

The SDK4ED Platform for Embedded Software Quality Improvement - Preliminary Overview

Miltiadis Siavvas[1(✉)], Dimitrios Tsoukalas[1,3], Charalampos Marantos[2], Angeliki-Agathi Tsintzira[3], Marija Jankovic[1], Dimitrios Soudris[2], Alexander Chatzigeorgiou[3], and Dionysios Kehagias[1]

[1] Centre for Research and Technology Hellas, Thessaloniki, Greece
{siavvasm,tsoukj,jankovic,diok}@iti.gr
[2] National Technical University of Athens, Athens, Greece
{hmarantos,dsoudris}@microlab.ntua.gr
[3] Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
angeliki.agathi.tsintzira@gmail.com, achat@uom.gr

Abstract. Maintaining high level of quality with respect to important quality attributes is critical for the success of modern software applications. Hence, appropriate tooling is required to help developers and project managers monitor and optimize software quality throughout the overall Software Development Lifecycle (SDLC). Moreover, embedded software engineers and developers need support to manage complex interdependencies and inherent trade-offs between design and run-time qualities. To this end, in an attempt to address these issues, we are developing the SDK4ED Platform as part of the ongoing EU-funded SDK4ED project, a software quality system that enables the monitoring and optimization of software quality, with emphasis on embedded software. The purpose of this technical paper is to provide an overview of the SDK4ED Platform and present the main novel functionalities that have been implemented within the platform until today.

Keywords: Software quality · Verification and validation

1 Introduction

The SDK4ED[1] project aims to minimize cost, development time, and complexity of low-energy software development processes, by providing an innovative platform for automatic optimization and trade-off calculation among important design-time and run-time software quality attributes. In brief, the main goal that this project aims to achieve is the provision of a set of solutions (i.e., toolboxes) integrated into the form of an easy-to-use platform, which will enable the developers and project managers of software products monitor and optimize the

[1] https://sdk4ed.eu/.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12252, pp. 1035–1050, 2020.
https://doi.org/10.1007/978-3-030-58811-3_73

quality of their produced software, with emphasis on the quality attributes of Maintainability, Dependability, and Energy Consumption. Moreover, additional mechanisms for quality forecasting and trade-off analysis will support decision-making throughout the overall software development cycle.

In summary, the purpose of the present technical paper is to provide an overview of the SDK4ED Platform and present the main novel functionalities that have been implemented within the platform until today. More specifically, in [1], an overview of the overall SDK4ED concept was presented and the internal architecture of the envisaged platform was provided based on the collected functional and non-functional requirements, as well as technical specifications. In the present paper, we delve into the technical characteristics of the platform and we demonstrate how these requirements have been realized so far into an actual functional platform.

This paper is expected to provide an early overview of the novel features that the SDK4ED consortium is working on within the context of the project, and demonstrate how these features are being merged into a unified and operationalized platform. In addition, this paper can be used by practitioners and researchers as a guideline in order to perceive how the features specified in [1] can be implemented in practice, using cutting-edge technologies (e.g., Microservices, Docker Engine, ReactJS, etc.), and, in turn, help them extend or even build similar platforms from scratch.

It should be noted that although several tools and platforms (either commercial or open-source) have been proposed over the years for quantifying and improving software quality [2–4], none of them has managed to cover both runtime (e.g., Energy Consumption) and design-time (e.g., Maintainability) quality attributes in a unified manner. In addition, none of the existing platforms focuses on the trade-offs between conflicting quality criteria, nor on the provision of recommendations for code refactoring taking into account the inter-relationships between different quality attributes. The SDK4ED project attempts to fill this gap by providing a platform that enables the evaluation of software with respect to both runtime and design-time quality attributes, and the improvement of its quality through refactoring recommendations that are produced based on the trade-offs among these quality criteria.

The main motivation of this paper is to raise the awareness of the software quality community about our ongoing work within the context of the SDK4ED project and especially regarding the novel features that we are integrating into a unified software engineering platform. We believe that a platform capable of monitoring important software quality attributes and evaluating the trade-offs between them is of paramount importance for both researchers and practitioners working on the embedded systems industry. In addition to this, we aim at showcasing how cutting-edge technologies are used for the implementation of the overall software evaluation platform, with the purpose to provide ideas to potential engineers for following a similar approach for the development of similar platforms.

The rest of the paper is structured as follows. Section 2 provides the overview of the SDK4ED Platform, along with important details regarding its technical implementation. In Sect. 3 a detailed description of the individual features (i.e., toolboxes) of the SDK4ED Platform is provided. In Sect. 4 the real-world applications that are used as pilot cases for the evaluation of the SDK4ED Platform are described. Finally, Sect. 5 concludes the paper and presents the next steps of the SDK4ED Project.

2 The SDK4ED Platform Overview

2.1 Overall Architecture

The high-level architecture of the SDK4ED Platform is illustrated in Fig. 1. As can be seen by Fig. 1 the overall platform consists of five core modules (i.e., toolboxes), namely Technical Debt Management, Energy Optimization, Dependability Optimization, Forecaster, and Decision Support. The first three modules provide features for monitoring and optimizing the quality attributes of Maintainability, Energy Consumption, and Dependability respectively. The latter two provide additional features (i.e., quality forecasts and trade-off analysis), which can further support decision-making during software development. At the moment, the SDK4ED Platform is able to analyze software applications written in Java, C, and C++ programming languages, covering a wide range of embedded software applications. To better understand the overall goal of the SDK4ED Platform, its core modules are briefly described in the rest of this section. A more detailed description is provided in Sect. 3.

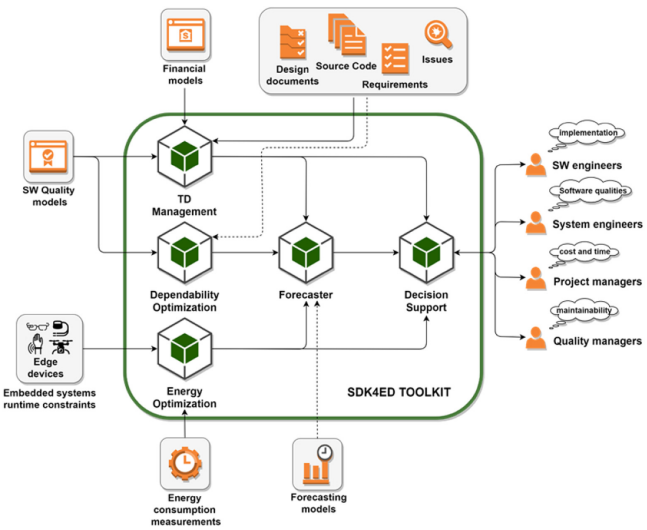

Fig. 1. The high-level overview of the SDK4ED platform.

Technical Debt Management. The Technical Debt (TD) management module is responsible for monitoring and optimizing the *Maintainability* of software products through the notion of Technical Debt [5]. More specifically, its main goal is to provide efficient support for TD quantification, prioritization, and handling in Embedded Software systems. TD is actually a measure of software Maintainability, similar to software metrics [6–9]. As discussed in Sect. 3.1, this is achieved through the provision of both state-of-the-art and novel mechanisms for TD Principal and TD Interest estimation, as well as extraction of refactoring opportunities.

Energy Optimization. The Energy Optimization module is responsible for measuring and minimizing the *Energy Consumption* of a given software application. In particular, as detailed in Sect. 3.2, this module provides mechanisms for estimating the energy consumption of a software application, identifying energy-hungry parts, and providing recommendations for potential energy optimizations.

Dependability Optimization. The Dependability Optimization module is responsible for assessing and optimizing the *Dependability* of software products. In particular, this module provides solutions for evaluating and optimizing two important facets of the Dependability [10] of the analyzed software, namely Security [11,12] and Reliability [13,14]. As discussed in Sect. 3.3, both security- and reliability-related services are provided by this module.

Forecaster. The Forecaster module is responsible for predicting the future evolution of the three quality attributes of interest, namely Maintainability (in fact, Technical Debt), Dependability, and Energy Consumption. This is achieved through the provision of advanced time series and machine-learning models (see Sect. 3.4). As depicted in Fig. 1, this module depends highly on the outputs of the three aforementioned modules.

Decision Support. The main purpose of the Decision Support module is to facilitate decision-making during the development and maintenance of embedded software applications. More specifically, as discussed in Sect. 3.5, the main responsibility of this module is to inform developers and project managers about the impact that refactorings for improving a specific quality attribute (e.g., Maintainability) will have on the other two attributes of interest (i.e., Dependability and Energy Consumption). Hence, this module receives input from all the other modules of the SDK4ED Platform and performs a trade-off analysis in order to determine these impacts and provide useful recommendations.

2.2 Technical Specifications

The SDK4ED Platform is actually a software quality monitoring system. As opposed to the majority of the existing software quality monitoring systems that are built as offline applications, we opted for a cloud-based web application.

The reasoning that led us to this option is (i) the increased visibility that a cloud-based platform offers, (ii) the high accessibility that is provided through the Internet, and (iii) the ease of use, since the tedious part of manually installing and configuring the tool is avoided. In addition to this, the platform can be used by end-users working on highly different operating systems, thus offering a cross-platform experience that would be extremely difficult to achieve in the case of an offline application.

For the development of the cloud-based SDK4ED Platform, we decided to use the Microservice Architecture (MOA) pattern [15], instead of the traditional Service-oriented Architecture (SOA) pattern. Both patterns are based on the concept of implementing the main functionalities of the application as individual services. The main difference between the two patterns is that while SOA requires all the services to be centrally implemented in the form of a monolithic application, in MOA, similar functionalities are grouped into components (i.e., microservices) with their own lifecycles, which can then be distributed over a network. These services can then collaborate together in order to form the broader application. Some of the main advantages of MOA over SOA (which helped us reach our final decision) are listed below [15]:

- Microservices can be deployed independently
- Microservices can be implemented using different technologies
- Microservices can be developed quickly, deployed and maintained by a small, independent team
- Microservices offer modular maintenance

Based on the MOA pattern, each one of the core modules of the SDK4ED Platform (see Fig. 1) has been implemented as an individual Microservice and the main functionalities of each module as individual web services. For the actual implementation of the Microservices, the Docker Engine[2] was used. Docker is an Enterprise Container Platform that allows applications to be packaged as individual containers along with their required parts (e.g., tools, configuration, dependencies, etc.) and communicate with each other through dedicated channels. In fact, the Microservices of the SDK4ED Platform are implemented as individual Docker Images, which are then deployed as independent Docker Containers.

Docker encompasses a set of key features that played an important role in our final selection. First of all, each Microservice (i.e., Docker Image) can be developed independently of one another, adopting different tools, frameworks, configurations, etc., without posing any restrictions to the other Microservices. Hence, Docker provides flexibility in the sense that it allows the individual applications to be implemented using highly different languages and technologies, encouraging in that way agile software development. In addition, the implemented Docker Images can be deployed and executed as Docker Containers either on the same machine or distributed over different locations. The execution of one Docker Container does not affect the execution of the others, increasing, in that

[2] https://www.docker.com/.

way, the reliability of the overall platform. Finally, the installation of the final integrated toolbox becomes an easy process, as it is reduced to the deployment of the individual Docker Containers that correspond to the Microservices of the platform, which can be performed automatically based on a description provided as a compose file.

Apart from the individual Docker Containers (i.e., Microservices) that form the back-end of the SDK4ED Platform, a front-end is also available in the form of a dashboard that facilitates its adoption in practice. The Home Page of the SDK4ED Dashboard is depicted in Fig. 2. The Home Page, built using the React[3] framework, provides a general overview of the analyzed projects by concentrating information retrieved from the individual web services.

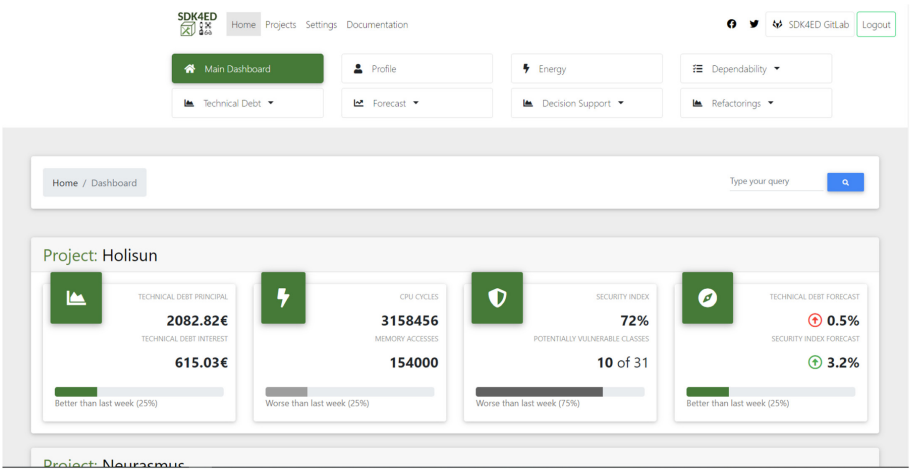

Fig. 2. The Home page of the SDK4ED Dashboard.

3 SDK4ED Toolboxes

3.1 Technical Debt Toolbox

The Technical Debt Management Toolbox is responsible for assessing the code quality (i.e., Maintainability) of a given software product, providing also suggestions on how to improve it (i.e., refactoring opportunities). It also provides novel mechanisms for assessing the TD that is introduced by new code added to the software product. More specifically, it comprises three main functionalities (i.e., components), which have been implemented in the form of individual web services. These services are described below:

[3] https://reactjs.org/.

- **Technical Debt Analysis:** This web service is responsible for analyzing and evaluating the TD aspects of a software application through its entire evolution. These TD aspects include TD Principal and Interest [16], as well as TD Interest Probability [17] and the Breaking Point [18] of the application. While the SonarQube[4] platform is used as the basis for the calculation of these TD-related aspects, Technical Debt Management Toolbox has been extended by introducing new tools developed in the context of the SDK4ED project to support the quantification of additional TD indicators (e.g., long methods, effort to resolve an inefficiency, etc.), as well as to provide appropriate rule configuration for the cases of C/C++, tailored to the needs of an embedded system developer.
- **Technical Debt New Code:** This web service is responsible for analyzing the commit history and providing a quality gate for the new code. In this way, if the new commit adds more TD in the existed code then the quality gate fails and an alert is sent to the user.
- **Refactorings:** This web service is responsible for analyzing the source code and suggesting potential refactoring opportunities based on Long Method code smell [19], as well as suggesting an optimal code design based on coupling and cohesion between the files.

The TD Management toolbox has been implemented as an individual Microservice, which encapsulates the three aforementioned web services. From a technical perspective, it has been deployed as an independent Docker Container, which actually constitutes the back-end of the TD Management toolbox. Dedicated front-end pages have been implemented as part of the broader SDK4ED Dashboard, in order to facilitate the invocation of the aforementioned services and the effective presentation of their results. In Fig. 3, the main front-end panel of the TD Management toolbox is presented.

As can be seen by Fig. 3, the user is able to select a specific project from the dedicated drop-down list, and then, the historical TD analysis is presented. More specifically, important TD-related measures are illustrated, including TD Principal, TD Interest, and TD Interest probability. Additional low-level information is also provided in the panel (e.g., code smells, bugs, etc.). Similar panels have been developed for the other two services. However, for reasons of brevity, screenshots are not provided in the present paper.

3.2 Energy Optimization Toolbox

The main purpose of the Energy Optimization toolbox is to provide support for measuring the energy consumption of a given software application, and for minimizing the energy consumption through the recommendation of energy optimizations that can be applied to the source code of the application. Hence, this toolbox provides two main functionalities, which have been implemented in the form of individual web services and are described below:

[4] https://www.sonarqube.org/.

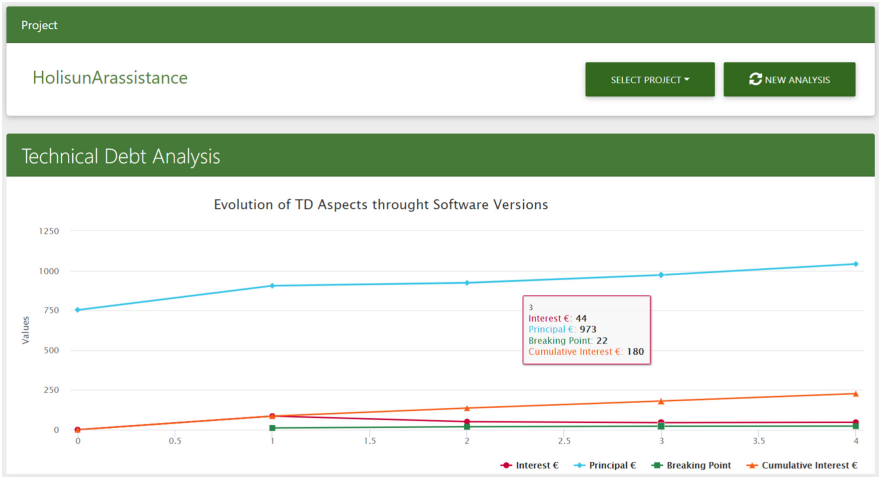

Fig. 3. The main front-end page of the Technical Debt toolbox.

- **Consumption Analysis**: The purpose of this service is the identification of the critical parts of the application source code in terms of CPU cycle. The identification of critical parts (i.e., hotspots) is based on the dynamic analysis of the application by integrating popular profiling tools, such as Valgrind[5] and Linux Perf[6]. The outcome of the analysis is a list containing the identified hotspots of the application source code (usually loops) and the values of a set of energy consumption indicators for each hotspot. The indicators are: (i) CPU-related (e.g., CPU cycles), (ii) memory-related (e.g., number of memory accesses) (iii) multithreaded-related (e.g., Lock contention), and (iv) accelerator-specific (e.g., memory/control/integer operations) indicators.
- **Estimation & Optimization.** This service is responsible for identifying possible optimizations at the level of the source code and then, for recommending these optimizations to application developers for improving the energy efficiency of the analyzed application. The identification of possible optimizations is indicated by the values of the energy consumption indicators for each one of the identified hotspots (as produced by the previous service). The service recommends the most suitable energy optimization for each one of the identified hotspots. These optimizations include: (i) data-flow optimizations (e.g., cache blocking), (ii) optimizations for concurrent data accesses (e.g., alternative lock mechanisms), and (iii) offloading recommendations (e.g., suggestion to offload the hotspot on GPU accelerators).

The aforementioned web services are part of the Energy Optimization toolbox Microservice, which, similarly to the other toolboxes, is deployed as an individual Docker Container. A dedicated front-end has been implemented as part of

[5] http://www.valgrind.org/.
[6] https://perf.wiki.kernel.org/index.php/Main_Page.

the SDK4ED Dashboard, with the purpose of facilitating the invocation of the services and presenting the results in an easy-to-understand manner. In Fig. 4 a screenshot of the Energy Optimization toolbox front-end page is illustrated. As can be seen by this figure, the user is informed about the hotspots that were identified in the analyzed project, along with their calculated energy indicators. For each hotspot, a specific energy optimization is provided in a separate table.

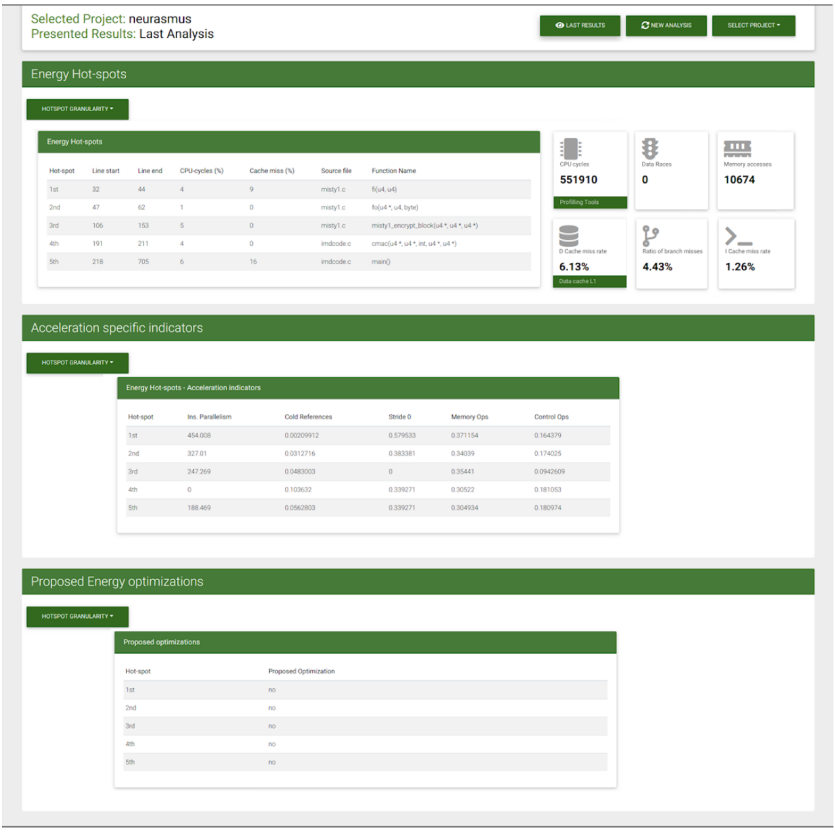

Fig. 4. The front-end page of the Energy Optimization toolbox.

3.3 Dependability Optimization Toolbox

As already mentioned, the purpose of the Dependability Optimization toolbox is to provide tools (i.e., services) for monitoring and optimizing the Dependability (i.e., Security and Reliability) of software applications. Hence, this toolbox offers three main features implemented in the form of individual web services. These services are listed below:

- **Quantitative Security Assessment:** This web service is responsible for evaluating the internal security level of a given software application. More specifically, it applies static analysis to a selected software application and aggregates the results in order to compute a high-level security indicator using a novel Security Assessment Model (SAM) [20], which has been developed within the course of the SDK4ED Project. For the case of software applications written in Java, the PMD[7] and CKJM Extended[8] static analysis tools are used by SAM, whereas for the case of C/C++ applications the CppCheck[9] and CCCC[10] static code analyzers are employed. The proposed model is based on popular international standards, including the ISO/IEC 25010 [21] and the ISO/IEC 27001 [22].
- **Vulnerability Prediction:** This web service is responsible for identifying security hotspots, i.e., parts of a given software product (e.g., classes and source code files) that are likely to contain vulnerabilities. Advanced Deep Learning models are applied (using the popular tensorflow[11] library) on the results of text mining in order to identify these vulnerability hotspots [11].
- **Optimum Checkpoint Interval Recommendation:** This web service is responsible for suggesting the optimum checkpoint interval for programs with loops. For this purpose, novel mathematical models for calculating the optimum checkpoint interval that optimizes important Quality-of-Service (QoS) attributes, including performance and energy consumption, have been developed and used as part of this module [23,24].

The Dependability Optimization toolbox has been implemented as an individual Microservice, which encapsulates the three aforementioned web services. From a technical perspective, it has been deployed as an independent Docker Container, which actually constitutes the back-end of the Dependability Optimization toolbox. To enhance the practicality of the Dependability Optimization toolbox, a front-end is also provided. The front-end of the Dependability Toolbox is part of the SDK4ED Dashboard (see the example in Fig. 5), whereas its purpose is to allow: (i) invocation of the individual web services, and (ii) retrieval and visualization of the produced results.

An example of the Dependability Optimization toolbox front-end is illustrated in Fig. 5. As can be seen by this figure, the user is able to select a project from a dedicated drop-down list and then decide whether they would like to perform a New Analysis or display the results of the previous analysis that has been performed for the selected project. Several useful information is displayed to the user regarding the selected project. For instance, in the example presented in Fig. 5, the user is informed about the overall Security Index of the selected application, whereas more detailed information about the dependability of the project is also provided through dedicated charts.

[7] https://pmd.github.io/.
[8] http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/.
[9] http://cppcheck.sourceforge.net/.
[10] http://cccc.sourceforge.net/.
[11] https://www.tensorflow.org/.

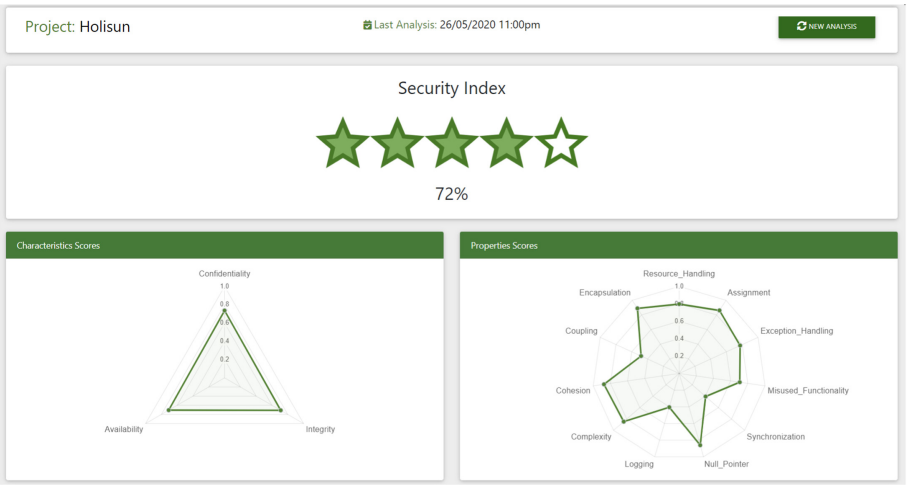

Fig. 5. The front-end page of the Dependability Optimization toolbox.

3.4 Forecasting Toolbox

The purpose of the Forecasting Toolbox is to provide predictive forecasts regarding the evolution of the three core quality attributes targeted by the SDK4ED platform, namely TD, Energy, and Dependability. Towards building a solid basis for the Forecasting Toolbox realization, the suitability of various dedicated time series was investigated [25], whereas advanced ML models have been introduced as part of the business logic of the toolbox, following the overall concept described in [26]. Hence, this toolbox offers three main features implemented in the form of individual web services. These services are briefly described below:

- **TD Forecaster:** This web service is responsible for generating TD forecasts for a given software application. A TD forecast represents the predicted evolution of the total remediation effort (measured in minutes) to fix all code issues (e.g. code smells, bugs, code duplication, etc.) of a software application, up to a future point specified by the user.
- **Energy Forecaster:** This web service is responsible for generating Energy forecasts for a given software application. An Energy forecast represents the predicted evolution of the total energy consumption (measured in Joules) of a software application, up to a future point specified by the user.
- **Dependability Forecaster:** This web service is responsible for generating Security forecasts for a given software application. A Security forecast represents the predicted evolution of the Security Index (see Sect. 3.3) of a software application, up to a future point specified by the user.

Similarly to the other Toolboxes of the SDK4ED platform, the Forecasting Toolbox has also been implemented as an individual Microservice, which encapsulates the three aforementioned web services and has been deployed as an

independent Docker Container. In addition to the back-end of the Forecasting Toolbox, a front-end is also provided as part of the overall SDK4ED Dashboard. More specifically, there are three panels on the navigation menu of the SDK4ED Dashboard that are dedicated to TD, Energy, and Security forecasting. As an example, the main screen of the TD Forecasting panel is depicted in Fig. 6.

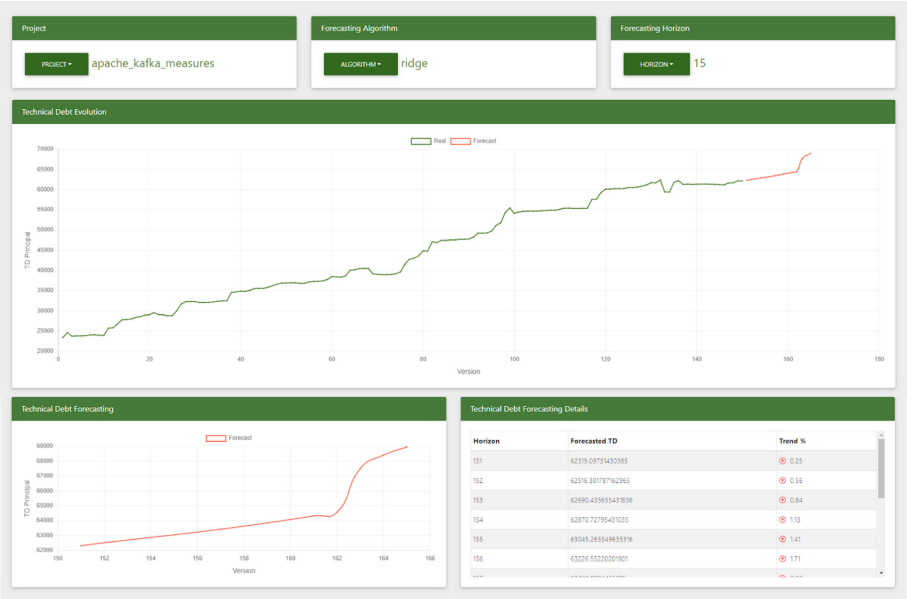

Fig. 6. The front-end page of the Forecasting toolbox.

The main screen of the TD Forecasting panel comprises a dropdown button, two interactive plots, and one table. The dropdown button allows the user to select the forecasting horizon for which they would like to see predictions. Once the forecasting horizon is selected, the back-end server invokes the proper forecasting algorithm (depending on the selected horizon) and returns the predictions back to the dashboard. The first plot shows the entire evolution followed by the forecasted evolution of the application, whereas the second plot focuses solely on the forecast, giving a more fine-grained view. In addition to the plots, a complementary table comprising the detailed results of the forecasts is presented at the bottom-right part of the screen. This table presents the forecasted TD values for the upcoming weeks, as well as the difference between the current TD value and the forecasted TD values per week, which may serve as an indicator of whether the TD Principal will increase or decrease, and to what extent. This additional information is expected to help the developers take even more informed decisions regarding the prioritization of their TD repayment activities.

3.5 Decision Support Toolbox

The main purpose of the decision support module is to facilitate the decision-making process during the development and maintenance of embedded software applications. More specifically, the main objective of this toolbox is to aggregate the refactoring suggestions provided by the individual toolboxes for improving a single quality attribute and then perform a trade-off analysis in order to determine how the refactorings for improving a specific quality attribute (e.g., Maintainability) may affect the other two quality attributes of interest (i.e., Dependability and Energy Consumption). This is expected to allow the developers and project managers make more informed decisions regarding which sub-group of refactorings to apply to the source code of their software application.

The Decision Support system is an ongoing research task, which is expected to be delivered as part of the final version of the SDK4ED Platform. Numerous theoretical and empirical analyses have been conducted within the course of the SDK4ED Project regarding the inter-relationships between the quality attributes of Maintainability, Dependability, and Energy Consumption [27–29]. The results of these studies identified various observable impacts that refactorings may have and led us to develop an approach (along with its tooling) that will perform this trade-off analysis. Concepts from the field of fuzzy multi-criteria decision making were also adopted (e.g., [30]). In addition to this, more advanced concepts for facilitating decision-making from interdisciplinary fields will be also studied (e.g., [31–33]).

4 Applications of the SDK4ED Platform

As already mentioned in Sect. 1, the SDK4ED Platform targets mainly the embedded software community, in which Energy Consumption and Dependability are aspects of major concern. However, it should be noted that the SDK4ED Platform is not limited only to embedded software and can be leveraged for improving the quality of any kind of software in general. Within the context of the SDK4ED project, real-world software applications were utilized as the basis for showcasing the capabilities of the produced platform, as well as for evaluating its correctness. These applications come from the airborne, automotive, and healthcare domains, and are provided by three industries, namely Airbus, Holisun, and Neurasmus respectively.

Regarding the airborne use case, embedded software running on Unmanned Aerial Vehicles (UAV) provided by Airbus Defence is used as a testbed. These software applications are written in C/C++ programming language and are responsible for critical tasks of UAVs missions, including algorithms for real-time obstacle avoidance, algorithms for improving autonomy capacity, and algorithms for route planning and navigation. Airbus is interested in optimizing the software running on UAVs with respect to their energy consumption, as improved energy efficiency will lead to longer flight duration, maintaining at the same time high-levels of Dependability and Maintainability.

As far as the automotive use case is concerned, specific focus is given on augmented reality software applications that are running on smart glasses, with the purpose to facilitate the remote assistance of technicians operating in the industrial field with engineers in support centers. Solutions provided by Holisun are used as testbeds. These software applications are actually Java (i.e., Android) applications, with high demands in computational power, data transmission volumes, and latency. Holisun, through the adoption of the SDK4ED Platform, is willing to improve the real-time performance and energy efficiency of their application, while maintaining their Dependability and Maintainability.

Finally, as far as the healthcare use case is concerned, embedded software running on implantable devices is used as a testbed. These applications, which are developed by Neurasmus, are written in C programming language and they are characterized by high demands with respect to energy efficiency and dependability. Hence, through the SDK4ED Platform, Neurasmus will attempt to improve the energy consumption and the dependability of their applications, while maintaining the code sufficiently maintainable.

5 Conclusion

In this paper, we presented the current state of the SDK4ED Platform (developed as part of the ongoing EU-funded SDK4ED Project), the purpose of which is to enable the monitoring and optimization of important quality attributes (i.e., Maintainability, Energy Consumption, and Dependability) of software applications, with emphasis on embedded software. More specifically, the overview of the SDK4ED Platform was initially provided, followed by a more detailed description of its core modules. Emphasis was given on their technical characteristics, as well as on the main research achievements that have been achieved so far. Building upon [1], in which the overall concept, requirements, and architecture of the SDK4ED Platform were presented, this paper demonstrated our progress towards the actual realization of these specifications into an actual operationalized platform.

In the remaining period until the end of the project, emphasis will be given on the finalization of the Decision Support module of the SDK4ED Platform, which is the only module currently in progress. More specifically, a novel trade-off analysis mechanism will be developed, which will be based on the outputs of the other four modules of the platform and multi-criteria decision-making techniques that encompass fuzzy logic. Subsequently, the effort will be directed towards the implementation of the final version of the SDK4ED Platform, and its release as an open-source software application.

Acknowledgements. This work is partially funded by the European Union's Horizon 2020 Research and Innovation Programme through SDK4ED project under Grant Agreement No. 780572.

References

1. Jankovic, M., Kehagias, D., Siavvas, M., Tsoukalas, D., Chatzigeorgiou, A.: The SDK4ED approach to software quality optimization and interplay calculation. In: 15th China-Europe International Symposium on Software Engineering Education (2019)
2. Heitlager, I., Kuipers, T., Visser, J.: A practical model for measuring maintainability. In: 6th International Conference on the Quality of Information and Communications Technology (2007)
3. Wagner, S., et al.: Operationalised product quality models and assessment: the quamoco approach. Inf. Softw. Technol. **62**, 101–123 (2015)
4. Siavvas, M.G., Chatzidimitriou, K.C., Symeonidis, A.L.: Qatch-an adaptive framework for software product quality assessment. Expert Syst. Appl. **86**, 350–366 (2017)
5. Cunningham, W.: The wycash portfolio management system. ACM SIGPLAN OOPS Messenger **4**(2), 29–30 (1993)
6. Misra, S., Akman, I., Colomo-Palacios, R.: Framework for evaluation and validation of software complexity measures. IET Softw. **6**(4), 323–334 (2012)
7. Misra, S., Adewumi, A., Fernandez-Sanz, L., Damasevicius, R.: A suite of object oriented cognitive complexity metrics. IEEE Access **6**, 8782–8796 (2018)
8. Kumar, L., Misra, S., Rath, S.K.: An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes. Comput. Stand. Interfaces **53**, 1–32 (2017)
9. Baski, D., Misra, S.: Metrics suite for maintainability of extensible markup language web services. IET Softw. **5**(3), 320–341 (2011)
10. Sommerville, I.: Software Engineering. Addison-Wesley, Boston (1995)
11. Siavvas, M., Gelenbe, E., Kehagias, D., Tzovaras, D.: Static analysis-based approaches for secure software development. In: Gelenbe, E., et al. (eds.) Euro-CYBERSEC 2018. CCIS, vol. 821, pp. 142–157. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95189-8_13
12. Gelenbe, E., et al.: NEMESYS: enhanced network security for seamless service provisioning in the smart mobile ecosystem. In: Gelenbe, E., Lent, R. (eds.) Information Sciences and Systems 2013. LNEE, vol. 264, pp. 369–378. Springer, Cham (2013)
13. Behera, R.K., Shukla, S., Rath, S.K., Misra, S.: Software reliability assessment using machine learning technique. In: Gervasi, O., et al. (eds.) ICCSA 2018. LNCS, vol. 10964, pp. 403–411. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95174-4_32
14. Shukla, S., Behera, R.K., Misra, S., Rath, S.K.: Software Reliability Assessment Using Deep Learning Technique. Towards Extensible and Adaptable Methods in Computing (2018)
15. Wolff, E.: Microservices: Flexible Software Architecture. Addison-Wesley, Boston (2016)
16. Ampatzoglou, A., Michailidis, A., Sarikyriakidis, C., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P.: A Framework for Managing Interest in Technical Debt: An Industrial Validation (2018)
17. Charalampidou, S., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P.: Assessing code smell interest probability: a case study. In: XP2017 Workshops (2017)
18. Chatzigeorgiou, A., Ampatzoglou, A., Ampatzoglou, A., Amanatidis, T.: Estimating the breaking point for technical debt. In: IEEE 7th International Workshop on Managing Technical Debt (2015)

19. Charalampidou, S., Arvanitou, E.M., Ampatzoglou, A., Avgeriou, P., Chatzigeorgiou, A., Stamelos, I.: Structural Quality Metrics as Indicators of the Long Method Bad Smell. In: 44th Conference on Software Enginering and Advanced Applications (2018)
20. Siavvas, M.: Static analysis for facilitating secure and reliable software. Ph.D. thesis, Imperial College London (2019)
21. ISO/IEC: ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. ISO/IEC (2011)
22. ISO/IEC: ISO/IEC 27001:2013(en) Information technology - Security techniques - Information security management systems - Requirements. ISO/IEC (2013)
23. Siavvas, M., Gelenbe, E.: Optimum checkpoints for programs with loops. Simul. Modell. Practice Theory **97**, 101951 (2019)
24. Siavvas, M., Gelenbe, E.: Optimum interval for application-level checkpoints. In: 6th International Conference on Cyber Security and Cloud Computing (2019)
25. Tsoukalas, D., Jankovic, M., Siavvas, M., Kehagias, D., Chatzigeorgiou, A., Tzovaras, D.: On the Applicability of Time Series Models for Technical Debt Forecasting. In: 15th China-Europe International Symposium on Software Engineering Education (2019)
26. Tsoukalas, D., Siavvas, M., Jankovic, M., Kehagias, D., Chatzigeorgiou, A., Tzovaras, D.: Methods and tools for td estimation and forecasting: a state-of-the-art survey. In: 2018 International Conference on Intelligent Systems (IS) (2018)
27. Papadopoulos, L., Marantos, C., Digkas, G., Ampatzoglou, A., Chatzigeorgiou, A., Soudris, D.: Interrelations between software quality metrics, performance and energy consumption in embedded applications. In: Proceedings of the 21st International Workshop on Software and Compilers for Embedded Systems (2018)
28. Siavvas, M., et al.: An empirical evaluation of the relationship between Technical Debt and Software Security. In: 9th International Conference on Information Society and Technology (2019)
29. Siavvas, M., Marantos, C., Papadopoulos, L., Kehagias, D., Soudris, D., Tzovaras, D.: On the relationship between software security and energy consumption. In: 15th China-Europe International Symposium on Software Engineering Education (2019)
30. Guo, S., Zhao, H.: Fuzzy best-worst multi-criteria decision-making method and its applications. Knowl. Based Syst. **121**, 23–31 (2017)
31. Behera, R.K., Rath, S.K., Misra, S., Damaševičius, R., Maskeliūnas, R.: Large scale community detection using a small world model. Appl. Sci. **7**(11), 1173 (2017)
32. Vafeiadis, T., et al.: Data analysis and visualization framework in the manufacturing decision support system of COMPOSITION project. Procedia Manuf. **28**, 57–62 (2019)
33. Behera, R.K., Naik, D., Rath, S.K., Dharavath, R.: Genetic algorithm-based community detection in large-scale social networks. Neural Comput. Appl. **32**(13), 9649–9665 (2019). https://doi.org/10.1007/s00521-019-04487-0