI training. Self-adaptive systems is presented to express the desired runtime behaviour of systems execution. The main goal of the AI Personalization Adaptation is to keep track of the clusters evolution and to enable the creation of new User Processes. Then, in Monitor phase, AI personalization adaptation monitors the macro-clusters. Then it determines if there are changes in monitored macro-clustered, using clustering history database. If there are any changes, the system will enter the Plan process with the help of Domain Expert.

Conclusion

This work emphasizes how personalization and self-adaptation within the e-Health domain can be beneficial in addressing social sustainability. The reference architecture achieved self-adaptation on three levels:

1) adaptation to the users and their environment,
2) adaptation to smart objects and third-party applications, and
3) adaptation according to the data of the AI-based personalization, ensuring that users receive personalized activities that evolve with the users’ runtime changes in behaviour.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 14: Human Sustainability in Software Development” Essay. In Software Sustainability. Cham: Springer, 2021.
3rd April 2023

Introduction

The focus of the fourteenth chapter of Calero et al.’s “Software Sustainability” focuses on the effect of ethically outsource marginalized people, and the benefits and challenges for client organizations.

Body

There are three main actors in this chapter : "marginalized people" (disadvantaged individuals who have few opportunities for employment), outsourcing supplier (sometimes referred to as the providing organization), and "client organizations” (outsourcing companies).

Client organizations do not always reap benefit, due to traditional outsourcing practice (outsourcing practices that do not consider corporate social responsibility, focuses on maximizing profits). Traditional outsourcing practice may end-up in high staff turnover and poor marketing (due to bad brand reputation, caused by negative publicity about working conditions) can result in an increase of the total costs.

These new approaches implement corporate social responsibility (CSR). CSR is a theory that emphasizes that companies should implement policies and practices toward the good of society.
Three outsourcing approaches that consider CSR:

- Impact sourcing, sometimes referred to as “social outsourcing” or “developmental outsourcing,” is the act of outsourcing to marginalized people who would otherwise have difficulty finding employment.
- Ethical outsourcing, also referred to as “socially responsible outsourcing,” occurs when the client organization imposes minimum social and environmental standards on the organization supplying the outsourced service.
- Fair Trade Software is a form of software development collaboration with teams from both developing and developed countries with a focus on the transfer of knowledge from the teams from developed countries to the teams from developing countries

There is limited data available to assess the benefits of Fair Trade Software. One of the potential benefits of the Fair Trade Software model, for teams in developing countries, is capacity building. No literature was found on the benefits for teams in developed countries, but based on an interview with the founder of the Fair Trade Software Foundation, Andy Haxby, we discovered that the benefits for teams in developed countries are motivation and self-satisfaction for individuals and marketing opportunities and brand enhancement for the companies they work for. However, implementation of Fair Trade Software still have multiple problems, such as compatibility of tools and technologies, data transfer, cultural and language barrier, and interpersonal relation between developers.

Conclusion

There are three key advantages of impact outsourcing. First, impact sourcing gives marginalized people the chance to earn money, advance their careers, and form social networks. Second, compared to conventional outsourcing, impact sourcing offers businesses further chance for growth, reduces costs, lowers employee turnover, and improves corporate social responsibility. Third, impact out-sourcing protect brand reputation and can enhance stakeholder management. However, there are also drawbacks, including the potential for reduced company competitiveness (investments needed), and employee stress.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 15: The Importance of Software Sustainability in the CSR of Software Companies” Essay.
In Software Sustainability. Cham: Springer, 2021.
3rd April 2023

Introduction

The fifteenth chapter of Calero’s Software Sustainability book focuses on how well the policies of companies that develop software are aligned with Software Sustainability, as well as to give recommendations on including specific actions in their CSR to promote Software Sustainability.

Body

Business that fails to include sustainable development may receive public criticism and lose market legitimacy. Therefore, fulfilling sustainable business models (SBMs) is not just a passing fancy but a mandatory requirement. This action may be done under “strategic sustainability”.

To find out if, and to what degree, software companies are concerned about the environmental aspects of the sustainability of the software they develop, the authors analysed the CSR of several leading software companies: Apple, Microsoft, IBM, Oracle, SAP, Symantec Corp, EMC, Hewlett-Packard, VMware, CA Technologies.

The template we built in order to collect from the companies’ CSRs data about their actions on software sustainability includes the following sections:

- Category, which includes the general categories used to group together related actions, such as “People,” “Planet,” etc.
- Subcategory, to cover specific categories for the actions, for example, “Empowering communities” or “Empowering employees,” which fall under the general category “People.”
- Action, which covers the specific actions carried out in the context of the CSR, for example, the action “Employee feedback counts”.
- Sustainability dimension, used to classify a given action according to the particular dimension or dimensions (dimensions consist of environmental, human, and economic) in which it is applicable.

After that, each researcher (the four authors of this chapter) was responsible for filling in the templates of two to three companies. Then, each researcher reviewed the templates that had been filled in by the other researchers. Finally, after discussion of the filled and agreed form, the researchers met a proposed list of actions that could be valuable for software companies from the point of view of sustainability.

Having defined the actions specific to software sustainability that the software industry should be including in their CSR, the authors then analyse the CSR of a specific medium-sized company in Spain. One of the main points of this company is a concern for the quality of life of its workers and high chance of employee mutation within Spain. The company’s CSR is based on three basic foundations: the company, the people, and the planet. They have drawn up their own code of ethics which is based on the principles of honesty, integrity, and respect. And received HappyIndexAtWork Certification.

They are supporting mental and physical health of their employee with various activities, like recreational retreat or healthy breakfasts. They raised employees’ awareness of the environmental importance and impact of their actions with various campaign, such as removing plastic utensils. This company's concern for their employees and planet are clear, but from the CSR quantifying method the authors were made (based on the method that they found), the company still has big homework, in consideration to software sustainability.

Conclusion

Organizations around the world are more aware for socially responsible behavior, and sustainability is a core aspect of this. Environmental aspects of software systems is a key factor in sustainable development, and any company aspiring to be considered as a first-class corporate citizen should provide for it in their CSR.

Cindy Aprilia
Prof. Jari Porras - CT10A7004 Sustainability and IT
Calero, Coral, M. Ángeles Moraga, and Mario Piattini. “Chapter 16: Sustainability ArchDebts: An Economics-Driven Approach for Evaluating Sustainable Requirements” Essay. In Software Sustainability. Cham: Springer, 2021.
3rd April 2023

Introduction

The goal of last chapter of Calero’s Software Sustainability book is to develop an objective decision-support framework for reasoning about sustainability requirements in relation to architecture decisions under uncertainty. Therefore, the authors proposed economics-driven architectural evaluation method called Cost Benefit Analysis Method (CBAM).

Body

An architecture is a set of design decisions that intends to meet all of the functional requirements while optimizing the desired quality attributes. Incomplete knowledge of the system of its environment and capacities, poor estimation or misperceptions of system requirements conditions may start problems in developing architectures. Software developer teams tends to forget that architecture design parameters are stochastic, and not deterministic.

Therefore, the authors proposed an economics-driven architectural evaluation method called Cost-Benefit Analysis Method (CBAM) and integrates the principles of modern portfolio theory to evaluate software architectures for sustainability. The analysis can provide insights on the extent to which architectural design decisions overperform or still have technical debt (Technical debt is trade-off between the short-term benefits of rapid development and the long-term costs of poor code quality).

Existing sustainability requirement assessment methods are generally applied at the architectural level, once the requirements have been selected and specified. This results in a gap between understanding the value of sustainability requirements and the impact on the architecture. Therefore, this work is focused on the requirement level, linking architectural decisions with requirement analysis, instead of being only focused on architectures.

As portfolio viewpoint is needed to manage risks, survey of portofolio management was undertaken. A number of requirements that any useful model ought to address were identified:

- Linking technical decisions to value (cost, value, and risk).
- Evaluating sustainability as an architecturally significant requirement.
- Evaluating technical debt in the requirement phase of system design.
- Evaluating the impact of sustainability on architecture selection decisions.
- Evaluating the impact of uncertainty in cost estimates as a risk.
- Evaluating the impact of obstacles to operationalizing goals as a measure of risk

Value breaks down into three components: risks, costs, and benefits. A traditional value-based approach consists of : (1)identification of the decision parameters, (2)identification and analysis of risks attached to those decisions, (3) a cost-benefit analysis is done to quantify the costs and the benefits of the decisions, and at the end (5)the value of an architecture plays a significant role in determining how well it meets the initial requirements. Therefore, it is necessary to explicate the relationships between the sustainability requirement and the value components.

Advancing CBAM, the author proposed new methodology, which were designed to support developers and analysts in systematically estimating the value of prioritized or selected architecturally significant requirements and architectures:

- Step 1: Elicit and Prioritize the Goals, Cost, and the Desired Sustainability Threshold for These Goals
  defining the goals against which to evaluate the decisions.
- Step 2: Develop Architectural Strategies for the Goals, Elicit Their Parameter Values, and Define the Architectural Decision Model
  defining the goals against which to evaluate the decisions, are identified using a goal-oriented requirement engineering method.
- Step 3: Determine Architectural Strategies’ Impact on Sustainability

Relationship with sustainability goals is determined, after the options have been developed and their parameter values elicited.

- Step 4: Elicit Sustainability Goals, Design Decisions, Obstacle, and Risk Analysis
  Goal-oriented requirement engineering (GORE) approach are used for specifying and refining challanges assessment.
- Step 5: Determine the Expected Benefit of Each Design Option
  Benefits on goals/scenarios along with the sustainability goals was tracked using CBAM principles here.
- Step 6: Analyzing the Costs, Benefits, Risks, and Value Using Portfolio Thinking
  Goals/scenarios along with the sustainability goals was tracked using CBAM principles here.
- Step 7: Identify the Optimal Architecture, Calculate Debt, and Rank Other Architectures
  The architecture is evaluated by considering the amount of sustainability debt that each architecture may incur as the deciding factor.

Proposed framework are more debt-aware, optimized and holistic on evaluating architectures for sustainability. In particular, than traditional CBAM. This concept offers a better coverage at optimizing architectural investment decisions and ranking candidate architectures based on debt.

.The results of the implementation of the proposed framework may be evaluated according to this steps:
Step 1: Elicit and Prioritize the Goals, Cost, and the Desired Sustainability Threshold for These Goals
Step 2: Develop Architectural Strategies for the Goals, Elicit Their Parameter Values, and Define the Architectural Decision Model
Step 3: Determine Architectural Strategies’ Impact on Sustainability
Step 4: Elicit Sustainability Goals, Design Decisions, Obstacle, and Risk Analysis
Step 5: Determine the Expected Benefit of Each Design Option
Step 6: Analyzing the Costs, Benefits, and Risks Using Portfolio Thinking
Step 7: Identify the Optimal Architecture, Calculate Debt, and Rank Other Architectures

Conclusion

For tackling unnecessary technical debts, the authors introduced a new dimension of using sustainability debt as a decision factor for elaborating and managing requirements through architectural evaluation. They incorporated the debt analysis at the requirement analysis and early architectural decision levels.

However, requirements tend to change over time. But some process can be reiterated with clear standard. The ability of an architecture to respond to change can also be modelled as a dimension for technical sustainability subject to the extent to which we can anticipate changes. The evaluation method largely depends on the expertise of the analysts and developers. Therefore, these summaries may be presented:

- The method to guide the evaluation needs to be customizable. It provided useful guidance to conduct the case studies.
- Good understanding of organizational requirements is vital for the success of selection process.
- Sustainability debt is subjective.
  It can be affected by the following: (1) the decision makers and their expertise (e.g., stakeholders), (2) the elicited parameter values, (3) the importance of the requirements to the system design, and (4) the potential use and value of the system. The threshold of debt depends on the acceptance of stakeholders too.
- Value is subjective. It depends on the different variables.