Green Software Development and Research with the HADAS toolkit

Daniel-Jesus Munoz
Universidad de Málaga
Andalucía Tech, Málaga, Spain
danimg@lcc.uma.es

Mónica Pinto
Universidad de Málaga
Andalucía Tech, Málaga, Spain
pinto@lcc.uma.es

Lidia Fuentes
Universidad de Málaga
Andalucía Tech, Málaga, Spain
lff@lcc.uma.es

ABSTRACT

Energy is a critical resource, and designing a sustainable software architecture is a non-trivial task. Developers require energy metrics that support sustainable software architectures reflecting quality attributes such as security, reliability, performance, etc., identifying what are the concerns that impact more in the energy consumption. A variability model of different designs and implementations of an energy model should exist for this task, as well as a service that stores and compares the experimentation results of energy and time consumption of each concern, finding out what is the most eco-efficient solution. The experimental measurements are performed by energy experts and researchers that share the energy model and metrics in a collaborative repository. HADAS confronts these tasks modelling and reasoning with the variability of energy consuming concerns for different energy contexts, connecting HADAS variability model with its energy efficiency collaborative repository, establishing a Software Product Line (SPL) service. Our main goal is to help developers to perform sustainability analyses finding out the eco-friendliest architecture configurations. A HADAS toolkit prototype is implemented based on a Clafer model and Choco solver, and it has been tested with several case studies.

CCS CONCEPTS

• **Software and its engineering → Abstraction, modeling and modularity**; **Requirements analysis**; **Software design tradeoffs**; *Development frameworks and environments*; *Software libraries and repositories*; *Collaboration in software development*; Software performance; • **Hardware → Power estimation and optimization**;

KEYWORDS

Clafer, CVL, Software Product Line, Variability, Energy Efficiency, Metrics, Repository, Optimisation

ACM Reference format:
Daniel-Jesus Munoz, Mónica Pinto, and Lidia Fuentes. 2017. Green Software Development and Research
with the HADAS toolkit. In *Proceedings of ECSA '17, Canterbury, United Kingdom, September 11–15, 2017,* 7 pages.
DOI: 10.1145/3129790.3129818

1 INTRODUCTION

Running software do not directly consume energy, but for instance, Intel estimates that energy-efficient software can realise savings of a factor of three to five beyond what can be achieved through energy-efficient hardware [14]. As a result, there has recently started a new trend named *Green Computing* aiming to promote *sustainable software architectures* by making a responsible usage of hardware resources, focusing especially on requirements engineering and methodological analysis, as well as on collaborative aspects. Having SPL as a powerful approach to model the commonality and variability of software systems, is possible to reason about the quality of different variants of a given domain with variability models, generating custom made configurations. Hence, to characterise interesting configurations, software architects and analysts need to identify and model common *Energy Consuming Concerns* (ECC), their design and implementation variants, and existing dependencies. The main goal is to provide developers with the necessary means to analyse the power consumption of alternative solutions to develop an application. Thus, software developers could be conscious of the impact of the software in the hardware energy consumption (e.g., CPU, GPU, Memory, etc.).

We must include, alongside our variability model, energy metrics, so the service can reason about the energy consumption in time of different ECCs. Several empirical papers [2, 3, 5, 7, 13] show what are the most energy efficient implementations for some concerns, so part of this work is already done. The published results of these studies are not complete and are difficult to understand, hindering the reuse by third party developers. Also, these papers do not formally model the factors that influence in the power consumption of their respective ECCs, so there is no evidence that all possible eco-friendly configuration were considered in the study. Then, one of the challenges to address is *give the possibility to software developers to automatically find and reason about energy consumption researches of different designs and implementations of each ECC.*

A plethora of experimental studies [4, 6, 7, 10–12, 15] have been published since last decade, trying to identify the parts of a software that influence the most in the total energy consumption of a software system. Nonetheless, these studies only focus on one specific concern (e.g., Java collections [7]) evaluating the power consumption of a limited set of design and/or implementation alternatives (e.g., Java Trove Collections Framework). So, another of the challenges to address is to *identify and model the ECCs, the list of large energy consumers and that are recurrent, with at least one of them being present on any software.* Examples of ECCs are security, compression, storage, in conjunction with its respective design alternatives (e.g., using the AES encryption algorithm, MP3 compression algorithm, dynamic memory) and implementations (e.g., Apache Commons Crypto, Franhofer MP3 Encoder, Java Trove Collections Framework).

Coping with these challenges we propose to define HADAS[1], a toolkit that aims to promote Green Computing among software developers community. Concretely:

(1) **HADAS will work with a model that mirror the high level variability of existing ECCs, regarding design, implementation and energy context.** This is the core of HADAS, which we will exploit for generating: (1) the information provided in the default User Interface (UI), so concrete configurations can be select for its their sustainability analysis; (2) a database structure and a system of queries to store and obtain concrete metrics; and (3) a plotting system, providing graphs of energy and time consumption. This is not a trivial task; variability models should be extended with pre-defined features, covering energy-related data defining the configurations, metrics and meta-data, alongside optimisation, plotting and analysis settings, not previously found all-together in an SPL tool or service.
(2) **HADAS must work with a repository that can store several energy and time consumption of all the possible complete configurations present in the model** We propose a collaborative approach, where the goal is to provide researchers a shared place to register their empirical results among software developers.
(3) **HADAS will provide the necessary data for performing richer analysis in comparison to the ones found in articles and energy experiments.** Normally, researchers do not publish the whole bunch oisf metrics, nor they analyse every possible configuration, even though they have the proper data. If more than one research team has added the data to our repository (collaborative approach), the analysis is even extended further, as more than one source can be used for reasoning, so the community would be able to perform sustainability analysis with graphs, advices, and the metrics itself, automatically from every registered scenario.

The paper is organized as follows. Section 2 clearly states the motivation with the development of HADAS Toolkit. Section 3 details the approach with a general vision of both, the assistant and the repository. In Section 4, real use cases are evaluated from both perspectives, researcher and developer, with HADAS prototype, highlighting the improvements in the sustainability analysis, as well as its scalability. We end in Section 5 outlining current conclusions and future work.

2 MOTIVATION

HADAS must cope with several questions, covering the needs of: energy researchers wishing to expose their energy models and measurements, so the community can access and reason about; and developers reusing that knowledge in their applications development process.

RQ1: How to model the complete variability of ECCs alongside the energy information reasoning about efficiency: By now, we reduce the ECCs to six: Security, Compression, Distribution, Storage in Memory, Cache and Communication. The number of designs and technologies of each ECC is immense, so a variability model is an appropriate option to support reasoning about eco-efficient configurations. We propose to model the *Energy Context* and the *Design Technologies* as distinct features of the variability model. As part of the Energy Context, we have three types of inputs determining the exact energy model of every possible complete configuration. They are: (1) Parameters (e.g., number of users, data size, load); (2) DataType (e.g., PHP file, object, Integer); and (3) Operation (e.g., HTTP Request, Iterate, Insert). Also, we must have a repository that stores the energy and time consumption measurement of each complete configuration, considering the parameters that affects these configurations (e.g., file size affects the communication framework consumption at transferring data). We propose a relational database, that defines tables mirroring the structure and node names of the variability model, all-together with the metrics and the meta-data (i.e., experimental context: devices, CPU type and frequency, etc.). We should be able to, as well, specify cross-tree constraints to, for example, restrict the use of a framework for a given programming language, or operating system (e.g., Nginx web server implies having FastCGI PHP mode, Sign in implies Storage in Memory). To recap, we propose a four-level with three sub-levels variability tree structure: (1) ECCs, (2) Designs, (3) Technologies, (4) Energy Context (4.1) Parameters, (4.2) Data Type, (4.3) Operation.

RQ2: How to insert empirical energy-related measurements into the repository: We base in a collaborative and incremental approach, where the repository is ready to include the design and technology variability information, alongside the energy metrics provided by professionals, researchers or even extracted from published articles. An energy efficiency repository in the cloud is an extensive task, and also, we cannot take the responsibility of measuring and storing the energy of every possible configurations of each ECC, and in short periods of time. So far, in initial versions of HADAS, the evolution of the variability model will be performed by HADAS developers, because it is not a trivial task to automatically classify features of an energy model into our four levels structure. Regarding the metrics, we propose a file with CSV extension that shares the structure of the variability model, together with the development of an automatic SQL query generator that parses the file, inserting and linking the metrics with its respective complete configurations.

RQ3: How to reason over the variability, allowing eco-efficiency analysis: We chose Clafer, as its variability model can be transformed into a constraint satisfaction problem [1] to automatically generate valid configurations reasoning about them (throughout Choco solver), fulfilling model constraints. To automatically access the proper data of the repository, we must develop another SQL query generator to select the metrics parsing the complete configurations provided by the execution of Clafer with Choco. About the analysis, on the one hand, developers can partially select/deselect ECCs (e.g., storage in memory), the alternatives that implement them (e.g., Java collections implementations of different frameworks) and their energy context values (e.g., insert operation).

[1]HADAS project, funded by the Spanish Government https://www.researchgate.net/project/HADAS-A-toolkit-for-analysis-and-development-of-sustainable-software

HADAS must have different interfaces to perform this process, being one a default user-friendly web UI, with a tree view *form*, and being another as web micro-services, where the selection process and retrieval of data is made through a proper URL (e.g., an Eclipse plugin that shows the energy consumption of a function, straight in the IDE while coding, and a comparison with its alternatives). Normally, developers are not totally restrict to an specific energy context (e.g., which web application technology is the greenest one depending on the data type, the most frequent operation, etc.). HADAS generates, in its default web UI, graphical information of energy data supporting an energy analysis by graphs comparison. Regarding micro-services, even if it is possible for HADAS to respond with graphs, it is common to just request sorted metrics, delegating the representation of the data to the plugin itself.

Fourth, ***RQ4: How to represent in the model any type of energy profile:*** Clafer variability models supports numeric variable features, and even to constraint them into numerical intervals. But we do not consider numerical changes as different configurations since: (1) Clafer models does not support a list of values definition or constraint, numeric or not; (2) Clafer models does not implement enumerate types, just Integers; and (3) The repository runs a single query obtaining all the metrics for the different parameters in the interval at once, instead of executing the entire process per value, improving performance and scalability. Furthermore, in case of graphs generation, a base parameter must be chosen (the x-axis variable), so we extended Clafer model with this functionality. By now, we treat tree levels and the extra features just discussed (e.g., Energy Context) by locating certain 'key strings' in our Clafer model, acting as tags defining the level (e.g., Design) of the sub-tree.

3 HADAS APPROACH

The overview of HADAS is represented in Figure 1 with two clearly distinguishable roles, *Developer* and *Researcher*, and two types of interfaces to access the repository, default web UI and micro-services.

Researchers (left side of Figure 1) have produced experimental data and want to add it to our repository. According to Figure 1, they have to represent the energy model and metrics in an understandable format (currently *Input Data Format* in CSV), so HADAS developers can update the *Variability Model*, if necessary, and the, link and insert automatically the metrics into the *Energy Efficiency Repository* with a script.

The format consists in a nine columns and several rows table (Figure 2). Each column represents a level of the variability tree 3, the Energy Context sub-levels (Parameters, Data Type and Operation), the metrics (Joules and Seconds) and the meta-data. Character '*' in the first column indicates that the row belongs to the same configuration (is caused by multiple technologies or parameters). For example, Figure 2 reflects the research of a different technology alternatives (*FastCGI/Apache*, *Mod_PHP/Apache*,...) of a design variant *Server* of the ECC *Distribution*, alongside *Joules* and *Seconds* consumption, for a different concurrent number of *Users* and PHP computation loads (in this case *Heavy*) in a HTTP *Request* operation.

Developers (right side of Figure 1) have the need to detect energy hotspots, performing an eco-efficient analysis of an specific use case. As shown in Figure 1, he access *HADAS Web User Interface* and navigate throughout HADAS form, selecting the ECC(s) variants that he wants to analyse. The form allows to select, apart of a partial configuration, the x-axis plot variable (needed to sort the metrics and generate the desired graphs). Once finished selection, HADAS set a *Partial/Customized Configuration* in the Clafer mode, and ends up generating automatically the *Sustainability Charts*.

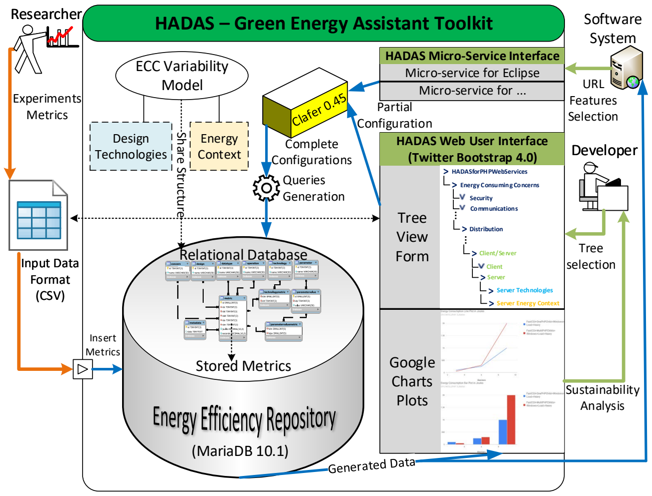
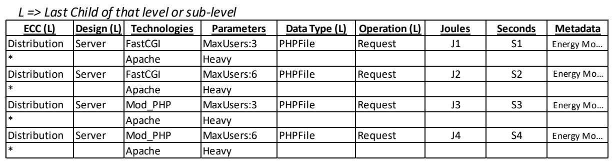


Figure 1: HADAS eco-assistant toolkit overview



L => Last Child of that level or sub-level

ECC (L)	Design (L)	Technologies	Parameters	Data Type (L)	Operation (L)	Joules	Seconds	Metadata
Distribution	Server	FastCGI	MaxUsers:3	PHPFile	Request	J1	S1	Energy Mo...
*		Apache	Heavy					
Distribution	Server	FastCGI	MaxUsers:6	PHPFile	Request	J2	S2	Energy Mo...
*		Apache	Heavy					
Distribution	Server	Mod_PHP	MaxUsers:3	PHPFile	Request	J3	S3	Energy Mo...
*		Apache	Heavy					
Distribution	Server	Mod_PHP	MaxUsers:6	PHPFile	Request	J4	S4	Energy Mo...
*		Apache	Heavy					

Figure 2: HADAS CSV Input Data Format

3.1 Variability Tree

HADAS variability tree has four levels (left side of Figure 3):

(1) **ECCs**: The first level is devoted to specifying common ECCs, which are six by now: Security, Compression, Distribution, Storage in Memory, Cache and Communication.
(2) **Design variants**: Each ECC has a sub-tree with its design options. In variability model terminology, they are optional features, with only one of them able for selection in complete configurations (1..n legend in Figure 3). Subject to analysed variability, can be one or more sub-levels. Each leaf of the design sub-tree (e.g., Design 2.1 in Figure 3) has two sub-trees defined (*Technologies* and *Energy Context*), which are mandatory features in the variability model (always present at every configuration).
(3) **Technology alternatives**: It defines different implementations for each design. We add sub-tree of different levels, modelling the variability of frameworks, APIs, etc. Hence, HADAS provides to developers the energy consumption of a concrete design, implemented with different technologies. Because the semantics of the cardinality in the form

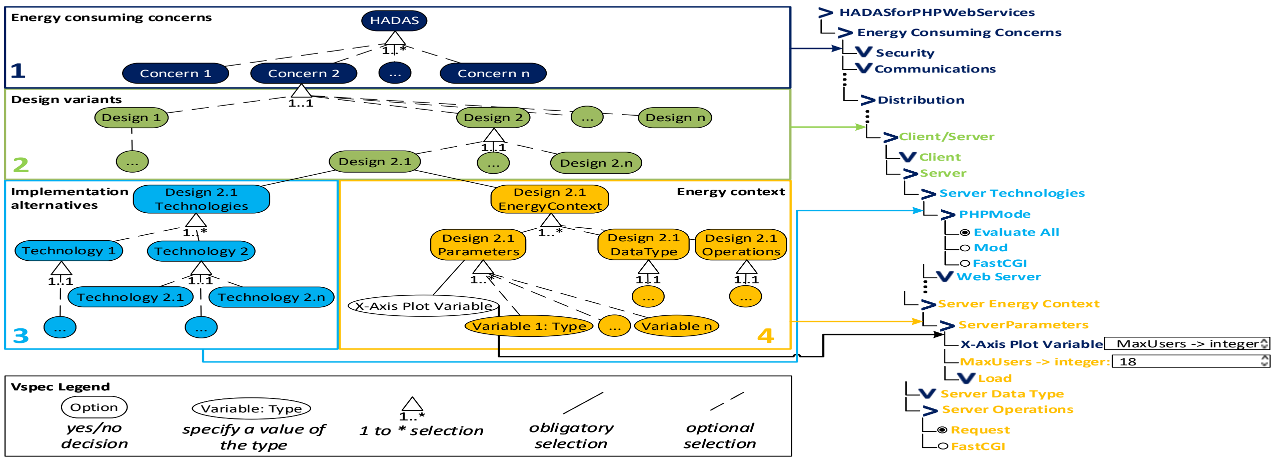


Figure 3: HADAS variability tree structure by levels and the UI form

differ from the ones defined by a variability model, in the prototype the user can select just one leaf of a concrete subtree level or, instead, check 'Evaluate All' (see right side of Figure 3). This option mirrors de functionality of selecting the parent, so the whole sub-tree variability is analysed further in the solver, being more intuitive for users.

(4) **Energy Context**: The energy context contains three children:
   (a) **Parameters**: All parameters must have a value, so for Integers, we set by default the maximum size available in the repository (indicated in the variability model with a constraint like `[MaxUsers <= 18]`). The user can decrease this value in the form. There are text parameters (PHP benchmark computation load: Light, Medium or Heavy), with the last option selected by default. Additionally, the user must select which variable will be the x-axis of the graphs, where the first numeric one of the variability model is selected by default.
   (b) **Data types**: Software systems work with different types of data (e.g., plain text, XML, Object, Integer, Varchar(50)), so one option must be selected, having the last option selected by default.
   (c) **Operations**: Each design covers a series of operations that are involved in the application (e.g., for data collections: Insert, Search, etc.). In the prototype, one option must be selected, having the last option selected by default.

   Occasionally, a technology does not implement an operation, and/or an operation does not suit a data type. This is formally specified in the variability model with cross-tree constraints like `[Technology a => not DataType b]`.

As specified, HADAS works with a Clafer [1] model, formatted as text file (.txt). With it, variability models can be easily transformed into a CSP to automatically generate and reason about valid configurations with Choco solver, fulfilling model constraints and providing every complete configuration that fits with the declared constraints in the model.

3.2 Default Web UI

The UI form shows the details of the variability model in a friendly unfolding tree-view that facilitates user navigation and selection, as in the right side of Figure 3. Once Clafer model is defined, we automatically parse the text file and generate a dynamic web form, allowing developers to select different scenarios. This process has to fit with the cross-tree constraints, following the same colour legend as Figure 3 scheme and guiding developers to comprehend at which level that option belongs to. The form considers constraints dynamically; the branches subject to constraints will be marked and displayed automatically if activated. Summarising, we have defined a mapping between the formal variability model in Clafer and the web form provided to the user.

3.3 Energy Efficiency Repository

We have showed in Figure 1 that the metrics alongside their configuration structure (design and implementation features as in the variability model) are stored in a MariaDB relational database, forming what we have called the energy efficiency repository. As we can see in Figure 4, the structure of the database mirror the levels and features of the variability model, where each level and sub-level has its referent in a specific table (tables Concern, Design, Technology, DataType, Operation, Parameter).

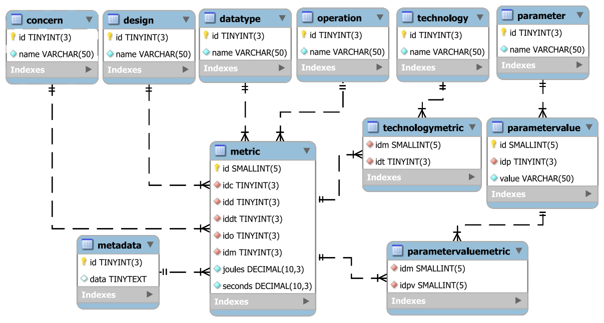


Figure 4: Energy Efficiency Repository Database Diagram

Because a metric can depend on several Technologies, parameters, and values of those parameters, for consistency and avoiding duplicity, we have modelled many-to-many intermediate tables. To allow the users and micro-services to perform richer analyses, HADAS allows the inclusion of energy data collected with the same configuration from different sources. This is achieved by including the meta-data table, that contains additional information of the energy model like the measurement system and the hardware used to obtain the data. In addition, the Metric table relates these tables, plus the meta-data table, with energy consumption (millijoules) and execution time (milliseconds). After Clafer model with Choco Solver have generated the configurations, the results are parsed with our PHP Script, creating the proper SQL queries to retrieve the metrics from the repository, sending them to the sustainability charts process, or to the micro-service output.

3.4 Sustainability Charts

To proper represent the data from the repository in the default UI, after a parse, HADAS creates a web view plotting that data (with the x-axis variable indicated) in a proper format making use of the Google Charts API. In a preliminary version two types of graphs will be generated, lines and bars, with two graphs of each type (Joules, and Seconds, per x-axis variable value) giving a total of four graphs (relevant consumption and execution runtime on lines and bar graphs). The developer, then, have the necessary information to perform a use case sustainability analysis in an intuitive way.

3.5 Variation Points

As we continue with this research: (1) there is still an unsettled decision regarding the variability model; where to locate programming languages and operating systems in a quite evolve variability tree and to which level they belongs (by now we propose Technology level, but it represents high repetition of features); (2) the variability model can get really complex (as it evolves), and the developers can get disoriented in the selection process; (3) if a metric depends on a lot of technologies and parameters at the same time (a complex configuration in a variability model), the database will slow down considerably, affecting the response time of HADAS interface and micro-services (4) the sustainability analysis depends on the data stored, which, at the same time, depends on the energy models that has been used during the measurement or simulation (machine, configuration, measurement system and tools), and the accuracy of the metrics; we try to solve this with the meta-data information, leaving to the developer the decision of the level of accuracy of the metrics; (5) even if we provide software efficiency insights through comparisons, they metrics depends at some degree on the hardware in which the use case run, and the tools used for measurement or simulation; we assume that these differences do not change the sustainability analysis.

4 EVALUATION

We started to evaluate HADAS toolkit in two ways: (i) discussing that the tool gives more configuration insights to developers than the conclusions showed in published papers, and (ii) checking the scalability of the HADAS approach considering different complexity degrees of the variability tree. Up to now, four different use cases have been considered.

4.1 Case Studies

Our first case was to incorporate the energy consumption information of a PHP web server into HADAS repository. Taking the researcher role, the first step was to evolve our variability tree by adding new features to the *Distribution* ECC and its design variant *Client/Server->Server*. We needed, as well, to add the corresponding trending implementation alternatives and energy context variability as in Figure 5. As it was made in our own HADAS server by our team, we provided the data in the desired format, including the metrics in Joules and Seconds. We really speeded up the process from measuring the data, to realising a broad and rich energy sustainability analysis. Summarising, for better performance and not so many concurrent users, Nginx with several FastCGI processes is the best option. Instead, for lowest energy consumption we rather choose Apache with mod_php, leaving IIS for a balanced solution. Even if it is important to explain the technical side of the data to understand the results that have been obtained through the experimentation, using HADAS the developer can get the same conclusions without any technical explanation, just focusing on applying the knowledge previously stored in HADAS by the researchers, even if he cannot clearly explain or understand it.

Our second case was the study of Java collections [7], where measurements are provided in several CSV files, and different implementations of different frameworks are compared. The research side of the CSV parsing was quite complex, because they follow the GreenMiner format, so we needed to process manually several folders and files, then make a median calculation of the consumption and running time (as they didn't provide a final value, but 32 different runs). We, as well, modelled the variability, checking its validity with Clafer. The number of theoretical configurations for this use case, considering all the variables considered in the experiments presented in the paper is 11.904. This high level of variability proves that is not viable to do a manual energy efficiency analysis. Even if the results are published in a paper, with HADAS a developer can make different analysis, automatically generating graphs that provide additional conclusions to the ones published.

Our third case was the study of cryptographic primitives in Android where the the information is available at http://www.lcc.uma.es/~monte/CryptoConsumption. We followed a similar process as for the second case, and after incorporating into HADAS all the energy measured taken by the authors of this work, we realised that there was much more interesting information than the one commented on the paper. For instance, using HADAS we were able to analyse the energy consumption of the key generation operation between different primitives, which is not mentioned in the paper. Spongy Castle is the provider that consumes less, independently of the primitive. As well, the Cipher primitive is the most energy efficient for the key generation operation. Also, that SHA-1 is the most energy efficient Hash. Usually, papers analyse the big picture, leaving details of much more specific use cases aside of the conclusions.

Our fourth case, still under development, is to stablish a micro-service for Android Studio. Specifically, we are enhancing the Android Security API so that software developers can receive contextual help of energy consumption previously calculated and stored in the repository (previous use case). This process is transparent

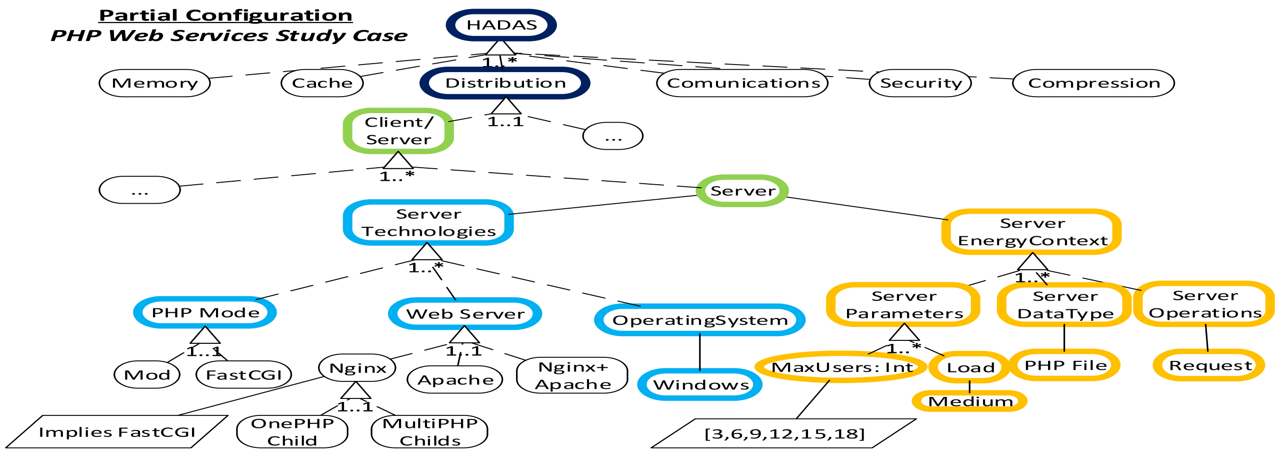


Figure 5: PHP web services variability model

Table 1: The HADAS toolkit scalability

Nodes Dependency	Configurations	Milliseconds
5	**10**	**80**
5	50	410
6	**10**	**100**
6	100	1000
7	**10**	**125**
7	1000	2300

to software developers who can continue to program their applications as usual, just activating our Green Security plugin, so the information is provided to them as they are writing the code in Android Studio. The main idea is to build a plugin that can provide not just 'recommendations', but precise information for concrete devices in which the applications will run. Performance information of cryptographic primitives and the trade-off between energy and runtime will also be incorporated in the plugin.

4.2 Scalability

We have tested the scalability of our solution in the two scenarios (default UI and Micro-service) considering the two processes that are more subject to overload. They have been carried out on an Intel (R) Core i7-4790 CPU @ 3.60 GHz processor with 16 GB of memory RAM and an SSD disk under Microsoft Windows 10 x64 in high performance mode. The parts of HADAS with a high computation load are Clafer (for the generation of configurations) and the access to the repository (for the generation of SQL queries and the retrieval of data from the database).

So, we have focused our tests on testing complex configurations (more nodes per configuration that creates longer SQL queries) and large configurations (more SQL queries). As clarification of *Nodes Dependency*, for the use case of cryptographic primitives in Android, we have a baseline of 5 nodes: *Provider, Algorithm, KeySize value, Byte and Operation*. The number increases to 6 in the case of the *Sign* primitive, where we need to add a *Hash* operation, and to 7 in the case of the *Cipher* primitive where *Padding and Mode* are added. Table 1 covers the average whole spectrum of the analysis that our available use cases can realise. Regarding the number of *Configurations* to compare, we have set 10 as the proper average number of options that a developer would like to analyse, but we have also tested quite high number of options that could be necessary if a micro-service wants to obtain just the most efficient single configuration for an ECC (without declaring constraints). For average cases (10 *Configurations*), the response times are extremely low (around a tenth of a second), independently of the *Nodes Dependency*. We can see how a heavy case of dependency, 7 generating 1000 different *Configurations*, takes a bit more of 2 seconds. However, this is quite unlikely to be requested, once again, unless we want to calculate the single most efficient configuration for every possible combination of an ECC. In general, as noted in Table 1, the *Nodes Dependency* is the critical variable in performance, rather than the number of queries, so as the data depends on more technologies and parameters, higher times can be observed. We tried to check HADAS service limits with a *Nodes Dependency* of 12. Firstly, Clafer was tested with 100.000 complete configurations, obtaining a processing time near 3 minutes. Secondly, the database was tested with 50 complete configurations (mapped into the same number of queries), resulting in HADAS extracting metrics in less than 3 minutes. They are quite unlikely thresholds to occur in real use case configurations, but even ought, it is currently being improved. Regarding micro-services data bandwidth, the input and output strings are small for a web micro-service, so it does not affect the performance of the whole plugin system, even for slow connections. Summarising, HADAS scales pretty well, and suffices for our case study with low response times.

5 RELATED WORK

On the one hand, there are several researches about sustainability analyses. In [2] it is compared the energy consumption of different web technology services (e.g., Apache, PostgreSQL and PHP), based mainly on measuring the time in which the web service components are in use. It is verified that, of all network protocols, HTTP/2 is the most client-side energy efficient, given the requests reduction to the web server, regardless of the client. Nonetheless, it is as well interesting to evaluate the server-side protocol as done in [13]. In that paper are compared several protocols and web servers, concluding that HTTP/2 consume the less, moreover if used in conjunction with the H2O web server. In the case study of Java collections [7], measurements are made, and different implementations with different frameworks are compared, and it is concluded that the ideal configuration depends on the context (e.g., type and size of the data, and the operation). Although, all these works that compare different implementations conclude interesting information, that knowledge is not stored, so it is difficult to be accessed by developers.

On the other hand, there are articles that highlight the importance of having a repository with software energy metrics such as Selflab [8], which contains as future work the deployment of a repository offering several eco-efficient configurations to the developer. One of the most used is the one offered by GreenMiner [9], which is a complete solution as it provides power tools, as well as a repository that stores the experimental data, where big data techniques can be applied. The paper discusses the adversity of energy data processing since there is a lack of normalisation of extracted data, and a base format is not defined, in which we could fit the results of all the experiments. In addition, this repository is local (deployed individually) to the measurement system. In HADAS, instead, we propose a collaborative approach, where the results of the experiments done by different researchers are dumped. A similar approach is followed in [6], highlighting the usefulness of an imaginary cloud repository that helps identifying software design, architecture, and components that are responsible of energy leakage. The authors comment the complication of performing energy data analysis of a specific use case, for which, they define an Energy modeler that allows to restrict the repository search results. In [11] is proposed an Internet of Things (IoT) components repository instead of a generic one like HADAS propose. This article uses one repository to search for components that consume less, providing a *required and provided* interface as specified into the architecture of an IoT service. This approach does not include measurements of specific technologies as proposed by HADAS, and it reasons only at the architectural level, so its actual usefulness is quite limited. HADAS bridge the gap between researchers that produce energy-related data and developers that wants to reuse the knowledge behind experimental data.

6 CONCLUSIONS AND FUTURE WORK

In this article, we have introduced HADAS, the energy aware software eco-assistant. This tool exploits the advantages of variability models to reason, and do a selection between various designs and implementation alternatives considering energy consumption. From the researchers' point of view, HADAS allows their data to be accessible through a collaborative repository. For developers, it allows them to have better understanding of all the variants that exists and to perform a richer software sustainability analysis. As well, we evaluated HADAS in four different cases (PHP web server, Java Collections, Android Security and an Android Studio micro-service) Having tested the scalability, we found that is suffices more than enough our purposes, with response times at the level of seconds. Finally, a prototype is made available to researchers and application developers so they can test their advantages.

As a future work, besides improving the usability and interfaces functionality, we intend to work in the following two ways: (1) developing more micro-services for different systems (2) having richer visualisation results and offer advices based on Multi-objective optimisation.

ACKNOWLEDGMENTS

This work is supported by the projects Magic P12-TIC1814 and HADAS TIN2015-64841-R (co-financed by FEDER funds).

REFERENCES

[1] K. Antkiewicz Michal, A. Murashkin, R. Olaechea, J. H. J. Liang, K. Czarnecki, M. Antkiewicz, K. Bąk, A. Murashkin, R. Olaechea, J. H. J. Liang, and K. Czarnecki. 2013. Clafer tools for product line engineering. *Software Product Line Conference (SPLC)* (2013), 130.

[2] L. Bertini, J. C. B. Leite, and D. Mossé. 2007. Statistical QoS guarantee and energy-efficiency in web server clusters. *Proceedings - Euromicro Conference on Real-Time Systems* (2007), 83–92. DOI:http://dx.doi.org/10.1109/ECRTS.2007.31

[3] R. Bianchini and R. Rajamony. 2004. Power and Energy Management for Server Systems. *IEEE 0018-9162* 04 (2004), 1–11.

[4] A. Brunnert, C. Vögele, and H. Krcmar. 2013. Automatic performance model generation for java enterprise edition (EE) applications. In *EPEW*, Vol. 8168 LNCS. 74–88. DOI:http://dx.doi.org/10.1007/978-3-642-40725-3-7

[5] S. A. Chowdhury, V. Sapra, and A. Hindle. 2016. Client-side Energy Efficiency of HTTP / 2 for Web and Mobile App Developers. *Proceedings, 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering* (2016), 529–540. DOI:http://dx.doi.org/10.1109/SANER.2016.77

[6] K. Djemame, D. Armstrong, R. Kavanagh, A. J. Ferrer, D. G. Perez, D. Antona, J. C. Deprez, C. Ponsard, D. Ortiz, M. Macias, J. Guitart, F. Lordan, J. Ejarque, R. Sirvent, R. Badia, M. Kammer, O. Kao, E. Agiatzidou, A. Dimakis, C. Courcoubetis, and L. Blasi. 2014. Energy efficiency embedded service lifecycle: Towards an energy efficient cloud computing architecture. In *CEUR Workshop Proceedings*, Vol. 1203. 1–6.

[7] A. Fallis. 2013. Energy Profiles of Java Collections Classes - PHD. *Journal of Chemical Information and Modeling* 53, 9 (2013), 1689–1699. DOI:http://dx.doi.org/10.1017/CBO9781107415324.004 arXiv:arXiv:1011.1669v3

[8] M. A. Ferreira, E. Hoekstra, B. Merkus, B. Visser, and J. Visser. 2013. Seflab: A lab for measuring software energy footprints. In *2013 2nd International Workshop on Green and Sustainable Software, GREENS 2013 - Proceedings*. 30–37. DOI:http://dx.doi.org/10.1109/GREENS.2013.6606419

[9] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky. 2014. GreenMiner: a hardware based mining software repositories software energy consumption framework. In *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*. 12–21. DOI:http://dx.doi.org/10.1145/2597073.2597097

[10] G. Kalic, I. Bojic, and M. Kusek. 2012. Energy consumption in android phones when using wireless communication technologies. In *MIPRO, 2012 Proceedings of the 35th ....* 754–759. http://ieeexplore.ieee.org/xpl/login.jsp?tp=

[11] D. Kim, J.-Y. Choi, and J.-E. Hong. 2017. Evaluating energy efficiency of Internet of Things software architecture based on reusable software components. *International Journal of Distributed Sensor Networks* 13, 1 (2017), 155014771668273. DOI:http://dx.doi.org/10.1177/1550147716682738

[12] I. Manotas, L. Pollock, and J. Clause. 2014. SEEDS: a software engineer's energy-optimization decision support framework. In *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. ACM Press, New York, New York, USA, 503–514. DOI:http://dx.doi.org/10.1145/2568225.2568297

[13] V. Sapra and A. Hindle. 2016. Web Servers Energy Efficiency Under HTTP/2. *PeerJ Preprints* May (2016), 1–18. DOI:http://dx.doi.org/10.7287/peerj.preprints.2027v1

[14] B. Steigerwald, R. Chabukswar, K. Krishnan, and J. Vega. 2008. Creating energy-efficient software. *IWP* (2008).

[15] C. Stier, A. Koziolek, H. Groenda, and R. Reussner. 2015. Model-Based Energy Efficiency Analysis of Software Architectures. In *9th European Conference on Software Architecture, ECSA 2015*, Vol. 9278. 221–238. DOI:http://dx.doi.org/10.1007/978-3-319-23727-5