Description of Work

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

Fenyvesi, Róbert
fenyvesr@gmail.com

November 26, 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.
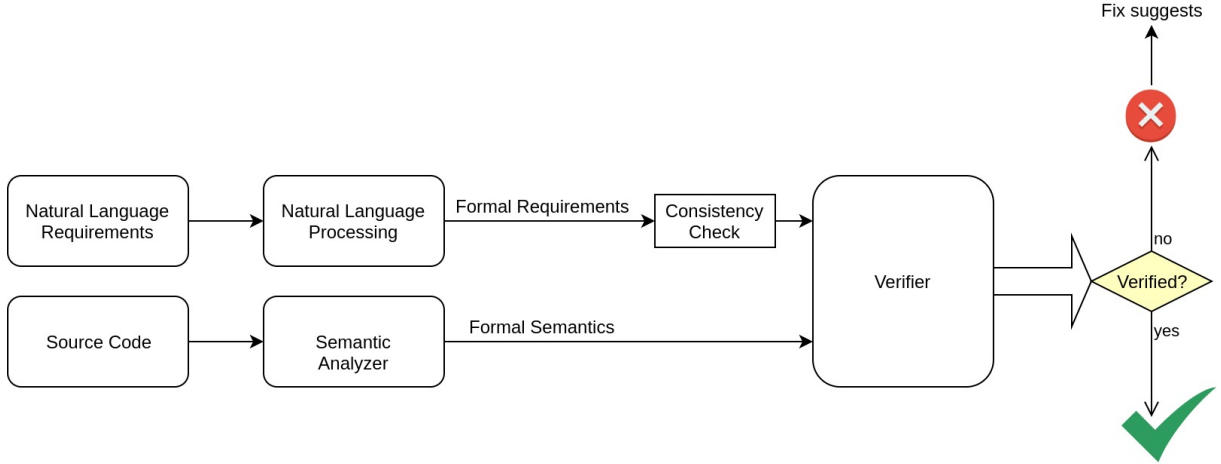

Figure 1: Project Architecture

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 Chargueraud, 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	Dissemi- nation level	Delivery date
D1.1	First Annual Report	1	R	PU	13
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	CO	7
D3.2	Final NLP Prototype	3	P	CO	13
D3.3	NLP Project Paper	3	O	PU	13
D4.1	First Semantic Analyzer Prototype	4	P	CO	8
D4.2	Final Semantic Analyzer Prototype	4	P	CO	16
D4.3	Semantic Analyzer Project Paper	4	O	PU	16
D5.1	Consistency Check Prototype	5	P	CO	18
D6.1	First Verifier Prototype	6	P	CO	24
D6.2	Final Verifier Prototype	6	P	CO	30
D6.3	Verifier Project Paper	6	O	PU	30
D7.1	Code Fix Suggester Prototype	7	P	CO	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:	9	3	3	3		

6.1.1 Objectives

- O1.1 Meet Project Milestones: The management aims for keeping the project on track and meet the milestones. To achieve this, state of the art project management tools will be used such as Rational Team Concert (RTC). The Universities will be overseen by ELTE while the companies manage themselves.

- O1.2 Manage Necessary Deliverables: The management aims for reporting annually a clear view over the project for supervisory authorities.

6.1.2 Description of work

- O1.1 Meet the Project Milestones

  - T1.1.1: Bi-weekly management meetings should be held between the project participants and the meeting minutes should be stored at a central location accessible by all stakeholders.
  - T1.1.2: Up-to-date reports should be available and presented at the stakeholder meetings.

- O1.2 Manage Necessary Deliverables:

  - T1.2.1: The management should create and present the First Annual Report to the supervisory authorities.
  - T1.2.2: The management should create and present the Second Annual Report to the supervisory authorities.
  - T1.2.3: The management should create and present the Final Annual Report to the supervisory authorities.

6.1.3 Deliverables

- D1.1 First Annual Report: It should contain the work done during the first year and the achievements while introducing the time plan of next year. The report should be easily understandable for the authorities and should cover all fields of work.

- D1.2 Second Annual Report: It should contain the work done during the second year and the achievements while introducing the time plan of next year. The report should be easily understandable for the authorities and should cover all fields of work.

- D1.3 Final Project Report: It should contain the work done during the whole project life cycle and the achievements. The report should be easily understandable for the authorities and should cover all fields of work.

6.2 Work Package 2

Work package number	WP2	Start date or starting event:			1	
Work package title	Module definitions					
Activity Type	RTD					
Participant number	1	5	6	7		
Person-months per participant:	5	3	3	4		

6.2.1 Objectives

- O2.1 Define module interfaces: Module's and each submodules' both entry and exit interfaces shall be defined and agreed with contributors

6.2.2 Description of work

- O2.1 Define module interfaces

  - T2.1.1: Entry and exit interfaces for whole module shall be defined and agreed by contributors and stakeholders
  - T2.1.2: Entry and exit interfaces for NLP submodule shall be defined and agreed by contributors and stakeholders
  - T2.1.3: Entry and exit interfaces for Semantic Analyzer submodule shall be defined and agreed by contributors and stakeholders
  - T2.1.4: Entry and exit interfaces for Consistency Check shall be defined
  - T2.1.5: Entry and exit interfaces for Verifier submodule shall be defined and agreed by contributors and stakeholders

6.2.3 Deliverables

- D2.1 Module Specification: It should contain detailed specification of module's and each submodules' interfaces. Specifications must be reviewed and agreed by contributors.

6.3 Work Package 3

Work package number	WP3	Start date or starting event:			4		
Work package title	NLP for Requirements Analysis						
Activity Type	RTD						
Participant number	4	7	8	9			
Person-months per participant:	20	3	5	8			

6.3.1 Objectives

- O3.1 Create the NLP: The NLP should be able to transform the natural language requirements into a formalized description of the requirements. The architecture should match the problem at hand. It has to be fast and easily integrable into multiple Requirement Engineering Tools such as DOORS or Polarion.

  The NLP should be trained on historical data and on small batches to learn basic relationships. The NLP should be able to reproduce the same or better formal definition as the ones it was trained on.

  The NLP should be validated on the companies real life historical project requirements. The results should be reported and sent back for perfection.

- O3.2 Present Research Work: The research results should be presented and summarized at the end of the work.

6.3.2 Description of work

- O3.1 Create the NLP:

  - T3.1.1: The Technical University Berlin (TUB) team should define the architecture for the NLP. The architecture should be adequate for the task at hand.
  - T3.1.2: The architecture has to be documented and presented to the stakeholders until M1.
  - T3.1.3: The TUB team should train the NLP for the task at hand. All training data should be available for all stakeholders and the training data should be accepted by all stakeholders. The training data can be changed during the project, but it has to be always accepted by all stakeholders.
  - T3.1.4: The companies have to validate the trained NLP on their historical real life project requirements. The companies has to be involved as soon as as possible and the validation should be repeated as often as possible. The results should be documented at a central location accessible by all stakeholders. The results should be taken into consideration for any change in the architecture or in the training data.

- O3.2 Present Research Work

  - T3.2.1: The work should be summarized and be published as an article until M1.

6.3.3 Deliverables

- D3.1 First NLP Prototype : The first prototype should be delivered at the end of the 7th months. It has to be able to run without crash and fulfill basic functionalities.

- D3.2 Final NLP Prototype : The final prototype should be delivered until M1 and it has to fulfill all functionalities.

- D3.3 NLP Project Paper : The project paper should be delivered until M1 and should contain the results of WP3.

6.4 Work Package 4

Work package number	WP4	Start date or starting event:			4		
Work package title	Semantic Analyzer						
Activity Type	RTD						
Participant number	3	2	5	6			
Person-months per participant:	24	12	6	6			

6.4.1 Objectives

- O4.1 Create the Semantic Analyzer: The Semantic Analyzer should be able to translate source code into its formal equivalence. Translated text should be constructed in the way that it can be compared to formal requirements.

  The Semantic Analyzer have well defined interfaced so it can be built into popular and requested integrated development environment.

- O4.2 Present Research Work: The work should be summarized and be published as an article until M2.

6.4.2 Description of work

- O4.1 Create the Semantic Analyzer

  - T4.1.1: The Royal Institute of Technology (KTH) team should define the architecture for the Semantic Analyzer. The architecture should be adequate for the task at hand.
  - T4.1.2: The architecture has to be documented and presented to the stakeholders until M2.
  - T4.1.3: The translated formal description should be analyzed and tested. Detailed report of both source code and its translated equivalence should be created.
  - T4.1.4: The Semantic Analyzer should be validated by independent comity as a prove of its correctness.

- O4.2 Present Research Work

  - T4.2.1: The work should be summarized and be published as an article until M2.

6.4.3 Deliverables

- D4.1 First Semantic Analyzer Prototype : The first prototype should be delivered at the end of the 8th months. It has to be able to run without crash and fulfill basic functionalities.

- D4.2 Final Semantic Analyzer Prototype : The final prototype should be delivered until M2 and it has to fulfill all functionalities.

- D4.3 Semantic Analyzer Project Paper : The project paper should be delivered until M2 and should contain the results of WP4.

6.5 Work Package 5

Work package number	WP5	Start date or starting event:			12	
Work package title	Consistency Check					
Activity Type	RTD					
Participant number	5	1	7			
Person-months per participant:	12	12	6			

6.5.1 Objectives

- O5.1 Check Formal Requirements: The Consistency Check feature should be able to decide whether the generated formal requirements are free of contradictions and ambiguities.

- O5.2 Feasibility: The Consistency Check feature should be able to decide whether the generated formal requirements are feasible so a solution which fulfills the formal requirements can be found.

6.5.2 Description of work

- O5.1 Check Formal Requirements:

  - T5.1.1: A generic approach should be applied to decide whether the formal requirements are free of contradictions. Formal proof should be found in case the requirements are contradictory. The Université Côte d'Azur (UCA) should create the formal environment for this feature.
  - T5.1.2: A generic approach should be applied to decide whether the formal requirements are free of ambiguities. Formal proof should be derived to pinpoint the ambiguities and maybe suggest a refinement to fix the ambiguity. The Eötvös Loránd University (ELTE) should create the formal environment for this feature.
  - T5.1.3: The ELTE-Soft should validate the feature based on their expertise and historical data.

- O5.2 Feasibility:

  - T5.2.1: The generated formal requirements should be checked whether a solution can be found which fulfills the formal requirements. A generic approach should be applied to show if no solution is feasible and formally prove this condition.
  - T5.2.2: The ELTE-Soft should validate the feature based on their expertise and historical data.

6.5.3 Deliverables

- D5.1 Consistency Check Prototype: The final prototype should be delivered at the end of the 18th months so until M3. It has to be able to run without any crash and fulfill all functionalities. The formal proof environments should be assessed by an independent expert committee.

6.6 Work Package 6

Work package number	WP6	Start date or starting event:			18		
Work package title	Verifier						
Activity Type	RTD						
Participant number	2	1	3	4	7	8	9
Person-months per participant:	30	25	25	25	5	5	5

6.6.1 Objectives

- O6.1 Create the Verifier: The Semantic Analyzer should be able to compere outputs from NLP and from Semantic Analyzer. Verifier's constructor shall be able to build the system and construct inputs in the way when same statements of requirement and code are compared.

  The Verifier should create a clear report of analyzed data, both in successful and unsuccessful results.

- O6.2 Present Research Work: The work should be summarized and be published as an article until M5.

6.6.2 Description of work

- O6.1 Create the Semantic Analyzer

  - T6.1.1: The Aalto University (Aalto) team should define the architecture for the Verifier. The architecture should be adequate for the task at hand.
  - T6.1.2: The architecture has to be documented and presented to the stakeholders until M5.
  - T6.1.3: The verified report should be analyzed and tested. Detailed report of compared objects and its result should be created.
  - T6.1.4: The Verifier should be validated by independent comity as a prove of its correctness
  - T6.1.5 Conditional output should be created.
  - T6.1.6 Interface to Code Fix Suggester should be defined in case if compared object are different and failed analysis

- O6.2 Present Research Work

  - T6.2.1: The work should be summarized and be published as an article until M5.

6.6.3 Deliverables

- D6.1 First Verifier Prototype : The first prototype should be delivered at the end of the 24th months. It has to be able to run without crash and fulfill basic functionalities.

- D6.2 Final Verifier Prototype : The final prototype should be delivered until M5 and it has to fulfill all functionalities.

- D6.3 Verifier Project Paper : The project paper should be delivered until M2 and should contain the results of WP6.

6.7 Work Package 7

Work package number	WP7	Start date or starting event:			30		
Work package title	Code Fix Suggestion						
Activity Type	RTD						
Participant number	6	5	1				
Person-months per participant:	15	10	5				

6.7.1 Objectives

- O7.1 Code Fix suggestion: In case of an invalid implementation if it is possible the final feature should suggest feasible fixes for the source code. In case of a huge difference between the formal requirements and the formal semantics of the source code the feature should not suggest any fix.

- O7.2 Multiple suggestions: In case of an invalid implementation if it is possible the final feature should suggest feasible fixes for the source code. The goodness of these suggestions should be measured with the difference between the formal requirements and the formal semantics of the source code.

6.7.2 Description of work

- O7.1 Code Fix suggestion:

  - T7.1.1: UCA should be responsible to implement the functionality which can suggest fixes for the incorrect implementation. These suggestion should be all feasible solutions for the generated formal requirements.
  - T7.1.2: The suggestions should be sorted according to their goodness factor. The suggestions should be able to be applicable with the least amount of effort from the user side.

- O7.2 Multiple suggestions:

  - T7.2.1: University of Trento (UNITN) should be responsible for evaluating the difference between the formal requirements and the formal semantics of the source code.
  - T7.2.2: ELTE should review the results and consult with UNITN. In case of an agreement the feature should perform correctly.

6.7.3 Deliverables

- D7.1 Code Fix Suggester Prototype: The final prototype should be delivered at the end of the 36th months so until M6. It has to be able to run without any crash and fulfill all functionalities. The final results of the systems should be heavily tested by the co-operative companies.

6.8 Work Package 8

Work package number	WP8	Start date or starting event:				1	
Work package title	Lessons Learned						
Activity Type	OTHER						
Participant number	1	2	3	4	5	6	
Person-months per participant:	2	0.4	0.4	0.4	0.4	0.4	

6.8.1 Objectives

- O8.1 Create Lessons Learned reports: WP responsibles should be sure that needed data and information for reports are collected. Lessons Learned should be part of Annual Reports and Demonstrations. Lessons Learned reports should be sent to Annual Reports and Demonstration WP responsibles.

- O8.2 Document mistakes and failures: Contributors should be reporting about their mistakes and failures. Data should be kept up-to-date and be in usable state.

6.8.2 Description of work

- O8.1 Create Lessons Learned reports

  - T8.1.1: Responsibles should create and present the NLP Lessons Learned Report to the supervisory authorities.
  - T8.1.2: Responsibles should create and present the Semantic Analyzer Lessons Learned Report to the supervisory authorities.
  - T8.1.3: Responsibles should create and present the Verifier Lessons Learned Report to the supervisory authorities.
  - T8.1.4: Responsibles should create and present the Final Lessons Learned Report to the supervisory authorities.

- O8.2 Keep collected data up-to-date

  - T8.2.1: Monthly data from contributors shall be collected
  - T8.2.2: Summarize and sent Lessons Learned reports to Annual Reports and Demonstration WP responsibles.
  - T8.2.3: Update and validate last collected before every milestone

6.8.3 Deliverables

- D8.1 NLP Lessons Learned Report: The Lessons Learned report done after NLP is finished should contain the mistakes and things what could have been done differently. The report should be easily understandable and be included into annual reports and demonstrations.

- D8.2 Semantic Analyzer Lessons Learned Report: The Lessons Learned report done after Semantic Analyzer is finished should contain the mistakes and things what could have been done differently. The report should be easily understandable and be included into annual reports and demonstrations.

- D8.3 Verifier Lessons Learned Report: The Lessons Learned report done after Verifier is finished should contain the mistakes and things what could have been done differently. The report should be easily understandable and be included into annual reports and demonstrations.

- D8.4 Final Lessons Learned Report: The Lessons Learned report done after the whole project is finished should contain the mistakes and things what could have been done differently. The report should be easily understandable and be included into final report and demonstrations.

6.9 Work Package 9

Work package number	WP9	Start date or starting event:				1	
Work package title	Demonstration						
Activity Type	DEM						
Participant number	1	7					
Person-months per participant:	3	1					

6.9.1 Objectives

- O9.1 Hold Annual and Final Demonstrations: WP responsibles should be sure that needed data and tools for holding Annual and Final Demonstrations are collected. Lessons Learned should be part of Demonstrations. Members of every beneficiary should be sent and represent their organizations.

- O9.2 Keep collected data up-to-date: Contributors should be reporting every month and before every milestone. Data should be kept up-to-date and every major change must be reported to the supervisory authorities.

- O9.3 Participate in FM 2021: Project aims to participate in "Formal Methods 2021" conference held in Beijing. Deadlines must be met.

6.9.2 Description of work

- O9.1 Hold Annual and Final Demonstrations

  - T9.1.1: Responsibles should create and present the First Annual Demonstration to the supervisory authorities.
  - T9.1.1: Responsibles should create and present the Second Annual Demonstration to the supervisory authorities.
  - T9.1.1: Responsibles should create and present the Final Annual Demonstration to the supervisory authorities.

- O9.2 Keep collected data up-to-date

  - T9.2.1: Monthly data from contributors shall be collected
  - T9.2.2: Summarize most important data from Lessons Learned report and merge it into Annual Demonstrations
  - T9.2.3: Update and validate last collected before every milestone

- O9.3 Participate in FM 2021

  - T9.3.1: Create Description of Work for FM 2021
  - T9.3.2: Participate in FM 2021
  - T9.3.3: Retrospect participation and make report

6.9.3 Deliverables

- D9.1 First Annual Demonstration: The work done during the first year should contain the achievements. The demonstration should be easily understandable for the authorities and should cover all fields of work. Demonstration should show aims for the next period.

- D9.2 Participation in FM 2021 Conference: The work which was done before conference should contain the achievements. The demonstration should cover importance and goals of work and its influence on state-of-the-art.

- D9.3 Second Annual Demonstration: The work done during the second year should contain the achievements. The demonstration should be easily understandable for the authorities and should cover all fields of work. Demonstration should show aims for the next period.

- D9.4 Final Project Demonstration: The work done during the whole year should contain the achievements. The demonstration should be easily understandable for the authorities and should cover all fields of work. Demonstration should show aims for the next period.

7   Milestones

Milestone number	Milestone description	WP involved	Expected date	Means of verification
M1	NLP for Requirements Analysis	3	13	Laboratory prototype completed and running
M2	Semantic Analyzer	4	16	Laboratory prototype completed and running
M3	Consistency Check	5	18	Laboratory prototype completed and running
M4	Participation in Formal Methods 2021	9	19	Succeeded participation
M5	Verifier	6	30	Laboratory prototype completed and running
M6	Project Completion	1-9	36	Final prototype completed and quality validated

Table 5: Table of Milestones

Acronyms

Aalto Aalto University. 13

ELTE Eötvös Loránd University. 12, 14

KTH Royal Institute of Technology. 11

NLP Natural Language Processing. 2, 5, 6, 8–10, 13, 15, 19

RTC Rational Team Concert. 7

SRS Software Requirements Specification. 2

TUB Technical University Berlin. 9

UCA Université Côte d'Azur. 12, 14

UNITN University of Trento. 14