Exploiting Goal-oriented Requirements Models for Increasing Energy Awareness: a Research Preview

Denisse Muñante*, Anna Perini†, Angelo Susi†
*ENSIIE & SAMOVAR, Évry, France
†Fondazione Bruno Kessler, Trento, Italy
denisse.munantearzapalo@ensiie.fr, {perini, susi}@fbk.eu,

Index Terms—**Goal-oriented modelling, Energy awareness, Energy efficiency, Goal-oriented testing**

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]–[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.

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]; and *(ii)* 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 [12].

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.

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 [13], which is a Java-based software energy monitoring tool. Then, energy consumption data are analysed. In particular, they are used to assess energy

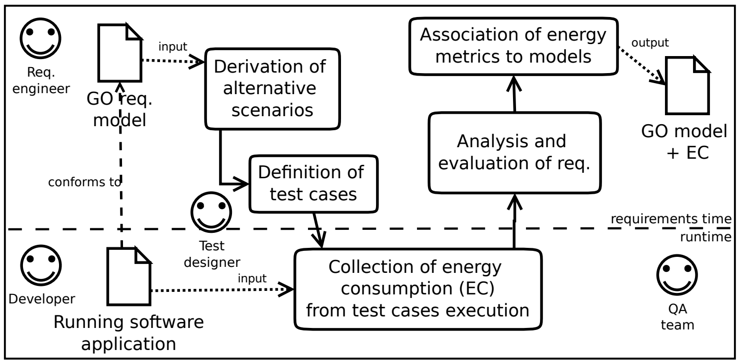
Req. engineer
GO req. model
input
Derivation of alternative scenarios
Association of energy metrics to models
output
GO model + EC
Analysis and evaluation of req.
conforms to
Definition of test cases
Test designer
requirements time
runtime
Developer
Running software application
input
Collection of energy consumption (EC) from test cases execution
QA team

Fig. 1. Process of the proposed REST method

[1]That is methods that connect RE and Software Testing (ST) [12].

consumption related to **alternative requirements** modelled in OR refinements. The collected data is then aggregated and associated back to requirements GO models.

In order to evaluate the feasibility of our proposal, we use the Tele Assistance Service (TAS) system [14].

Design time: Figure 2 depicts an excerpt of the iStar2.0 early requirement model of the *TAS* system. 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 advice and easy access, 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 are 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 elements 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.

Run time: 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 button` and the resource dependency `vital parameters`, for which we executed and tested the *TAS* system, using JoularJX to measure energy consumption. 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. To manage fluctuations of the gathered energy consumption measurements, we run each scenario 120 times. An excerpt of the collected data are given in Table I.

Back to design time: Their analysis 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 approximately 3.25% more energy than that mapped to the task `pressing panic button` (this difference was determined as significant after running the **wilcoxon** test), thus raising attention to the refinement of this latter GO element.

TABLE I
THE DISTRIBUTION OF THE ENERGY CONSUMED (EC) BY THE TAS SYSTEM'S SCENARIOS, COMPUTED ON 120 EXECUTIONS. WHERE: RSDEC = RELATIVE STANDARD DEVIATION OF EC

Scenarios	EC Average	EC Median	RSDEC
Pressing button Message	18.44	18.03	11.62
Vital parameters Message	19.06	18.61	11.88

Discussion and next steps: 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. 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.

Concerning our longer term research, Figure 3 depicts an overview of our envisioned approach to increase energy awareness at requirements time, and shows that the proposed REST method corresponds to a first key step in the whole process. 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 focused on two model indicators in GO models that can lead 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. The exploration for other model indicators will be pursued together with the applicability of concepts of bad smells in models. Finally, the Step 3 in Figure 3, corresponds to the exploitation of the energy model predictor in a new project by approximately deducing energy consumption from the analysis of the corresponding GO models.

ACKNOWLEDGMENTS

This work was partially supported by the "GreenSE4IoT: Towards Energy-efficient Software for Distributed Systems" project whose code is STIC-AMSUD 22-STIC-04, and was partially carried out at the Energy4Climate Interdisciplinary Center (E4C) of IP Paris and École des Ponts ParisTech, which is in part supported by 3rd Programme d'Investissements d'Avenir [ANR-18-EUR-0006-02].

REFERENCES

[1] R. Verdecchia, P. Lago, C. Ebert, and C. de Vries, "Green it and green software," *IEEE Software*, vol. 38, no. 6, pp. 7–15, 2021.

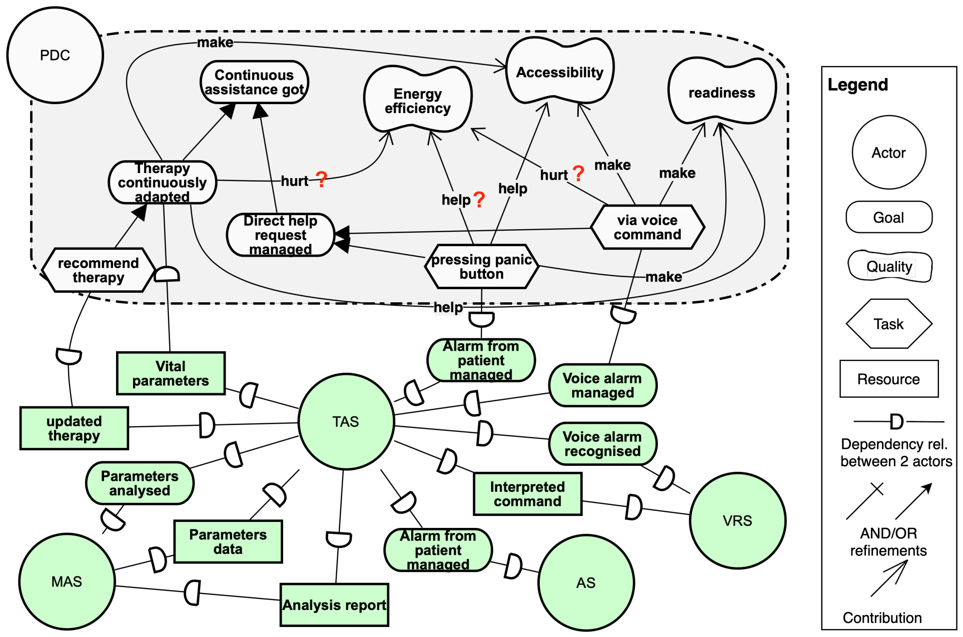

Fig. 2. Early requirements GO model of the Tele Assistance Service. iStar 2.0. notation [15].

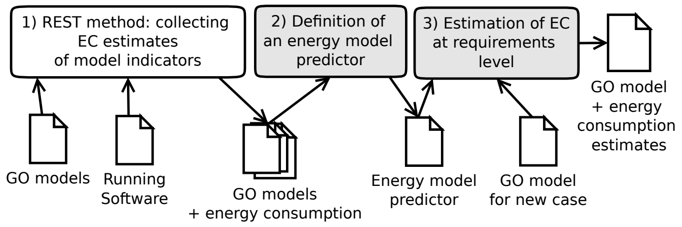
1) REST method: collecting EC estimates of model indicators
2) Definition of an energy model predictor
3) Estimation of EC at requirements level
GO model + energy consumption estimates
GO models
Running Software
GO models + energy consumption
Energy model predictor
GO model for new case

Fig. 3. Overview of the envisioned approach for energy awareness at requirements model time

[2] B. Penzenstadler and H. Femmer, "A generic model for sustainability with process- and product-specific instances," in *Workshop on Green in/by Software Engineering*, ser. GIBSE '13. New York, USA: Association for Computing Machinery, 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, and 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, and C. C. Venters, "Requirements engineering for sustainability: an awareness framework for designing software systems for a better tomorrow," *Requir. Eng.*, vol. 25, no. 4, pp. 469–492, 2020.

[5] J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, and 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, and 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, and 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 and M. Felderer, "Energy-saving strategies for mobile web apps and their measurement: Results from a decade of research (preprint)," *preprint arXiv:2304.01646*, 2023.

[9] S. Chowdhury, S. Borle, S. Romansky, and A. Hindle, "Greenscaler: training software energy models with automatic test generation," *Empirical Software Engineering*, vol. 24, no. 4, pp. 1649–1692, 2019.

[10] J. Mylopoulos, L. Chung, S. S. Liao, H. Wang, and E. S. K. Yu, "Exploring alternatives during requirements analysis," *IEEE Softw.*, vol. 18, no. 1, pp. 92–96, 2001.

[11] J. Horkoff, T. Li, F. Li, M. Salnitri, E. Cardoso, P. Giorgini, and J. Mylopoulos, "Using goal models downstream: A systematic roadmap and literature review," *Int. J. Inf. Syst. Model. Des.*, vol. 6, no. 2, pp. 1–42, 2015.

[12] M. Unterkalmsteiner, R. Feldt, and T. Gorschek, "A taxonomy for requirements engineering and software test alignment," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 23, no. 2, pp. 1–38, 2014.

[13] 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.

[14] D. Weyns and R. Calinescu, "Tele assistance: A self-adaptive service-based system exemplar," in *10th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015, Florence, Italy, May 18-19, 2015*, P. Inverardi and B. R. Schmerl, Eds. IEEE Computer Society, 2015, pp. 88–92.

[15] F. Dalpiaz, X. Franch, and J. Horkoff, "istar 2.0 language guide," *CoRR*, vol. abs/1605.07767, 2016. [Online]. Available: http://arxiv.org/abs/1605.07767

[16] S. A. Chowdhury and A. Hindle, "Greenoracle: estimating software energy consumption with energy measurement corpora," in *Proceedings of the 13th International Conference on Mining Software Repositories, MSR 2016, Austin, TX, USA, May 14-22, 2016*, M. Kim, R. Robbes, and C. Bird, Eds. ACM, 2016, pp. 49–60.