The SDK4ED Approach to Software Quality Optimization and Interplay Calculation

Marija Jankovic*, Dionysios Kehagias*, Miltiadis Siavvas*, Dimitrios Tsoukalas*, Alexander Chatzigeorgiou†,
* *Centre for Research and Technology Hellas, Thessaloniki, Greece*
† *Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece*
jankovicm@iti.gr, diok@iti.gr, siavvasm@iti.gr, tsoukj@iti.gr, achat@uom.gr

***Abstract*—While necessary for the successful embedded software development and maintenance, software quality optimization is a complex activity with immense issues. Various design and run-time qualities should be continuously monitored and optimized during the whole Software Development Life Cycle (SDLC). Moreover, embedded software engineers and developers need to manage complex interdependences, and inherent trade-offs between design and run-time qualities. The paper presents an innovative approach and integrated platform to resolve these complexities. The proposed approach is a result of the European research project SDK4ED (Software Development Toolkit for Energy Optimization and Technical Debt Elimination). The paper identifies existing challenges in the domain of embedded software quality optimization and points at the benefits of the new approach.**

***Keywords*—quality optimization, embedded systems, technical debt, energy efficiency, dependability**

I. Introduction

The software engineering research has been focused on the improvement of software quality in various phases of Software Development Life Cycle (SDLC), from requirements specification to final integration and testing activities [1], [2]. However, to ensure software quality in today's sophisticated IoT-based platforms, which often include various embedded devices, is still challenging. Embedded system applications should satisfy strict run-time constraints, such as energy efficiency, performance, and reliability. As most of the embedded devices are battery dependent, the need for ultra-low power consumption is of supreme importance. To reduce the embedded system power consumption, various techniques at both hardware and system level should be combined.

Investment in the embedded software quality is gaining more interest, as the hardware limitations are usually more challenging and costlier to implement. Moreover, the life-time expectancy of embedded systems is continuously increasing, and brownfield systems should be maintained in parallel with the design and implementation of the greenfield ones. As a consequence, maintenance is characterized as one of the most time and cost demanding activities in the SDLC [3]. Still, most of the embedded software companies trade design-time quality to shorten product time-to-market development cycle. Moreover, to satisfy the ultra-low power operation requirement, embedded software developers are opting for non-clean and unconventional code development. As a consequence, these actions are leading to increased technical debt, which can be characterized as a financial overhead in future maintenance activities because of shortcuts taken during development [4].

However, there is currently a lack of support in the broader embedded systems industry settings to find the right balance between often conflicting design (e.g., maintainability) and run-time (e.g., energy-efficiency, performance) quality attributes. Specifically, the lack of methods and supporting tools presents a problem for a broad community of embedded software engineers and developers. To address these needs, the SDK4ED project will develop a novel Embedded Systems (ES) specific development platform for automatic optimization of various design-time and run-time quality attributes. Moreover, SDK4ED platform is going to support advanced trade-off analysis between selected design-time and run-time constraints.

In summary, this paper presents the SDK4ED approach to enable the detailed specification and implementation of an innovative platform for low-power embedded systems design and analysis to support quality optimization and provide trade-off recommendations. It discusses the current issues from a low-power embedded systems point of view and gives the recommendations for the application of the novel approach.

The rest of the paper is structured as follows. Section II discusses related work. Section III presents the SDK4ED objectives, while Section IV describes the overall approach. Section V illustrates the SDK4ED high-level conceptual architecture. Finally, conclusions and plans are summarized in Section VI.

II. Related Work

In this section, we discuss the state of the art of existing software quality models and indicators. The most well-known standard to measure software product quality is the international ISO/IEC 25010 standard [1]. Although ISO/IEC 25010 defines eight quality attributes and relevant sub-categories, its main drawback is that it does not specify which metrics should be used for software quality measurement.

A. *Design-time Software Quality Attributes*

The embedded software community often overlooks the importance of design-time qualities and related quality models and metrics. Although there are various metrics for assessing software **maintainability** and **reusability** in the literature [5], [6], these metrics are still lacking precision and do not take into account software development history and particular domain-specific characteristics of software. Recently the **Technical Debt (TD)** Metaphor has been introduced to express in monetary terms the additional maintenance cost caused by presence of inefficiencies in an existing software system [4]. Several empirical studies have indicated the importance of efficient TD management [3], [7]. Various tools have been introduced to estimate the principal of Technical Debt, that is, the effort required to resolve all identified issues and bring a software system to a near-optimum state in terms of internal software quality [8]. However, the underlying rulesets are generic and domain-agnostic and, as a result, the identified technical debt liabilities may or may not be relevant to embedded system applications.

According to [9], one of the most critical systems property is the dependability, which covers various system quality attributes (i.e., security, reliability, and availability). The security of software applications is a broad aspect that can be treated both as a design-time and as a run-time quality attribute [10]. Traditionally, software security, as a run-time attribute, was considered as an added feature in the overall SDLC [11]. However, the inadequacy of external protection mechanisms to fully protect software applications against attacks forced software development enterprises to shift their focus towards **security by design**, which corresponds to building software applications that are highly secure initially [12]. Measuring software security is important for decision-making. Although mature techniques for measuring the external security of software products exist (e.g. the Attack Surface Metrics), no well-accepted models can be found in the related literature for quantifying the internal software security [13]. Apart from these issues, no security assessment model specifically focusing on evaluating the security level of low-power embedded system applications exists in the related literature. The identification of potential indicators of vulnerabilities is another research topic that has recently attracted the attention of the research community [12]. Although several software-related factors have been studied for their ability to indicate the existence of vulnerabilities in software products in general, no studies have particularly focused on IoT-based embedded system applications [12].

SDK4ED will contribute novel methods and tools for improving and optimizing design software qualities and especially maintainability, security by design and reliability.

B. Run-time Software Quality Attributes

Energy efficiency is an important run-time quality, and its optimization is a main objective of many studies in the relevant literature [14]. The problem of efficient energy management of IoT embedded devices is studied in [15]. Energy consumption occurs at hardware level, but in the embedded software domain, it is software executing on programmable hardware platforms that determines the overall energy consumption. The SDK4ED will build upon the large amount of existing software optimization techniques to reduce the energy consumption on low-power devices (e.g., loop unrolling) [14]. It is important to point that software optimization to increase **performance**, and consequently reduce energy consumption often fails to provide design guidelines and might incur TD [16]. In other words, current state of practice in embedded system design often completely neglects the notion of TD resulting in systems that are very expensive to reuse and maintain.

C. Trade-offs Between Run-time and Design-time Quality Attributes

The interrelationships between various design-time qualities have been studied in relevant literature. For example, authors in [17] have performed an empirical study to evaluate the interplay between software security and technical debt. Similarly, the relationships between multiple run-time quality attributes have been the subject of several studies [18], [19].

To the best of our knowledge, the SDK4ED is going to be the first platform that will handle the interplay between various, and often conflicting quality attributes such as maintainability, dependability, and energy efficiency. However, there are several studies with the focus on the interplay between design-time and run-time qualities [16], [20], [21]. The authors in [16] state that the consequences of various techniques that embedded software developers apply to improve run-time qualities such as energy consumption or performance might affect the design-time qualities of the software (e.g., maintainability or reliability). Likewise, the various refactoring techniques for source code optimization might have direct negative or positive consequences on the run-time quality attributes. The authors in [16] have concluded that various interconnections between design and run-time qualities exist, and pointed out the need for further investigation in this field.

III. Objectives of the Project

The SDK4ED will address the challenges discussed in the previous sections under a common integrated platform, by pursuing a number of technical, business and social objectives.

A. Technical Objectives

SDK4ED should provide novel methods and services/tools integrated under common platform, which will continuously monitor and manage software qualities at different software development phases (design, implementation, etc.) and levels of granularity (component, class, method, etc.). Envisioned SDK4ED platform should enable embedded systems industries to deliver quickly high-quality software by maximizing levels of important run-time and design-time quality parameters with emphasis on maintainability, security, reliability and energy consumption. Moreover, it should include support for continuous optimization of trade-offs between various design-time and run-time qualities by applying big data and software analytics technologies. This is particularly important, since the application of refactoring activities, like repaying TD on selected components, might affect their run-time qualities (e.g., energy efficiency or execution performance).

B. Business Objectives

SDK4ED should establish set of forecasting methods and best practices to assist embedded system engineers and project managers in making decisions regarding the choices for software quality improvements like TD refactoring. Moreover, SDK4ED should demonstrate the usability and efficiency of delivered set of methods and tools, through number of representative use cases, which should be deployed in a real operational environment.

C. Social Objectives

We consider it important to establish a culture of paying attention to accumulated TD at all levels of SDLC. Thus, we have set the objective to illustrate the importance of TD and demonstrate the benefits of proper TD management for low-energy software application development. The most convincing approach to illustrate the importance of TD is to empirically quantify its impact on the cost of developing and maintaining embedded system software.

IV. Overall Approach

SDK4ED has an agile, user-centered methodology, which ensures the acceptance and usability of produced outcomes. The envisioned SDK4ED platform aims at facilitating the high-level needs of all stakeholders involved in the design, development and deployment of low-power embedded systems like software engineers, quality managers, project managers, and embedded system engineers as illustrated in Figure 3. In the present section, we briefly present the main phases and steps of the overall SDK4ED development approach.

A. Conceptual Design

1) Functional, non-functional, and hardware requirements specification: We derived **functional requirements** based on the results of the empirical study performed together with the use case providers. According to the guidelines suggested in [22], [23], we designed an exploratory, embedded and multiple-case study. The goal of the case study was to understand the needs and challenges faced by embedded systems developers, system engineers and quality managers. Particularly, we designed the questionnaire with the goal to identify desired use case provider's design-time and run-time qualities and complex interrelationships between required trade-off decisions. First, data were collected using both interview and focused group instruments [24], [25]. Second, the data collection analysis was performed according the recommendations in [26], [27]. Next, the functional requirements of the SDK4ED platform were documented following the IEEE 29148-2011 standard guidelines [28]. Similarly, the overall elicitation process of **non-functional requirements** was based on the well-established ISO/IEC 25010 standard, which constitutes the de facto standard for software quality evaluation and modelling [1]. Based on quality aspects described in [1], we selected seven representative non-functional requirements categories (maintainability, serviceability and manageability, performance, reliability, availability, security and usability). The elicited non-functional requirements were prioritized using MoSCoW technique [29], which led to the definition of the final subset of 65 non-functional requirements, where 42 were considered highly significant from end-user point of view. Besides, we had to perform the detailed analysis and specification of **hardware requirements**. Specifically, we identified the requirements of the SDK4ED use case applications, taking into account required run-time qualities that should be monitored. The analysis showed that the current use case platforms do not support measurement of run-time energy and power consumption. Finally, a list of monitors and additional functionalities, which should be supported by the SDK4ED platform, has been compiled.

2) System analysis and architecture specifications: Considering specific functional, non-functional and hardware requirements, based on IEEE 1471 standard recommendation, we have selected three viewpoints for envisioned SDK4ED platform (i.e., logical, functional and deployment) [30]–[33]. **Logical viewpoint** is designed starting from functional requirements specification. For the initial system analysis and design, we have selected Data Flow Diagram (DFD) technique, which provides a convenient graphical means for analyzing the system's structure at different levels of abstraction [33], [34]. At the top level of abstraction, the SDK4ED system is represented as a black box using a context diagram illustrated in Figure 1. The SDK4ED context diagram was useful to: (1) visualize the scope of the system, (2) highlight the roles that interact with the system, (3) and to summarize input and output flows. For example, it is obvious that SDK4ED user (e.g., embedded system developer) requests the TD feedback from the SDK4ED platform, and receives the appropriate TD assessment result. Moreover, the context diagram indicates that various external tools (e.g., static code analysis) will be employed to perform required code analysis. Context diagram is further elaborated and decomposed into corresponding Level 1 and Level 2 diagrams, which where useful for SDK4ED component identification and specification. Actually, the resulting high-level SDK4ED conceptual diagram, which is given in Figure 1, maps the SDK4ED processes from the Level 1 into conceptual entities (i.e., modules) and connections between them (i.e., associations).

Another useful viewpoint was functional, where we defined interactions and interfaces between components using both high-level system and specific MVC-based SDK4ED sequence diagrams [35], [36]. Finally, we specified **run-time and deployment viewpoint** to provide indicative software and hardware specifications for each system component.

B. Technical Development

1) Research and innovation to implement the architecture: Research and innovation to implement the architecture. In this step, two essential activities are of interest. The first one, which is the most important task, is to define an appropriate list of indicators for each software quality. For that purpose, we have performed the following activities for the core quality attributes, i.e., maintainability, dependability and energy ef-

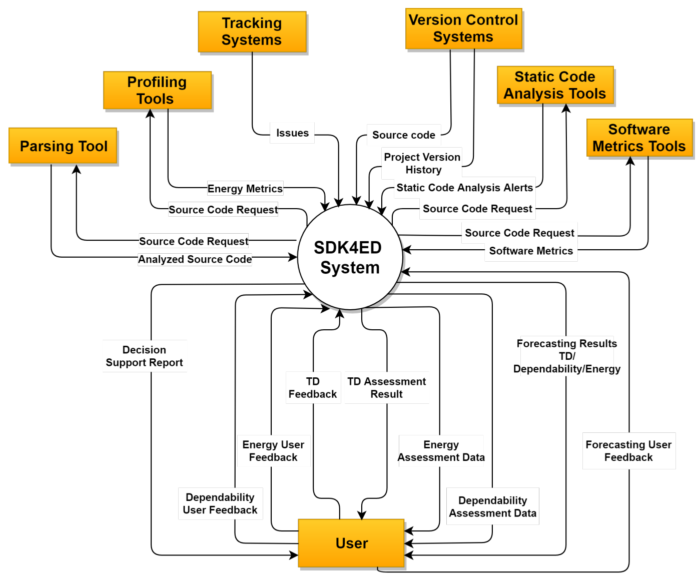

Fig. 1. The Level Zero Data Flow Context Diagram of the SDK4ED Platform

ficiency. First, we conducted a detailed state-of-the-art study to select candidate indicators for realization in the SDK4ED platform. To achieve this goal, we used various sources (i.e., existing scientific literature, tool-kits and services available on the web). As a result, we defined the initial set of indicators, and specified the gaps that should be addressed as part of the SDK4ED project. However, the final set of indicators is selected based on the analysis of industrial partner's needs through a number of relevant empirical studies. The selected indicators for inclusion in the SDK4ED platform are specified per programming paradigm (i.e., object-oriented and non-object oriented) and language (Java, C, C++). Moreover, we discussed the existing tools that might be reused or extended and indicate novel tools that should be developed.

In the second task, appropriate optimization techniques and algorithms will be investigated. In particular, the goal is to examine the suitability of existing approaches first, and, if necessary, to develop novel optimization techniques for all selected TD, security and energy efficiency indicators. Furthermore, in this task, we will perform a design space exploration that will allow the users to choose the Pareto optimal point in terms of TD, energy efficiency and security optimizations.

2) Development of toolkits for handling quality requirements: Development of toolkits is an ongoing activity. We selected the micro-service architectural pattern as the one that best meets the functional and non-functional requirements of the SDK4ED platform [37]–[39].

Six core quality attributes (agility, portability, testability, performance, scalability and ease of development) were selected as a basis for analysis. A number of advantages have contributed to the selection of micro-service architecture over layered, event-driven, micro-kernel and SOA architectural styles [38]. As illustrated in the Figure 2, each of the core

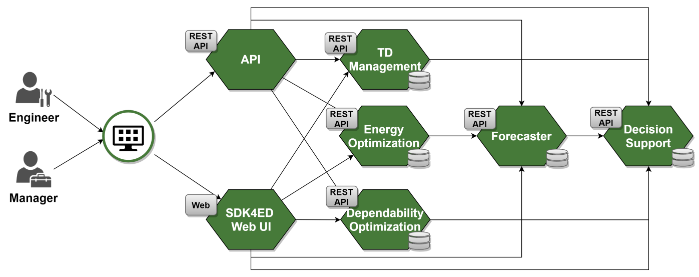

Fig. 2. High-level view of the SDK4ED Platform using micro-service pattern

modules, namely TD Management, Energy Optimization, Dependability Optimization, Forecaster and Decision Support are going to be implemented as individual services. The different services will be available to the users through Web UI or relevant APIs. Moreover, the ability to exploit the independent micro-services rather than complete functionality, will greatly facilitate the implementation, deployment, testing and eventually use of the platform.

3) System integration and testing strategy: System integration and testing strategy will be implemented on top of continuous integration principle, and will include three different levels of testing (i.e., unit level testing, integration testing and system testing).

C. Proof-of-Concept

1) Deployment of the pilot case studies in a real industrial environment: The representative use cases from the UAV (Unmanned Aerial Vehicle), health-care and automotive industries will be deployed in a real operational environment to assess the impact of the project's innovations, and practical advantages.

2) End-user verification and validation acceptance based on the fulfillment of KPIs (Key Performance Indicators): We will pay specific attention to provide high-quality educational service and supporting training materials. Two sets of training materials will be produced. A first set of the general training materials will be delivered after the integration of individual architectural components, and will incorporate detailed instructions followed by examples. However, towards the end of the project, training material will be specialized for specific target groups of users.

V. SDK4ED Integrated Platform

The SDK4ED platform will provide recommendations for the code optimization by parsing software artifacts (e.g., source code, design models or test cases) and analyzing these items from the perspective of various TD, energy and software security indicators. The high-level architecture of SDK4ED system, including the five core modules and their relationships to the external sources of information, are represented in Figure 3.

A. TD Management Module

The main goal of the TD management module is to provide efficient support for TD quantification, prioritization and han-

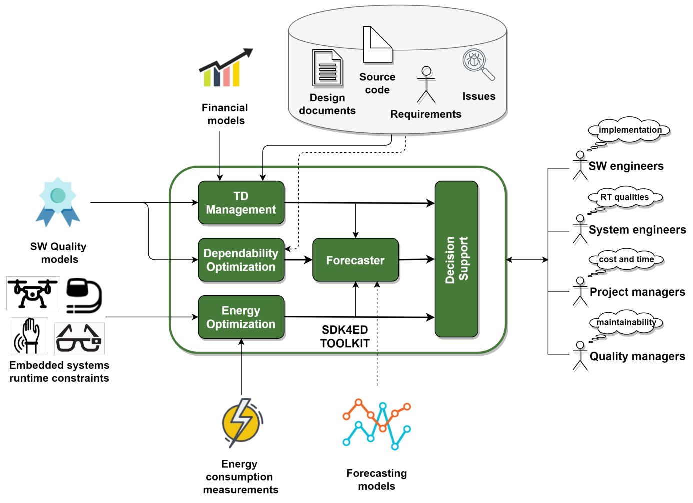

Fig. 3. The high-level overview of the envisaged SDK4ED Toolkit platform

dling in Embedded Software systems. Specifically, TD management module will be comprised of four main components: TD principal, TD interest, TD on new code estimation and Extraction of refactoring opportunities.

TD principal component will provide support for quantification of the effort that is required to address the difference between current and optimal level of design-time quality. TD Principal is often quantified through popular state-of-the-art tool SonarQube[1]. We are planning to extend the SonarQube to support quantification of the TD principal indicators (e.g., long method identification, effort to resolve inefficiency in minutes/currency, etc.). However, several limitations that should be addressed have been identified. For example, while Java programming language is well supported, we need to provide an appropriate rule configuration specifically for C and C++ that can be tailored to the needs of the corresponding embedded system developer (e.g., by enabling/disabling rules to be checked).

TD interest component will handle the additional effort that needs to be spent on maintaining the software because of decayed design-time quality. TD Interest estimation is vital, as it dictates the management decisions to repay TD. However, TD interest lacks appropriate quantification, and novel tools need to be implemented for both Java and C++ programming languages.

TD on the new code estimation component will monitor the evolution of the TD principal by comparing each code revision to the existing one. SonarQube aims to provide this feature but is not yet operational. The SDK4ED TD management toolkit will address this gap, by providing support for quantification of novel TD density on new code indicator.

Extraction of refactoring opportunities will provide support for both orthogonal and interdependent refactoring. Orthogonal refactoring component will show total number of refactoring for each quality attribute, which can be applied without having side effects on other software qualities. On the contrary, interdependent refactoring component will indicate suggestions that have possible side effects on other qualities.

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

B. Dependability Module

The main goal of the Dependability module is to provide full support for the assessment of the dependability of the embedded software solutions. Towards this direction, we have specified four components: Quantitative Security Assessment, Vulnerability Prediction, Exploitable Vulnerability Identification and Optimum Checkpoint Recommendation.

Quantitative Security Assessment component will implement the novel hierarchical security assessment model based on static analysis alerts and software metrics, which we initially developed for the quantification of the security level of software applications written in Java programming language. Afterwards, we extended the model to support the evaluation of software applications in C and C++ programming languages. We performed thorough evaluation of the proposed model, based on large volume of empirical data [40], [41].

Vulnerability Prediction component will be focused on the prediction of the existence of security issues or vulnerabilities. Specifically, we will implement a novel static-analysis model for prioritizing testing efforts, by identifying potentially vulnerable software components. We will focus our effort primarily on achieving a satisfactory trade-off between vulnerability prediction accuracy and performance.

Exploitable Vulnerability Identification component will implement the novel model for the assessment of the security-related artifacts produced by automatic static analysis tools. The purpose of the model is to enable identification of exploitable vulnerabilities, i.e., the vulnerabilities that are more likely to cause actual security issues.

Optimum Checkpoint Recommendation component will focus on identifying judicious locations for adding checkpoints and provide recommendations regarding their optimum inter-checkpoint interval. These recommendations are expected to enhance the reliability of the produced software without affecting other quality attributes like performance and energy consumption.

C. Energy Optimization Module

The aim of the Energy Optimization module will be based on two core components: Consumption Analysis and Energy Optimization. Consumption Analysis component will provide energy consumption analysis in both horizontal and vertical layers of the underlying architecture.

Energy Optimization component will provide relevant tools for energy estimation, optimization and measurement. The final set of indicators has been selected and its applicability in the context of the SDK4ED pilot use cases has been validated based on several empirical case studies. The indicators are grouped into four categories namely, CPU-related, Memory-related, Multithreading-related and related to accelerators.

D. Forecaster Module

Forecaster module will provide support for prediction of the three core quality attributes targeted by the SDK4ED platform: Technical Debt, Energy and Dependability. As a first step towards TD Forecasting component realization, we have developed and applied specific time series models for TD forecasting. Currently, we are investigating the ability of popular methods, such as Causal or Associative models as well as Machine Learning models, such as Support Vector Regression, Regression Trees, or Artificial Neural Networks to forecast the evolution of TD and Dependability indicators [42]. The special emphasis will be placed on modeling software evolution on the system level, rather than predicting the evolution of individual system properties.

E. Decision Support Module

The main purpose of the decision support module is to facilitate the decision-making process of various stakeholders involved in the development and maintenance of embedded software applications. Specifically, the end users are going to have an overview of the alternatives for improving the current software design. In particular, the module will have two core components: Refactoring Suggestions and Quality Attribute Impact Estimation.

Refactoring Suggestions component will provide semi-automated support for decision-making process on deciding on the refactoring priority. The ranking of the suggestions will be formulated based on TD, energy and dependability benefits/costs. As indicated in Figure 3, Decision Support module will receive required inputs from all other modules.

Quality Attribute Impact Estimation component will implement a trade-off management strategy. Having in mind that TD, energy efficiency and dependability of embedded systems are contradicting qualities, the component will provide two main functions. First, design-search space under at least two search parameters (i.e., TD, energy, dependability) will be provided. Next, the results will be formulated in the form of the Pareto frontier consisting of the Pareto efficient allocations, depicted graphically. The work on the investigation of the existence of the relations and trade-offs between optimizations for eliminating TD, and for improving energy efficiency and security is an ongoing activity. The current results have been reported in [16].

VI. Conclusions and Future Work

In this paper, we presented an innovative approach to design-time (i.e., maintainability, dependability) and run-time (energy-efficiency, performance) optimization, and interplay calculation. The three main phases of the SDK4ED project approach (conceptual design, technical implementation and proof-of-concept) are described, and corresponding steps within each phase are explained. The expected outcome of the project is the ES-specific integrated platform, which will be implemented on the top of novel quality models for quantification of specified indicators (e.g., software security model) and advanced optimization algorithms.

SDK4ED detailed system architecture is specified, following the rules of the micro-service architectural pattern, which is selected as a most appropriate option for the implementation effort. A comprehensive state-of-the-art study is performed, and suitable indicators are suggested for each core quality attribute. Future work is planned in several directions. We plan to use the experience and results from the specification of suitable quality indicators as a valuable input for the development of required optimization techniques and algorithms. Another planned activity is specification and implementation of trade-offs between each refactoring suggestion for TD, security and energy efficiency. In parallel with aforementioned research challenges, we will continue with the implementation of individual modules.

In summary, through technological innovations it introduces, the SDK4ED integrated platform is expected to:

- provide practical guidance and prompt support to ES software developers and engineers for monitoring, quantification, optimization, and forecasting of desired software qualities;
- efficient decision support for handling the interplay between various run-time and design-time quality attributes;
- accelerate the transition from embedded software analysis and design to production;
- facilitate the production of secure embedded software products with an emphasis on reliability and fully automated quantitative security assessment.

Acknowledgment

Work reported in this paper has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No. 780572 (project: SDK4ED).

References

[1] ISO/IEC, “ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models,” Tech. Rep., 2011.

[2] M. G. Siavvas, K. C. Chatzidimitriou, and A. L. Symeonidis, “Qatch - an adaptive framework for software product quality assessment,” *Expert Systems with Applications*, vol. 86, pp. 350 – 366, 2017.

[3] T. Amanatidis, N. Mittas, A. Chatzigeorgiou, A. Ampatzoglou, and L. Angelis, “The developer’s dilemma: factors affecting the decision to repay code debt,” in *Proceedings of the 2018 International Conference on Technical Debt - TechDebt ’18*. Gothenburg, Sweden: ACM Press, 2018, pp. 62–66.

[4] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, “The financial aspect of managing technical debt: A systematic literature review,” *Information and Software Technology*, vol. 64, pp. 52–73, Aug. 2015.

[5] M. Riaz, E. Mendes, and E. Tempero, “A Systematic Review of Software Maintainability Prediction and Metrics,” in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 367–377.

[6] A. Ampatzoglou, S. Bibi, A. Chatzigeorgiou, P. Avgeriou, and I. Stamelos, “Reusability Index: A Measure for Assessing Software Assets Reusability,” in *New Opportunities for Software Reuse*, ser. Lecture Notes in Computer Science, R. Capilla, B. Gallina, and C. Cetina, Eds. Springer International Publishing, 2018, pp. 43–58.

[7] A. Ampatzoglou, A. Michailidis, C. Sarikyriakidis, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, "A Framework for Managing Interest in Technical Debt: An Industrial Validation," in *Proceedings of the 2018 International Conference on Technical Debt*, ser. TechDebt '18. New York, NY, USA: ACM, 2018, pp. 115–124, event-place: Gothenburg, Sweden.

[8] D. Tsoukalas, M. Siavvas, M. Jankovic, D. Kehagias, A. Chatzigeorgiou, and D. Tzovaras, "Methods and tools for td estimation and forecasting: A state-of-the-art survey," in *2018 International Conference on Intelligent Systems (IS)*. IEEE, 2018, pp. 698–705.

[9] I. Sommerville, *Software Engineering*, 9th ed. USA: Addison-Wesley Publishing Company, 2010.

[10] G. McGraw, *Software Security: Building Security In*. Addison-Wesley Professional, 2006.

[11] B. Chess and B. Arkin, "Software Security in Practice," *IEEE Security Privacy*, vol. 9, no. 2, pp. 89–92, Mar. 2011.

[12] M. Siavvas, E. Gelenbe, D. Kehagias, and D. Tzovaras, "Static Analysis-Based Approaches for Secure Software Development," in *Security in Computer and Information Sciences*, ser. Communications in Computer and Information Science, E. Gelenbe, P. Campegiani, T. Czachórski, S. K. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, Eds. Springer International Publishing, 2018, pp. 142–157.

[13] P. K. Manadhata and J. M. Wing, "An Attack Surface Metric," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371–386, May 2011.

[14] Q. Xia, W. Liang, Z. Xu, and B. Zhou, "Online Algorithms for Location-Aware Task Offloading in Two-Tiered Mobile Cloud Environments," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, Dec. 2014, pp. 109–116.

[15] F. Samie, L. Bauer, and J. Henkel, "IoT technologies for embedded computing: A survey," in *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct. 2016, pp. 1–10.

[16] L. Papadopoulos, C. Marantos, G. Digkas, A. Ampatzoglou, A. Chatzigeorgiou, and D. Soudris, "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 - SCOPES '18*. Sankt Goar, Germany: ACM Press, 2018, pp. 62–65.

[17] M. Siavvas, D. Tsoukalas, M. Jankovic, D. Kehagias, A. Chatzigeorgiou, D. Tzovaras, N. Anicic, and E. Gelenbe, "An Empirical Evaluation of the Relationship between Technical Debt and Software Security," *9th International Conference on Information Society and Technology*, 2019.

[18] S. Cho and R. G. Melhem, "On the interplay of parallelization, program performance, and energy consumption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 3, pp. 342–353, 2010.

[19] Kihwan Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 18–28, Jan 2005.

[20] M. F. Oliveira, R. M. Redin, L. Carro, L. da Cunha Lamb, and F. R. Wagner, "Software quality metrics and their impact on embedded software," in *2008 5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software*. IEEE, 2008, pp. 68–77.

[21] R. Verdecchia, R. A. Saez, G. Procaccianti, and P. Lago, "Empirical evaluation of the energy impact of refactoring code smells." in *ICT4S*, 2018, pp. 365–383.

[22] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*, 1st ed. Hoboken, N.J: Wiley, Apr. 2012.

[23] P. Brereton, B. Kitchenham, D. Budgen, and Z. Li, "Using a Protocol Template for Case Study Planning," *Evaluation and Assessment in Software Engineering*, p. 8, 2008.

[24] J. Kontio, J. Bragge, and L. Lehtola, "The Focus Group Method as an Empirical Tool in Software Engineering," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. London: Springer London, 2008, pp. 93–116.

[25] D. McDonagh-Philp, M. Mdrs, and A. Bruseberg, "Using Focus Groups to Support New Product Development," *Institution of Engineering Designers Journal*, p. 6, 2000.

[26] B. Glaser and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. New Brunswick: Routledge, Jan. 2000.

[27] H. Boeije, "A Purposeful Approach to the Constant Comparative Method in the Analysis of Qualitative Interviews," *Quality and Quantity*, vol. 36, no. 4, pp. 391–409, Nov. 2002.

[28] "IEEE 29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering." [Online]. Available: https://standards.ieee.org/standard/29148-2011.html

[29] D. Clegg and R. Barker, *Case Method Fast-Track: A Rad Approach*, 1st ed. Wokingham, England ; Reading, Mass. : Berkshire, UK ; Redwood Shores, CA, USA: Addison-Wesley, Sep. 1994.

[30] "IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems." [Online]. Available: https://standards.ieee.org/standard/1471-2000.html

[31] P. Kruchten, "The 4+1 View Model of architecture," *IEEE Software*, vol. 12, no. 6, pp. 42–50, Nov. 1995.

[32] N. Rozanski and E. Woods, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*, 1st ed. Upper Saddle River, NJ: Addison-Wesley Professional, Apr. 2005.

[33] D. Avison and G. Fitzgerald, *Information Systems Development: Methodologies, Techniques and Tools*, 4th ed. London: McGraw-Hill Education / Europe, Middle East & Africa, Mar. 2006.

[34] T. Hathaway and A. Hathaway, *Data Flow Diagrams - Simply Put!: Process Modeling Techniques for Requirements Elicitation and Workflow Analysis*. BA-EXPERTS, Mar. 2015.

[35] A. Aguiar, A. Sousa, and A. Pinto, "Use-Case Controller," in *EuroPLoP*, 2001.

[36] E. Gamma, R. Helm, R. Johnson, J. Vlissides, and G. Booch, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Reading, Mass: Addison-Wesley Professional, Nov. 1994.

[37] E. Wolff, *Microservices: Flexible Software Architecture*, 1st ed. Boston: Addison-Wesley Professional, Oct. 2016.

[38] M. Richards and O. Media, "Software Architecture Patterns," p. 55.

[39] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, "Microservice Architecture: Aligning Principles, Practices, and Culture," p. 146.

[40] M. Siavvas, "Static Analysis for Facilitating Secure and Reliable Software," Ph.D. dissertation, Department of Electrical and Electronic Engineering, Imperial College London, 2019.

[41] M. Siavvas, D. Kehagias, D. Tzovaras, and E. Gelenbe, "A hierarchical model for quantifying software security based on static analysis alerts and software metrics," *Journal of Systems and Software*, 2019, (under review).

[42] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PloS one*, vol. 13, no. 3, p. e0194889, 2018.