DECODER - DEveloper COmpanion for Documented and annotatEd code Reference*

Victoria Torres[0000-0002-2039-2174] and Miriam Gil[0000-0002-2987-1825] and Vicente Pelechano[0000-0003-1090-230X]

Universitat Politècnica de València, València 46022, Spain
{vtorres, mgil, pele}@pros.upv.es

Abstract. Software is everywhere and the productivity of Software Engineers has increased radically with the advent of new specifications, design and programming paradigms and languages. The main objective of the DECODER project is to introduce radical solutions to increase productivity by increasing the abstraction level, at specification stage, using requirements engineering techniques to integrate more complete specifications into the development process, and formal methods to reduce the time and efforts for integration testing. DECODER project will develop a methodology and tools to improve the productivity of the software development process for medium-criticality applications in the domains of IoT, Cloud Computing, and Operating Systems by combining Natural Language Processing techniques, modelling techniques and Formal Methods. A radical improvement is expected from the management and transformation of informal data into material (herein called "knowledge") that can be assimilated by any party involved in a development process. The project expects an average benefit of 20% in terms of efforts on several use cases belonging to the beforehand mentioned domains and will provide recommendations on how to generalize the approach to other medium-critical domains.

Keywords: Requirements Analysis, Open Source Software, Software engineering, operating systems, computer languages.

1 Project summary

The DEveloper COmpanion for Documented and annotatEd code Reference (DECODER) project is a H2020 project (H2020-ICT-16-2018 Software Technologies call) that has received funding from the European Union's H2020 research and innovation program under the grant agreement 824231. The project has a duration of 36 months, starting in January 2019 and finishing in December 2021. Currently, the project has already reached the first six months, period in which all work packages have started, and some deliverables have also been submitted. Updated information about the project

* This work has been developed with the financial support of the European Union's Horizon 2020 research and innovation programme under grant agreement No. 824231 and the Spanish State Research Agency under the project TIN2017-84094-R and co-financed with ERDF

can be found in the https://www.decoder-project.eu web site. Regarding the project consortium, it is formed by contributors from seven partners from four different European countries (cf. Table 1).

Table 1. List of participating partners and key positions

Partner	Short name	Country	Key positions
Technikon	TEC	Austria	Project Leader and WP8 Leader
CEA Tech	CEA	France	WP1, WP3 Leader
Tree Technology SA	TREE	Spain	WP2 Leader
Capgemini España SL	CAPGEMINI	Spain	WP4 Leader
Universitat Politècnica de València	UPV (PROS)	Spain	WP5 Leader
Sysgo AG	SYSGO	Germany	WP6 Leader
OW2	OW2	France	WP7 Leader

2 Project motivation

Software drives our modern economy; it is indeed present everywhere, from critical infrastructures supporting our societies, such as energy supply and transportation, down to the smart devices connecting us to the internet (also called IoT). However, too much time is wasted during software development projects due to wrong decisions taken along the whole process. The main reason for taking such decisions is the amount of information stakeholders have to deal with and the lack of proper documentation. To this end, software production is insufficiently supported by effective tools and often, engineers lack a systematic approach for the development and safe reuse of components and their associated knowledge. In addition, a typical development process requires interactions of many stakeholders, at very different abstraction levels, and often over ambiguous and incomplete documents. This makes the integration and even more the maintenance of software systems extremely difficult and costly.

Within this context, support to properly handle project knowledge derived from all the involved artefacts (e.g., source code, specifications, informal documents, etc.) is required; it is software project intelligence that assist developers with an instantaneous access to its documentation, abstract models, verification data and traceability matrix.

3 Detailed description of the goals of the project

The main goal of DECODER is to build a smart environment that could assist and help developers, analysts, testers, etc. to improve the software development process. To achieve this goal, in DECODER we propose to apply and combine techniques from Natural Language Processing, Machine Learning, Process Modeling, Model Transformations, and Verification. Specifically, the detailed objectives are the following:

- Objective 1: High-level abstract models for engineers.
- Objective 2: Significantly increase the software development and maintenance efficiency.
- Objective 3: Drastically improving the use of informal knowledge and artefacts.
- Objective 4: Build collaborative knowledge and smart user interfaces.
- Objective 5: Improve the overall quality of software for medium-criticality domains.

To achieve the aforementioned goals, the project has been designed according to eight work packages as shown in Fig. 1. While the work developed in work packages 1 to 6 are focused mainly on the design and implementation of the technological innovation foreseen in DECODER, in work packages 7 and 8, common activities in such European projects such as dissemination and management activities will be performed.

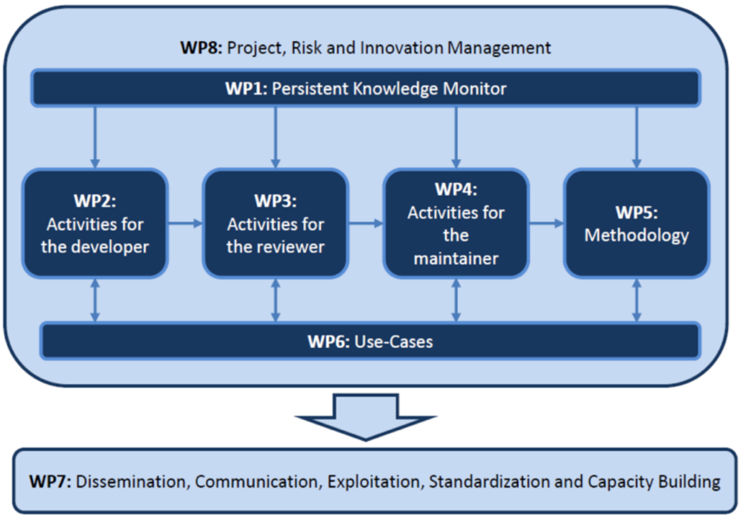

Fig. 1 DECODER Work package distribution and the relationships between them

4 Project expected achievements and outcome

The DECODER project addresses objective ICT-16-2018 scope a) and both areas thereof (i.e. code and resources abstraction, and advanced software systems development). The project will support this objective by obtaining the following achievements:

- Improve the productivity of software engineers of medium-criticality applications along the whole lifecycle process by several means: 1) increasing the abstraction level, namely at specification stage, 2) using requirements engineering techniques to integrate more complete specifications into the development process, and 3) using Formal Methods to reduce the time and efforts for integration testing, replacing it by formal analyses. This achievement is related to objectives 1 and 3.

- Development of novel languages defined from the abstraction of the formalisms used today for requirements analysis and specification: 1) an abstract formal design language, namely ASFM, to navigate between different levels of abstractions, and 2) an abstract graphical specification language, namely GSL, capable of intuitively specifying some code and generating detailed specifications in ACSL/ACSL++ and JML. This achievement is related to objective 1.
- Development of new languages and methods to formalize software requirements that are often informal based on NLP techniques to formalize in a human understandable formalism the informal requirements. These new languages will permit to describe data and processes amenable for specification and refinements. This achievement is related to objectives 1, 3 and 4.
- Demonstrate the applicability and viability of the proposed solution on several use cases from very different categories: 1) IoT/embedded systems, 2) Artificial Intelligence and IoT domains (computer vision), 3) enterprise computing (including Cloud computing/Big Data and Middleware/Cloud computing). This achievement is related to objectives 2 and 5.

Regarding the outcomes, one of the major outcomes of DECODER is the Persistent Knowledge Monitor (PKM) that will be developed in WP1. This PKM will provide a "central" infrastructure to store, access, and trace all the persistent data, information and knowledge related to a given software or ecosystem (notably its source code and related artefacts, and also derived information). As Fig. 1 shows, the PKM will be used in work packages 2, 3, and 4, where activities for developers, reviewers, and maintainers over the PKM are defined. Next, we detail the major outcomes of these work packages.

Regarding WP2 where activities for developers are defined, support for the transformation of informal code related data (e.g., text that captures requirements, informal specifications, internal documentation or even comments in the source code) into formal documentation and also summarize source code will be provided. The generation of formal documentation provides useful information for users who have created that piece of code and have to return to it at some point as well as future maintainers.

Regarding WP3 where activities for reviewers are defined, support for saving in the knowledge database is provided. In particular it will be stored what an external reviewer understands from the code, the comments and how the code intentions are automatically verified. This expert knowledge is usually lost after the code review and the advanced users need to recreate it repeatedly. In this WP, we formalize the results of the review activities into an Abstract Semi-Formal Model. Ideally, such a model would only contain formal properties of the code written in ACSL/ACSL++ and JML and automatically verified by formal deductive verification. However, the definition of such a model requires far too much resources and expertise to build it from scratch. To lower this expertise, we accept definitions coming from different sources: formal description, function calls without any side effect, sequence diagram, formal visualization and abstraction. The ASFM language will contain the functional logic notions of data structure invariant, type states, behaviors based on pre/post-conditions. If the reviewer can write them manually in the ACSL/ACSL++ and JML annotation languages, WP3 proposes many ways to incrementally build and enrich such Abstract Models between the code and the logic specification.

Regarding WP4 where activities for the maintainer are defined, support for controlling the impact of changes through implementation of traceability management will be provided. This means establishing links and maintaining cross-references between artefacts. For this purpose, NLP technologies will be used to build and manage a traceability matrix between requirements, code and documentation. This matrix is a sparse matrix of traces. A trace is made of two anchors (or trace location) with additional semantic attribute (role, level of confidence) and some more technical or management attributes. The anchor references an element in the PKM. The traceability matrix binds high-level requirements with fine-grain specifications of code to help controlling the impact of changes. The traceability matrix binds a piece of code with a piece of documentation to help experienced people to make explicit all implicit knowledge that exists in their mind and will help new staff members to quickly grasp the big picture and the crucial details before doing any change. Traceability management is extended to check consistency with test cases.

Besides these four work packages, in work package 5 methodology support for end-users (C, C++, and Java programmers) along the life cycle will be provided. To ensure that certain software properties are satisfied when applying the proposed methodology, an innovative methodology will be defined based on formal and agile techniques. As a result, the proposed methodology will define the different stages of the complete life cycle development, the different roles involved as well as the intermediate artefacts built, modified or just consumed in the different proposed stages. In addition, the supporting tool will integrate the set of tools proposed in WP1 and WP2 as well as the artefacts consumed and produced by these tools along the life cycle. As a result, the methodology will ensure the generation of better documentation, the construction of critical and medium critical applications ensuring the quality of the obtained artefact as well as the application of the existing standards.

Finally, the framework and tools developed in the previous work packages will be put into practice with real source code, specifically on large use cases provided by partners that are not themselves tool developers but applications developers. This will be performed within the context of WP6, where besides demonstration purposes, feedback on the quality of the maintenance activities in the form of measurements (productivity gains) and recommendations will be also generated. These experimental activities can be considered as a first step towards the later exploitation of the project's tools and framework. In addition to these six work packages DECODER defines two more packages, WP7 and WP8 which are intended mainly for dissemination and management purposes respectively.

5 Existing collaborations with other projects

At the current state, DECODER project maintains collaborations with several projects:

- Project VESSEDIA[1] (Verification Engineering of Safety and Security Critical Dynamic Industrial Applications): Our project will reuse the tools developed by VESSEDIA to develop modular specifications and proofs to render formal specification activities easier to manage.
- Project OpenReq[2] (Intelligent Recommendation & Decision Technologies for Community-Driven Requirements Engineering): Our project will get inspiration from the original requirements specification novelties to enforce its NLP activities and define better languages for expressing semi-formalized requirements. We will assist to the OpenReq week in September 2019 where DECODER project will be presented.

6 Interest in participating in the EU Project Space at PROFES

DECODER has different reasons to participate in PROFES. First, we would like to announce the project to the PROFES community, a community mainly focused on the software process improvement where DECODER is also putting all its efforts. Secondly, we would like to present all PROFES participants the major research outcomes achieved during the first six months of the project and discuss any potential improvement to them. Thirdly, we would like to learn from the PROFES community the last advances and research developed in this field in order to improve DECODER execution. Finally, we also want to attract DECODER early adopters from practitioners, researchers, and educators interested in the software process improvement. Besides PROFES, in DECODER we have a dissemination plan that has also resulted in the participation of different events during the first six months of execution (see Table 2). However, it is important to continue this task both in industry and in academic contexts. For this reason, the different partners from the consortium are actively working to participate in different types of events, not just to announce the DECODER project but also to discuss the research outcomes with the scientific and industrial communities interested in improving the software development process. In particular, DECODER contributors are working to participate in the next months in the events listed in Table 2.

Table 2. Past and upcoming events where DECODER has participated or will participate

Event	Country	Dates
Testnet Spring event	Nieuwegein, Netherlands	11 May 2019
OW2con’19	Paris, France	12-13 June 2019
OpenReq week	Hamburg, Germany	2-6 September 2019
EclipseCon Europe	Ludwigsburg, Germany	21-24 October 2019
DeVoxx	Antwerp, Belgium	4-8 November 2019
Paris Open Source Summit	Paris, France	10-11 December 2019

[1] https://vessedia.eu/
[2] https://openreq.eu/