Exploiting goal-oriented requirements models for increasing energy awareness: A research preview

Denisse Muñante[1], Anna Perini[2] and Angelo Susi[2]

[1]ENSIIE & SAMOVAR, Évry (France)
[2]Fondazione Bruno Kessler, Trento (Italy)

Abstract

Energy efficiency of software systems has been investigated by the Software Engineering research community along different aspects. For example, conceptual frameworks and guidelines have been defined to help increase stakeholders' awareness about energy efficiency; design patterns and tactics have been proposed to guide software system architects when evaluating possible alternative solutions; and techniques for collecting energy consumption data while the software is running have been developed. Research on practical methods and tools at support of requirements engineering for the development of software that meets energy efficiency requirements is still limited.

In this research preview, we introduce an approach that exploits goal-oriented requirements models and software testing to explore critical scenarios for energy consumption, and identify model factors that may recur when energy consumption is greater, we call them "model indicators" of energy consumption. Techniques for performing energy consumption measurements of running software are used. The resulting method connects artefacts at requirements and run-time levels. The identified model indicators can contribute to the prediction of energy consumption of software at requirements level, thus increasing energy awareness of the analysts and developers, and support them to take more energy-friendly decisions during software development and evolution. We describe this method with the help of an illustrative example. Next steps in the proposed research are presented.

1. Introduction

Optimising energy consumption of software systems has became a primary goal. Increasing attention to building energy efficient software systems is given by the software engineering research communities, under specific umbrellas, for instance, the GreenIT research community [1]. In Requirements Engineering (RE), conceptual frameworks and guidelines have been proposed to help increase stakeholders' awareness about sustainability requirements for software systems, including energy efficiency requirements [2, 3, 4]. Goal-oriented (GO) modelling for representing and analysing sustainability requirements (including energy optimisation) has been proposed in several work, as discussed for instance in [5]. Architectural patterns and tactics have been proposed to guide software system architects when evaluating possible alternative solutions [6, 7]. Techniques for measuring software energy consumption have been made available as well [8, 9]. Up to our knowledge, there is still limited availability of practical methods and tools to be used since the earlier phases in software development.

iStar'23: The 16th International i* Workshop, September 4, 2023, Hannover, GE
denisse.munantearzapalo@ensiie.fr (D. Muñante); perini@fbk.eu (A. Perini); susi@fbk.eu (A. Susi)
0000-0003-2621-8342 (D. Muñante); 0000-0001-8818-6476 (A. Perini); 0000-0002-5026-7462 (A. Susi)

In our research we aim at providing methods that can help software developers and requirements engineers to increase their awareness about energy consumption and the interplay among different quality requirements.

Towards this objective we leverage on: *(i)* GO requirements modelling and analysis, which has been largely exploited to support the evaluation of alternative ways to achieve high level goals, taking into account the impact in terms of positive and negative contributions to quality goals [10, 11]; *(ii)* the Goal-Oriented Software Testing methodology (*GOST*), which has been proposed by Nguyen et al. [12], that provides guidelines for the systematic derivation of test suites from a GO requirements or design model; and *(iii)* the potential of software testing (ST) in providing data useful to assess, and eventually refactor, design- and requirements-time artefacts, including requirements models. The resulting method can be considered a REST method [1], according to the taxonomy proposed in [13].

The ultimate objective is to help increase requirements engineers' awareness about possible issues of energy consumption in the modelled system, and support them in identifying parts of requirements models that may need refactoring towards improving energy efficiency of the system.

In the rest of the paper, we introduce the proposed REST method. Examples taken from the Tele Assistance Service (*TAS* for brevity) case study [14] are used for illustrative purposes. The envisioned application of the method is then outlined, together with future research.

2. The method

The proposed REST method includes a set of activities that are performed iteratively by the software system development team. They are depicted in Figure 1. Four roles are involved with responsibility on specific activities, namely the **Requirements engineer**, the **Developer**, the **Test designer**, and the quality assurance team (**QA team**). The **Requirements engineer** analyses the collected domain knowledge and builds a **GO requirements model**. S/he focuses on the analysis of the stakeholders' goals, including quality goals, whose achievement may be delegated to system actors. On the other side, the **Developer** builds the **running software application** following design decisions that conforms to the previous goal model. Traceability among the different artefacts in the development process, from requirements GO models to code is assumed to be established and maintained. Following the GOST methodology and focusing on **acceptance** and **integration** test cases derivations, **scenarios** from GO models are derived in a systematic way. Thus, a scenario represents the alternative path, composed of sub-goals and sub-tasks, that achieve the satisfaction of stakeholders' goals. Preconditions and post-conditions complete the definition of scenarios. The **Test designer** is then responsible to validate scenarios derivations and to select criteria to setup **test cases** that are inferred from scenarios. We focus on acceptance test cases, which provide a mechanism to define and assess system's external qualities, and integration test cases, which assess system's actors dependencies. The quality assurance team, **QA team** executes the test cases on the **running system**, which is instrumented with energy consumption metering system, as for example JoularJX [15], which is a Java-based software energy monitoring tool. Then, energy consumption data are analysed.

[1] That is methods that connect RE and Software Testing (ST) [13].

In particular, they are used to assess energy consumption related to **alternative requirements** modelled in OR refinements. The collected data is then aggregated and associated back to requirements GO models.

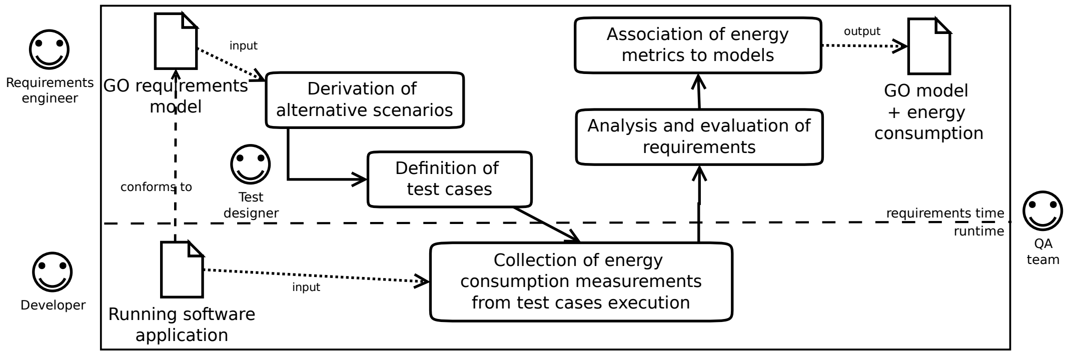

Figure 1: Process of the proposed REST method

Illustrative example: Figure 2 depicts an excerpt of the iStar2.0 early requirement model of the *TAS* system [14]. The patient (PDC actor) has the primary goal `Therapy continuously adapted`, that requires the exchange of information (the resources `Vital Parameters` and `Updated therapy`) with the TAS actor, i.e., the Tele Assistance Service, which is the main system actor. Among the relevant patient's quality goals, we distinguish having prompt and easily access advice, as well as having an energy efficient system. The impact of the main (functional) goals to such quality goals are modelled via contribution relationships using qualitative labels (e.g., hurt, make). A deeper assessment, such as what's the relative weights among such contribution relationships should be further analysed. The proposed REST method will help in regards to the contributions to the energy efficiency quality.

Pursuing the analysis of the TAS actor, alternative solutions for goal achievement are explored via goal refinements, which explicit sub-goals, tasks and their **utilisation of resources** (not shown in Figure 2 for space reasons), and provide rationale for **dependencies with external actors**. For example, the TAS actor interacts with a medical assistance actor, MAS, to analyse the vital parameters from the patient and with an alarm system, AS, that mange the alarms from the patient. The analysis of these dependencies helps assess how collaborative goals/tasks influence the total energy consumed by multi-actor systems. Indeed, as argued in [16], besides computing resources such as memory, CPU, files, networks, etc. also interactions of IoT entities are important elements to be analysed at requirements time with regard to energy consumption. We derive scenarios and associate test cases for GO model element that are involved in contribution relationships to the `Energy efficiency` quality goal. We can consider for instance the goals `therapy continuously adapted` and `direct help request managed`. Notice that the latter is refined in two alternative tasks that may have minor positive impact (`help?` contribution) or major negative impact (`hurt?` contribution) to energy efficiency. Running energy metering technique while executing the derived test cases provides data on energy consumption of code corresponding to alternative paths in the requirements GO model. In a first experiment, we focused on two testing scenarios corresponding to the task `pressing panic`

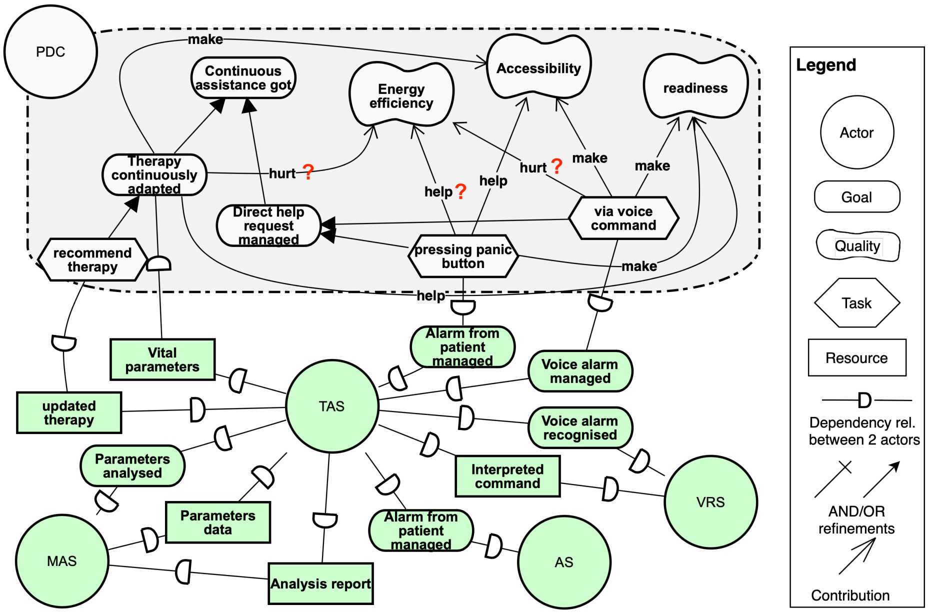

Figure 2: Early requirements GO model of the Tele Assistance Service. iStar 2.0. notation [17].

`button` and the resource dependency `vital parameters`, for which we executed and tested the *TAS* system, using JoularJX to measure energy consumption. To avoid fluctuations on the gathering energy consumption, we run each scenario 120 times. This experiment was conducted on an HP ProBook laptop (Intel Core i7-1165G7) running Ubuntu 20.04.6 LTS, Java 11, and JoularJX 2.0. The analysis of collected data allowed to identify which requirements alternative corresponds to the most energy consuming implemented solution. We identify that the code associated to the `vital parameters` dependency consumes 4.25% more energy than that mapped to the task `pressing panic button`, thus raising attention to the refinement of this latter GO element.

3. Discussion and Future work

So far we have applied the proposed REST method to the *TAS* case study. This first experiment served to assess the feasibility of the method and helped point out issues to be addressed for consolidating it. They include the automation of the test cases derivation procedure, and the definition of guidelines for performing the method's activities, as depicted Figure 1, in a systematic way.

The application of the method to other benchmark case studies will help collect further evidence about the appropriateness of the method. More specifically, this latter research will contribute to assess the usefulness of the identified model indicators of energy consumption, (e.g. number of actor's inter-dependencies), as well as to identify other possible model indicators. This is a key step in our envisioned approach to increase energy awareness at requirements

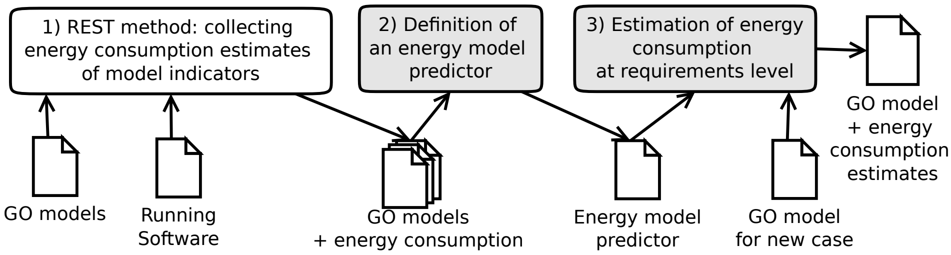

Figure 3: Overview of the envisioned approach for energy awareness at requirements model time

time, according to the overview depicted in Figure 3.

The following step in Figure 3 corresponds to the definition of an energy model predictor based on the identified model indicators of energy consumption. As mentioned, so far we are focusing on the study of two model indicators in GO models that can leads to greater energy consumption in the associated code, namely the number of means-end relationships modelling the utilisation of resources and the number of dependencies between system's actors. We thus studied the relationship, if any, between these model indicators and the energy consumed by software systems. The study could be extended to exploiting model anomalies as sources of energy efficiency improvements. In recent years, code refactoring was studied to improve energy efficiency of source codes, e.g., [18], however bad smells in models are not yet studied for this purpose, so it could be worth to be investigate further along this direction.

Finally, the Step 3 in Figure 3, corresponds to the moment when energy model predictor will be used to approximately deduce the potential energy consumed by new case studies from their GO models . The energy model predictor helps us to identify model indicators that can influence energy consumption and to assess how they can impact energy efficiency at requirements level.

References

[1] R. Verdecchia, P. Lago, C. Ebert, C. de Vries, Green it and green software, IEEE Software 38 (2021) 7–15.

[2] B. Penzenstadler, H. Femmer, A generic model for sustainability with process- and product-specific instances, in: Workshop on Green in/by Software Engineering, GIBSE '13, Association for Computing Machinery, New York, USA, 2013, p. 3–8.

[3] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, M. Mahaux, B. Penzenstadler, G. Rodríguez-Navas, C. Salinesi, N. Seyff, C. C. Venters, C. Calero, S. A. Koçak, S. Betz, The karlskrona manifesto for sustainability design (2014).

[4] L. Duboc, B. Penzenstadler, J. Porras, S. A. Koçak, S. Betz, R. Chitchyan, O. Leifler, N. Seyff, C. C. Venters, Requirements engineering for sustainability: an awareness framework for designing software systems for a better tomorrow, Requir. Eng. 25 (2020) 469–492.

[5] J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, J.-N. Mazon, Integrating sustainability in decision-making processes: A modelling strategy, in: 2009 31st International Conference on Software Engineering - Companion Volume, 2009, pp. 207–210.

[6] S. Gupta, P. Lago, R. Donker, A framework of software architecture principles for

sustainability-driven design and measurement, in: 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C), 2021, pp. 31–37.
[7] C. V. Paradis, R. Kazman, D. A. Tamburri, Architectural tactics for energy efficiency: Review of the literature and research roadmap, in: 54th Hawaii International Conference on System Sciences, HICSS, ScholarSpace, 2021, pp. 1–10.
[8] B. Dornauer, M. Felderer, Energy-saving strategies for mobile web apps and their measurement: Results from a decade of research (preprint), arXiv preprint arXiv:2304.01646 (2023).
[9] S. Chowdhury, S. Borle, S. Romansky, A. Hindle, Greenscaler: training software energy models with automatic test generation, Empirical Software Engineering 24 (2019) 1649–1692.
[10] J. Mylopoulos, L. Chung, S. S. Liao, H. Wang, E. S. K. Yu, Exploring alternatives during requirements analysis, IEEE Softw. 18 (2001) 92–96.
[11] J. Horkoff, T. Li, F. Li, M. Salnitri, E. Cardoso, P. Giorgini, J. Mylopoulos, Using goal models downstream: A systematic roadmap and literature review, Int. J. Inf. Syst. Model. Des. 6 (2015) 1–42.
[12] C. D. Nguyen, A. Perini, P. Tonella, Goal-oriented testing for mass, Int. J. Agent Oriented Softw. Eng. 4 (2010) 79–109.
[13] M. Unterkalmsteiner, R. Feldt, T. Gorschek, A taxonomy for requirements engineering and software test alignment, ACM Transactions on Software Engineering and Methodology (TOSEM) 23 (2014) 1–38.
[14] D. Weyns, R. Calinescu, Tele assistance: A self-adaptive service-based system exemplar, in: P. Inverardi, B. R. Schmerl (Eds.), 10th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015, Florence, Italy, May 18-19, 2015, IEEE Computer Society, 2015, pp. 88–92.
[15] A. Noureddine, Powerjoular and joularjx: Multi-platform software power monitoring tools, in: 18th Intl. Conf. on Intelligent Environments, IE 2022, Biarritz, France, June 20-23, 2022, IEEE, 2022, pp. 1–4.
[16] S. A. Chowdhury, A. Hindle, Greenoracle: estimating software energy consumption with energy measurement corpora, in: M. Kim, R. Robbes, C. Bird (Eds.), Proceedings of the 13th International Conference on Mining Software Repositories, MSR 2016, Austin, TX, USA, May 14-22, 2016, ACM, 2016, pp. 49–60.
[17] F. Dalpiaz, X. Franch, J. Horkoff, istar 2.0 language guide, CoRR abs/1605.07767 (2016). URL: http://arxiv.org/abs/1605.07767. arXiv:1605.07767.
[18] R. Morales, R. Saborido, F. Khomh, F. Chicano, G. Antoniol, EARMO: an energy-aware refactoring approach for mobile apps, IEEE Trans. Software Eng. 44 (2018) 1176–1206.