Description of Work

Szilvási, Krisztián
krisztian.szilvasi.3@gmail.com

Fenyvesi, Róbert
fenyvesr@gmail.com

November 23, 2019

SEVENTH FRAMEWORK PROGRAMME

ICTs

INFORMATION AND COMMUNICATION TECHNOLOGIES

Grant agreement for: Large-scale integrating project

Annex I. - "Description of Work"

Project acronym: ACROSS

Project full title: Automatic Code Verification by Formal Analysis

Grant agreement no.: FP7-199502

Date of preparation of Annex I (latest version): 2019.11.23

Date of approval of Annex I by Commission:

Contents

1 The project summary

Product Development is driven by stakeholder requirements. The larger the developed system, the harder it is to analyze and verify it. Software Projects are no exceptions. This project aims to show how the verification of huge software projects can be performed automatically against the given requirements. The project spreads across multiple areas of main stream research.

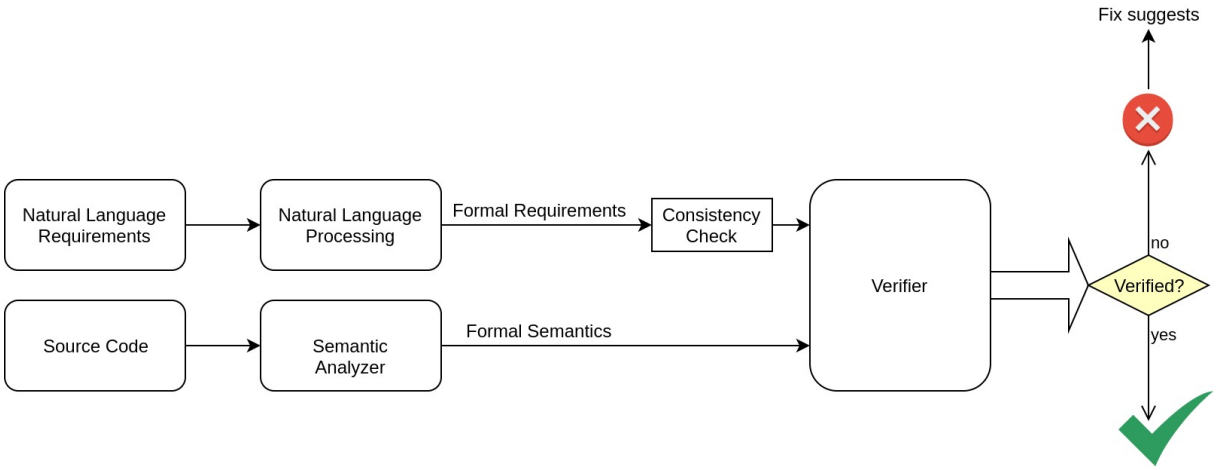

Figure 1: Project Architecture

Natural Language Processing (NLP) is used for formalizing the Software Requirements Specification (SRS). Since, natural language is widely understood by stakeholders, it is used as a common way for representing requirements. Representing requirements in natural language suffers from potential problems like ambiguity, inconsistency and incompleteness. A systematic literature review in the last two decades from 1995 till 2016 shows that collecting ambiguous requirements is one of the highest critical challenges in software engineering [2]. Since the advent of software engineering, researchers used formal and semi-formal methods to overcome this problem. However, even when formal and semi-formal languages are used, there is no escape from natural language as the initial requirements are written in natural language [6]. The consequences of ambiguous requirements will lead to excessive efforts, high cost and failure in some software projects. For example, software developers might decide a subjective interpretation of requirements based on their point of view. Ferrari et al. (2014) argued that this subjective interpretation leads to designing software in a different way from what was intended in the requirements [5]. For several decades, SRS processing and analysis has been the focus of research in software engineering discipline. Since natural language is ambiguous, a computer cannot provide full support to analyze SRS in an automatic fashion. Consequently, the analysis of SRS is conducted manually which consumes time, effort and cost. Most importantly, the manual analysis of requirements results in inefficiency and imprecise results [10]. The problem will be more obvious and critical when software projects involve thousands of requirements and hundreds of SRS documents. Conducting verification of thousands of requirements via humans will become extremely expensive [4]. Generally, the primary source of problems in requirement engineering is reliance on humans extensively [1]. This discussion leads to the importance of finding an automatic way for processing SRS. NLP was used as a possible solution to resolve ambiguity and to provide valuable information to the intended software developers. Ryan (1993) argued that: "It is highly questionable that the resulting system from NLP would be of great use in requirements engineering" [9]. Nazir et al. (2017) conducted a systematic literature review on NLP applications for software requirement engineering, he concluded that: "Manual operations are still required on initial plain text of software requirements before applying the desired NLP techniques" [7].

On the other hand the formal semantics of the source code is needed. A formal semantics should serve as a solid foundation for any programming language development, so it must be correct and complete (to be trusted and useful), executable (to yield a reference implementation), and appropriate for program reasoning and verification.

The five most popular programming languages according to GitHub in 2019 is JavaScript, Python, Java, PHP and C#. Several efforts to give JavaScript a formal semantics have been made, most notably by Politz et al. [8] and Bodin et al. [3]. But Unfortunately, these address fragments of the language and are not fully validated with a formal JavaScript semantics. Having to define two or more different semantics for a real-life language, together with proofs of equivalence, is a huge burden in itself, not to mention that these all need to be maintained as the language evolves. Due to the functional nature of their interpreters, these semantics cannot handle the nondeterminism of JavaScript well. Finally, their interpreters are not suited for symbolic execution, and thus for developing program reasoning tools.

References

[1] Usman Ahmed. A review on knowledge management in requirements engineering. pages 1–5, 02 2018.

[2] Souhaib Besrour, Lukman Rahim, and P. Dominic. A quantitative study to identify critical requirement engineering challenges in the context of small and medium software enterprise. pages 606–610, 08 2016.

[3] Martin Bodin, Arthur Charguéraud, Daniele Filaretti, Philippa Gardner, Sergio Maffeis, Daiva Naudziuniene, Alan Schmitt, and Gareth Smith. A trusted mechanised javascript specification. volume 49, pages 87–100, 01 2014.

[4] Gauthier Fanmuy, Anabel Fraga, and Juan Llorens. Requirements verification in the industry. pages 145–160, 01 2011.

[5] Alessio Ferrari, Giuseppe Lipari, Stefania Gnesi, and Giorgio Spagnolo. Pragmatic ambiguity detection in natural language requirements. 08 2014.

[6] Erik Kamsties. *Engineering and Managing Software Requirements*, pages 245–266. 01 2005.

[7] Farhana Nazir, Wasi Haider, Muhammad Anwar, and Muazzam Khattak. The applications of natural language processing (nlp) for software requirement engineering - a systematic literature review. pages 485–493, 03 2017.

[8] Joe Gibbs Politz, Spiridon Aristides Eliopoulos, Arjun Guha, and Shriram Krishnamurthi. Adsafety: Type-based verification of javascript sandboxing. *CoRR*, abs/1506.07813, 2015.

[9] K. Ryan. The role of natural language in requirements engineering. In *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 240–242, Jan 1993.

[10] Yinglin Wang. Automatic semantic analysis of software requirements through machine learning and ontology approach. *Journal of Shanghai Jiaotong University (Science)*, 21:692–701, 12 2016.

2 List of Beneficiaries

Beneficiary Number	Beneficiary name	Beneficiary short name	Beneficiary type	Country	Date enter project	Date exit project
1 (coordinator)	Eötvös Loránd University	ELTE	RTD Performer	Hungary	2019.12.01	2022.12.01
2	Aalto University	Aalto	RTD Performer	Finland	2019.12.01	2022.12.01
3	Royal Institute of Technology	KTH	RTD Performer	Sweden	2019.12.01	2022.12.01
4	Technical University Berlin	TUB	RTD Performer	Germany	2019.12.01	2022.12.01
5	Université Côte d'Azur	UCA	RTD Performer	France	2019.12.01	2022.12.01
6	University of Trento	UNITN	RTD Performer	Italy	2019.12.01	2022.12.01
7	Elte-Soft Nonprofit Kft.		SME Association	Hungary	2019.12.01	2022.12.01
8	Polarion Software		SME Association	Germany	2019.12.01	2022.12.01
9	Rational Software		SME Association	United States	2019.12.01	2022.12.01

Table 1: Table of Beneficiaries

3 The budget brakedown[1]

Participant number in this project	Organisation short name	Type	Funding %	Indirect costs	RTD / Innovation (A) costs	Demonstration (B) costs	Management (C) costs	Other (D) costs	Total (A+B+C+D)	Total receipts	Requested EU contribution
1	ELTE	RTD Performer	71.43%	50.000	80.000	5.000	50.000	25.000	160.000	210.000	150.000
2	Aalto	RTD Performer	63.89%	100.000	170.000	6.000	26.000	11.000	213.000	313.000	200.000
3	KTH	RTD Performer	55.55%	110.000	200.000	13.000	30.000	7.000	250.000	360.000	200.000
4	TUB	RTD Performer	55.55%	100.000	220.000	16.000	33.000	27.000	296.000	396.000	220.000
5	UCA	RTD Performer	62.5%	80.000	120.000	12.000	20.000	8.000	160.000	240.000	150.000
6	UNITN	RTD Performer	67.19%	75.000	120.000	20.000	25.000	13.000	178.000	253.000	170.000
7	Elte-Soft	SME Association	37.03%	50.000	70.000	5.000	10.000	-	85.000	135.000	50.000
8	Polarion Software	SME Association	0.00%	90.000	20.000	2.000	10.000	-	32.000	122.000	-
9	Rational Software	SME Association	0.00%	25.000	5.000	10.000	15.000	4.000	34.000	59.000	-
Total			54.6%	680.000	1.005.000	89.000	219.000	95.000	1.408.000	**2.088.000**	**1.140.000**

Table 2: Table of Costs

[1] All costs are in EUR

4 List of Work Packages

WP Number	WP Title	Type of activity	Lead beneficiary number	Person-month	Start month	End month
WP1	Project Management	MGT	1	18	1	36
WP2	Module Definitions	RTD	1	15	1	3
WP3	NLP for Requirements Analysis	RTD	4	36	4	13
WP4	Semantic Analyzer	RTD	3	48	4	16
WP5	Consistency Check	RTD	5	30	12	18
WP6	Verifier	RTD	2	120	18	30
WP7	Code Fix Suggestion	RTD	6	36	30	36
WP8	Lessons Learned	OTHER	1	4	1	36
WP9	Demonstration	DEM	1	4	1	36
Total:				311		

Table 3: Table of Work Packages

5 List of Deliverables

Del. no.	Deliverable Title	WP no.	Nature	Dissemination level	Delivery date
D1.1	First Annual Report	1	R	PU	12
D1.2	Second Annual Report	1	R	PU	24
D1.3	Final Project Report	1	R	PU	36
D2.1	Module Specification	2	O	CO	3
D3.1	First NLP Prototype	3	P	C0	7
D3.2	Final NLP Prototype	3	P	C0	13
D3.3	NLP Project Paper	3	O	PU	13
D4.1	First Semantic Analyzer Prototype	4	P	C0	8
D4.2	Final Semantic Analyzer Prototype	4	P	C0	16
D4.3	Semantic Analyzer Project Paper	4	O	PU	16
D5.1	Consistency Check Prototype	5	P	C0	18
D6.1	First Verifier Prototype	6	P	C0	24
D6.2	Final Verifier Prototype	6	P	C0	30
D6.3	Verifier Project Paper	6	O	PU	30
D7.1	Code Fix Suggester Prototype	7	P	C0	36
D8.1	NLP Lessons Learned Report	8	P	PP	14
D8.2	Semantic Analyzer Lessons Learned Report	8	P	PP	17
D8.3	Verifier Lessons Learned Report	8	P	PP	31
D8.4	Final Lessons Learned Report	8	R	PP	36
D9.1	First Annual Demonstration	9	D	PU	12
D9.2	Participation in FM 2021 Conference	9	D	PU	16
D9.3	Second Annual Demonstration	9	R	D	24
D9.4	Final Project Demonstration	9	R	D	36

Table 4: Table of Deliverables

6 Work Package Descriptions

6.1 Work Package 1

Work package number	WP1	Start date or starting event:				1
Work package title	Project Management					
Activity Type	MGT					
Participant number	1	7	8	9			
Person-months per participant:							

6.1.1 Objectives

6.1.2 Description of work

6.1.3 Deliverables

- D1.1 First Annual Report
- D1.2 Second Annual Report
- D1.3 Final Project Report

7